КЫРГЫЗ РЕСПУБЛИКАСЫНЫН МАМЛЕКЕТТИК ТУУСУ

КЫРГЫЗ РЕСПУБЛИКАСЫНЫН МАМЛЕКЕТТИК ГЕРБИ

КЫРГЫЗ РЕСПУБЛИКАСЫНЫН МАМЛЕКЕТТИК ГИМНИ

Сөзү: *Ж. Садыков, Ш. Кулуевдики*
Муз.: *Н. Давлесов, К. Молдобасановдуку*

Ак мөңгүлүү аска-зоолор, талаалар,
Элибиздин жаны менен барабар.
Сансыз кылым Ала-Тоосун мекендеп,
Сактап келди биздин ата-бабалар.

Кайырма:
Алгалай бер, кыргыз эл,
Азаттыктын жолунда.
Өркүндөй бер, өсө бер,
Өз тагдырың колуңда.

Аткарылып элдин үмүт, тилеги,
Желбиреди эркиндиктин желеги.
Бизге жеткен ата салтын, мурасын,
Ыйык сактап, урпактарга берели.

Кайырма:
Алгалай бер, кыргыз эл,
Азаттыктын жолунда.
Өркүндөй бер, өсө бер,
Өз тагдырың колуңда.

И. Н. Цыбуля, Л. А. Самыкбаева,
А. А. Беляев, Н. Н. Осипова, У. Э. Мамбетакунов

ЖАҢЫ КИТЕП

ИНФОРМАТИКА
7–9-класс

Окутуу кыргыз тилинде жүргүзүлгөн
жалпы билим берүүчү мектептер үчүн окуу китеби

Кыргыз Республикасынын Билим берүү жана илим министрлиги
тарабынан сунушталган

Бишкек – 2020

УДК 373.167.1:002
ББК 73 я 721
И 74

И 74 **Информатика: 7–9-класс**: Окутуу кыргыз тилинде жүргүзүлгөн жалпы билим берүүчү мектептер үчүн окуу китеби / И. Н. Цыбуля, Л. А. Самыкбаева, А. А. Беляев, ж. б.; – Б.: «Сорос-Кыргызстан» Фонду, 2020 – 205 б.

ISBN 978-9967-9270-3-2

Бул окуу китеби жалпы билим берүүчү мектептердин 7–9-класстарынын окуучулары, ошондой эле информатиканын негиздерин, программалоону үйрөнүүнү баштоого даяр бардык курактагы балдарга арналат. Окуу китеби маалыматтык процесстерди терең түшүнүүгө, тармактык технологияларды, булуттук сервистерди коопсуз колдонууга, сайттарды жана роботторду программа аркылуу кандайча башкарууну жана программалоону үйрөнүүгө жардам берет. Окуу китебинин темалары информатиканы үйрөнүүнүн төрт негизги бөлүмүндө ачылып берилет, алар: Информатика жана маалымат, Компьютерлер жана программалык камсыздоо, Компьютердик тармактар жана интернет, Программалоо. Бардык бөлүмдөр материалды үйрөнүүнүн көлөмүн көбөйтүү жана акырындап тереңдетип окутуу менен, ар бир класста кайталанат. Окуу китеби мектеп программасынын алкагында «Информатика» предметин окутууда да, Python программалоо тилин өз алдынча үйрөнүүдө да колдонулушу мүмкүн.

УДК 373.167.1:002
ББК 73 я 721

ISBN 978-9967-9270-3-2

КИРИШҮҮ

Урматтуу достор!

Силердин алдыңарда азыркы жашоонун маанилүү маңызы болгон – маалыматтык технология тармагындагы керектүү билимдерди бере турган окуу китеби турат. Биз бардыгыбыз азыркы учурда нефть жана газ сыяктуу эле баалуу ресурс болуп калган маалыматтын негизинде түзүлгөн маалыматтык коомдун бир бөлүгү болуп эсептелебиз. Силердин милдетиңер ушул баалуу ресурсту туура колдонгонго үйрөнүп, андан максимум пайда алууга жетишүү. Окуу китеби силерди видеофильмдерди жана веб-баракчаларды түзүү технологиялары, маалыматты коддоо принциптери жана компьютердик графика менен тааныштырат. Бул китепти окугандан кийин силер электрондук таблицаларды, презентацияларды, маалыматтар базасын кантип түзүүнү билесиңер, Wi-Fi локалдык тармагын тескегенди жана Python тилинде программалоого үйрөнөсүңөр. Силер өзүңөргө жана айлана-чөйрөгө башка көз караш менен карайсыңар, силер үчүн жаңы жана кызыктуу дүйнө ачылат! Силер өздөштүрө турган заманбап маалыматтык технологиялар айлана-чөйрөгө болгон көз карашыңарды бир топ өзгөртөт. Бул окуу китеби өзүңөрдүн өсүү деңгээлиңерди баалап тургандай болуп түзүлгөн: ар бир теманын аягында теориялык билимиңерди текшерүү үчүн суроолор жана тапшырмалар берилип турат. Андан тышкары, компьютердик практикумдар берилген, алардын жардамы менен өзүңөрдүн практикалык көндүмүңөрдү өнүктүрө аласыңар.

Окуу китебинде төмөнкүдөй шарттуу белгилер кездешет:

ЭСИҢЕ ТУТ
– маанилүү маалымат, аны жакшы эстеп калуу керек.

БУЛ КЫЗЫКТУУ!
– тема боюнча кошумча маалымат.

АНЫКТАМА
– жатка билүү керек болгон теориялык маалыматтар.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР
– окуу китебинин түшүндүрүүчү текстинде өзүн-өзү текшерүү үчүн.

КОМПЬЮТЕРДИК ПРАКТИКУМ
- компьютерде өз алдынча аткаруу үчүн тапшырмалар.

Дүйнө чечкиндүү жана билимге умтулгандарга ачылат – өзүңөргө суроо бергиле, аларга жооп издегиле, өздүк тартипти сактагыла, эксперимент жүргүзүүдөн коркпогула. Силердин колуңардан баары келет!

МАЗМУНУ

7-КЛАСС

8-КЛАСС


Сайттар
КОМПЬЮТЕР ЖАНА ПК

7 - класс


Python
программалоо тили
Интернет
Компьютер адамдын жашоосунда
Маалыматтык процесстер
Логика

1 - бөлүм

Информатика жана маалымат

1.1-тема:

Компьютер адамдын жашоосунда

Айлана-чөйрөнү карагылачы, бизди курчап тургандардын бардыгы: класстагы проектор, үйдөгү муздаткыч, көчөдөгү жол чырак, заманбап мультфильмдер жана онлайн оюндардан тартып атомдук станциялар жана Айдагы роботторго чейин компьютерлердин жардамы менен түзүлгөн жана башкарылат.

Маалыматтык технологиялар (МТ) биздин күнүмдүк жашообузга сиңип, алар турмушубузду ыңгайлуу жана коопсуз кылуу менен кандай гана маселе болбосун аларды тез арада чечкенге жардам берет. Азыр биз көз ирмемде керектүү маалыматты онлайн энциклопедиялардан таап, же болбосо башка өлкөдөгү досубузга билдирүү жөнөтө алабыз.

Учурда жасалма интеллект, виртуалдуу жана толукталган реалдуулук, жасалма органдарды өндүрүү үчүн биотехнологиялар сыяктуу фантастиканын чегиндеги технологиялар кеңири колдонула баштады. Мындан бир канча жыл мурда принтерде үч өлчөмдүү басып чыгаруу деген фантастика эле, азыр болсо бүтүндөй үйдү басып чыгаруучу принтерлер жасалып чыкты.

Ошол эле учурда глобалдуу компьютерлештирүү заманында компьютердин адамдын дене-бой жана психикалык ден соолугуна тийгизген таасири жөнүндө да суроолор жаралууда.

Компьютерде көп отуруу адамдын нервдик, эндокриндик, иммундук системаларынын өзгөрүүсүнө алып келип, анын сөөк-булчуң аппараттарына жана көзүнө терс таасирин тийгизет.

АНЫКТАМА:

Гиподинамия – бул организмдин кыймыл активдүүлүгүнүн төмөндөшү, жалпы дене-бой активдүүлүгүнүн азайышы.

Компьютер жана башка электрондук түзүлүштөр менен иштөөдө адамга тийгизген зыяндуу факторлор:

Технологиялардын өнүгүүсүнүн жакшы жактарына мисалдар:

1. ачык электрондук энциклопедиялардын эркин жеткиликтүүлүгү;
2. билдирүүлөрдү дароо жиберүү;
3. интернетти такси чакырууга, билеттерге буйрутма берүүгө колдонуу;
4. онлайн дүкөндөрдөн буюмдарды сатып алуу;
5. болжолдоо үчүн жасалма интеллектти пайдалануу;
6. реалдуу объекттерди 3Dда басып чыгаруу.

Компьютерде иштөөдө ден соолугуңарга зыян келтирбеш үчүн төмөнкү эрежелерди сактоо керек:

- компьютерде үзгүлтүксүз иштөө режими 20 мүнөттөн ашпашы керек;
- тыныгууларда жеңил көнүгүүлөрдү жасап туруу керек;
- көзүңөр ачышса, көрүүңөр кескин начарласа, колуңардын манжаларында ооруксынууну сезсеңер же жүрөгүңөрдүн тез-тез согушун байкасаңар тез арада иш орунуңарды таштап, мугалимге билдиргиле;
- көздөн экранга чейинки аралык - 50–70 см (сунулган колдун аралыгындай болуш керек);
- моюн кичине ийилген абалда;
- экран көздүн деңгээлинен кичине төмөнүрөөк жайгашышы керек;
- дене түз, ийиндер түшкөн жана эркин абалда болушу керек;
- тизелер жана чыканактар тик бурч менен бүгүлүшү керек;
- отургучтун чети тизенин артына тийбей турушу керек;
- буттар түз жайгашып жана бири-бирине учкаштырылбашы керек.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

Компьютердин адамдын жашоосуна тийгизген оң жана терс таасирлери боюнча таблицаны толуктагыла:

	ОҢ ТААСИРИ	ТЕРС ТААСИРИ
1	Баарлашуунун жеткиликтүүлүгү	Компьютердик көз карандылык
2	Каалагандай маалыматтын жеткиликтүүлүгү	“Керексиз маалыматтар менен “мээни толтуруу”
3	Жумуштарды автоматташтыруу	Экрандан окуу татаалдыгы
4	Жекече окуу мүмкүнчүлүгү	Спам, фрагменттүү окуу
5	Көрсөтмөлүүлүк, презентацияларды түзүү мүмкүнчүлүгү	Реалдуулукту виртуалдуу дүйнө менен алмаштыруу

1.2-тема:

Маалыматтык процесстер жана маалыматты сактоо

Адам маалымат дүйнөсүндө жашайт. Ар бир адам маалыматты өзүнүн эс тутумунда, андан тышкары ар түрдүү сырткы алып жүрүүчүлөрдө: мисалы, кагазда, флешкада же дисктерде жазуу түрүндө сактайт. Маалыматты алуу, сактоо, иштетүү жана берүү менен байланышкан процесстер **маалыматтык процесстер** деп аталат.

Качан гана сезүү органдарыбыздын жардамы менен айлана-чөйрөнү сезгенде - биз *маалыматты кабыл алган* болобуз. Башка адамдар менен баарлашуу процессинде бир эле убакта биз *маалыматты алабыз* жана *беребиз*. Алынган маалыматтарды эстеп калуу менен биз *маалыматты сактайбыз*. Кайсы бир максаттарга жетүү үчүн жана туура чечим кабыл алуу үчүн биз *маалыматты иштетебиз*.

Маалыматтын көлөмүн өлчөө

Сактоо формасы боюнча маалыматты аналогдук жана сандриптик деп экиге бөлүшөт. Эки форманын айырмасы – аналогдук маалымат үзгүлтүксүз, ал эми сандриптик – дискреттүү болгондугунда, б.а өзүнчө элементтердин көптүгүнөн турат. Каалагандай маалыматты сактоо үчүн анын көлөмүн өлчөө зарыл.

Информатикада маалыматты өлчөө үчүн эки негизги ыкма колдонулат:

1. Мазмундук ыкма;
2. Алфавиттик ыкма.

Мазмундук ыкма

Мындай ыкманын жардамы менен алынган маалыматтын көлөмү алынган билдирүүнүн маалыматтык мазмунунан көз каранды. Эгерде билдирүү маалыматтуу болбосо (билдирүүнү алуучу үчүн мааниси жок), анда маалыматтын көлөмү нөлгө барабар болот.

Бул учурда, билдирүүнүн маалымат берүүчүлүгү анын ичиндеги пайдалуу маалыматтын көлөмүнөн көз каранды болот. Мындай билдирүү кандайдыр бир кырдаалдын белгисиздигин толугу менен жокко чыгарат же азайтат.

Мисалы адамдын кийинки аялдамадан түшө тургандыгы жөнүндө маалыматтын белгисиздиги «ооба» же «жок» деген жооптун бир вариантын тандап алуудан турат.

Ыктымалдыгы барабар болгон окуялардагы маалыматтын көлөмүн (i) аныктоо үчүн $N=2^i$ формуласы колдонулат. Мында N – бул мүмкүн болгон окуялардын саны.

1-маселе. Жүргүнчүдөн сурашты: «Сиз кийинки аялдамадан түшөсүзбү?» –«Ооба», - деп жооп берди ал. Жооп канча маалыматты камтыйт?

Чыгаруу: $\mathbf{N = 2^i}$

N=2, («Ооба» же «Жок» деген жооптун варианттарынын мүмкүн болгон саны).

Анда: $\mathbf{2=2^i}$ $\mathbf{i=1}$

Жооп: 1 бит.

1-мисал. Китеп жогорку же төмөнкү текченин биринде турат. Китеп төмөнкү текчеде турат деген билдирүү, белгисиздикти эки эсеге азайтат жана 1 бит маалыматты камтыйт.

2-мисал. Информатика боюнча олимпиадага 4 окуучу катышат. 2-катышуучу эң көп балл алды деген билдирүү, баштапкы белгисиздикти туура 4 эсе (эки жолу экиден) азайтат жана 2 бит маалыматты камтыйт.

Алфавиттик ыкма

Мындай ыкмада маалыматтын көлөмү каалагандай тилдеги (табигый же формалдуу) текстте колдонулган символдордун маалыматтык салмагынан көз каранды болот.

Алфавиттик ыкмада **алфавиттин кубаттуулугу** деген түшүнүк колдонулат, анда алфавиттеги белгилердин жалпы саны, анын ичинде сандар жана тыныш белгилери көрсөтүлөт.

Мисалы: кыргыз тамгаларынын жана колдонулган символдорунун алфавитинин кубаттуулугу 57: 36 тамга + 10 цифра + 11 тыныш белги, кашаа, бош орун.

Эң аз кубаттуулукка компьютерде колдонулган алфавит (машинанын тили) ээ. Аны экилик алфавит деп аташат, анткени ал «0» жана «1» деген 2 белгиге ээ. Экилик алфавиттин символунун маалыматтык көлөмү маалыматты өлчөөнүн бирдиги катары кабыл алынган жана 1 бит деп аталат.

АНЫКТАМАЛАР

Табигый тил – адамдардын ортосунда баарлашуу үчүн колдонулган тил (кыргыз, орус, англис ж.б. тилдер)

Формалдуу тил – билдирүү түзүлүшүнүн так эрежелери менен мүнөздөлгөн жасалма тил: ноталар, Морзе алиппеси, химиялык элементтердин формуласы, программалоо тилдери.

Бит боюнча алфавиттеги бир символдун маалыматтык көлөмүн төмөнкү таблица боюнча эсептөөгө болот:

Варианттардын саны (алфавиттин кубаттуулугу)	2	4	8	16	32	64	128	256	512	1024
Маалыматтагы биттин саны	1	2	3	4	5	6	7	8	9	10

Демек, алфавиттин кубаттуулугунун формуласы төмөнкүдөй:

$$M=2^K,$$

мында **M**-алфавиттин кубаттуулугу,

K – биттин саны (символдун маалыматтык салмагы).

Билдирүүдөгү маалыматтын санын (S) аныктоо үчүн, ошол тексттеги символдордун санын (N) берилген алфавиттеги бир символду коддоо үчүн кеткен биттердин санына (K) көбөйтүү керек:

$$S=N*K.$$

1-маселе. Алфавит 32 тамганы камтыйт. Анын бир тамгасы кандай канча көлөмдөгү маалыматты камтыйт?

Чыгаруу:
Алфавиттин кубаттуулугу M=32.

1) $32=2^5$, демек бир символдун салмагы K = 5 бит болот.

Жообу: 5 бит.

2-маселе. 64 символдук алфавиттин тамгалары менен жазылган билдирүү 20 символду камтыйт. Байт менен бул канча көлөмдөгү маалыматты камтыйт?

Чыгаруу:
Алфавиттин кубаттуулугу M=64.

1) $64=2^6$, демек бир символдун салмагы K=6 бит.
2) 20 символ * 6 бит = 120 бит.
3) 120 бит : 8=15 байт.

Жообу: 15 байт.

3-маселе. Бир уруу 32 символдук, ал эми экинчи уруу 64 символдук алфавитке ээ. Уруулардын башчылары бири-бири менен кат алышты. Биринчи уруунун катында 80 символ, ал эми экинчисиникинде 70 символ камтылган. Каттардагы маалыматтын көлөмдөрүн салыштыргыла.

Чыгаруу:
Биринчи уруу: $2^K = 32$, K = 5 бит – ар бир символду алып жүргөн маалыматтын көлөмү, 80*5=400 бит.
Экинчи уруу: $2^K = 64$, K = 6 бит – ар бир символду алып жүргөн маалыматтын көлөмү, 70*6 = 420 бит.
Жообу: экинчи уруунун катындагы маалыматтын көлөмү көбүрөөк.

4-маселе. 12288 битти камтыган билдирүү канча килобайтты түзөт?

Чыгаруу:
1 килобайт = 1024 байт, 1 байт = 8 бит.
12288 : 8 : 1024 = 1,5 Кб.

Жообу: 1,5 Кб

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Эгерде үй 8 кабаттуу болсо, «Бакыт 5-кабатта жашайт» - деген билдирүүдөн канча бит маалымат алынды?*
2) *16 символдук алфавиттеги 384 символдон турган билдирүү канча килобайтты түзөт?*
3) *Программалоо тилинин алфавити Адан Zке чейинки чоң жана кичине тамгаларды жана арифметикалык амалдардагы белгилерди камтыйт. Бул программалоо тилинин алфавитинин кубаттуулугу кандай?*

1.3-тема:

Тексттик маалыматты коддоо

Каалагандай текст символдордон, ошондой эле тамгалардан (баш же кичине), цифралардан, тыныш белгилерден, атайын символдордон, мисалы «=», «(», «&» ж.б.у.с. жана сөздөрдүн арасындагы боштуктардан турат. Компьютердин эс тутумунда символдордун ордунда алардын номерлери – сандык коддору сакталат.

АНЫКТАМАЛАР

Код – бул маалыматты көрсөтүү үчүн шарттуу белгилердин жана эрежелердин системасы.

Коддоо – бул берилген коддун жардамы менен маалыматты көрсөтүү.

Текстти стандарттык коддоо

Коддук таблица – сандык код жана символдордун ортосундагы дал келүүчүлүктү түзүүчү таблица. Алгач бир символду коддоо үчүн 1 байт (8 бит) колдонулган. Мындай коддук таблица 256га чейин (2^8) символду гана камтыйт. Көп тилдүү тексттерди көрсөтүү үчүн 1991-жылы **Юникод** (англ. ***Unicode***) деп аталган жаңы эл аралык стандарт пайда болгон. Мында 1 символго 2 байт бөлүнөт, ал болсо 65536 символду коддоого мүмкүндүк берет. Ошентип, Юникод стандартынын толук спецификациясы дүйнөдөгү бардык алфавитти өзүнө камтыйт.

2020-жылы Юникод стандартынын 13.0 версиясы кабыл алынган. Ал 143 859 ар түрдүү символдорду камтыйт. Азыркы учурда көп байттуу коддоолор колдонулат (бир символду көрсөтүүгө бир нече байт керектелет). Алардын ичине ASCII коддоо системасы кирет.

ASCII таблицасы

ASCII (англ. ***American Standard Code for Information Interchange***) – маалымат алмашуу үчүн америкалык стандарттык код.

ASCII бул ондук цифраларды, латын жана улуттук алфавиттерди, тыныш белгилерди жана башкаруучу символдорду көрсөтүү үчүн кодго айландырууну түшүндүрөт.

0дөн 32ге чейинки коддор – бул атайын символдор, өзүнөн кийинки сапка өтүү, бош орун, киргизүү ж.б. кодун камтыйт. 33төн 127ге чейинки коддор интернационалдык деп аталат жана латын алфавитине, цифраларга, тыныш белгилерине, арифметикалык амалдарга туура келет.

Улуттук алфавиттин тамгаларын чагылдыруу үчүн 128ден 255ке чейинки коддогу символдорду колдонушат. Улуттук алфавитте бир эле кодго ар кандай символдор дал келет. Ошондуктан текстти туура көрсөтүү үчүн ага тиешелүү коддук баракты (программаларды тескөөдө) орнотуу керек.

1-маселе. Тексттик файл 32 баракtан турат. Ар бир бараkта 40 сап, ар бир сапта 48 символ бар. Ар бир символ 8 бит менен коддолгон КОИ-8 коддоосундагы макаланын көлөмүн аныктагыла.

Чыгаруу:

Макаладагы символдордун санын аныктайбыз: 32*40*48 = 61 440.

Бир символ бир байт менен коддолот. 1024 байт 1 килобайтты түзөт, ошондуктан макаланын маалыматтык көлөмү: 60 Кб.

Жообу: 60 Кб

2-маселе. Юникоддун бир коддоосунда ар бир символ 16 бит менен коддолот. Бул коддоодо төмөнкү сүйлөмдүн өлчөмүн аныктагыла:

Жети өлчөп, бир кес!

Чыгаруу:

Сүйлөмдө 20 символ бар. Демек, Юникод коддоосундагы бул сүйлөмдүн өлчөмү: 20 * 16 = 320 бит.

Жообу: 320 бит.

Кириллицаны чагылдыруу үчүн ASCII коддоосунун ар түрдүү бир канча стандарттары бар, анын ичинде: CP1251 (Windows), KOI-8 (UNIX), MacCyrillic (MacOS) ж.б. (1-тиркемени кара).

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Эгерде браузерде сиздин барагыңыз окулбаган текст менен көрсөтүлүп калса, эмне кылыш керек?*

2) *Эмне үчүн стандарттык ASCII коддоосу бардык алфавиттерди көрсөтүүгө жетишсиз?*

3) *Стандарттык ASCII коддоосунун жардамы менен кытай тилиндеги текстти көрсөтүүгө болобу? Эмне үчүн, түшүндүргүлө?*

4) *KOI-8 коддоосу колдонулуп жатса, төмөнкү фраза эстин кандай көлөмүн ээлей турганын аныктагыла:* ***Молекулалар атомдордон турушат.***

2 - бөлүм

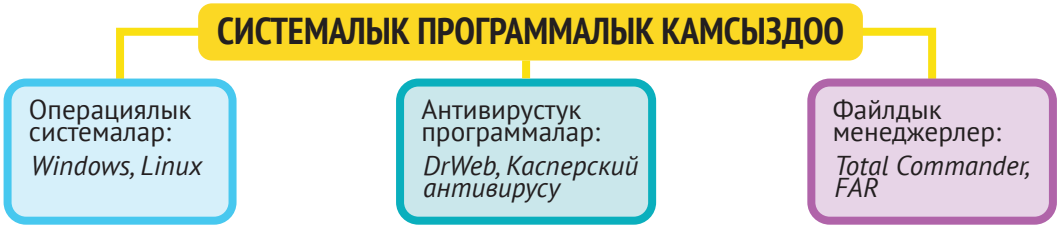
Компьютер жана ПК

2.1-тема:

Программалык камсыздоонун түрлөрү жана курамы

Салттуу түрдө программалык камсыздоону (ПК) үч түргө бөлүшөт: системалык, колдонмо ПК жана программалоо системалары.

Системалык ПК – бул эсептөөчү системанын процессор, байланыш жана перифериялык түзүлүштөр сыяктуу компоненттерин башкаруучу программалардын жыйындысы.



Системалык программалык камсыздоонун эң негизги бөлүгү болуп, төмөндөгү функцияларды аткарган операциялык системасы эсептелет:

- компьютердин ресурстарынын колдонулушун бөлүштүрөт жана дайындайт (процессор, ыкчам сактоочу түзүлүш (ЫСТ), дисктер);
- компьютердин ресурстарын колдонууну жана программаларды колдонуунун убактысын пландайт;
- компьютердин ишин көзөмөлдөйт.

Операциялык системага адам менен компьютердин баарлашуусун камсыз кылган колдонуучунун графикалык интерфейси да кириши мүмкүн. Операциялык системаны башка программалардын иштөөсү үчүн чөйрө деп айтсак да болот.

Көп маселелүүлүк – компьютерде бир эле убакта бир нече ишти аткарууга мүмкүндүк берүүчү механизм: мисалы, музыка угуу жана ошол эле учурда тексттик редактордо иштөө.

Көп маселелүүлүктү камтыбаган система

Көп маселелүүлүктү камтыган система

Көп маселелүүлүктү камтыган системаларда компьютер көбүрөөк натыйжалуу колдонулат.

Виртуалдуу эс тутумдун механизми сырткы эс тутумдун бир бөлүгүн бөлүүгө мүмкүндүк берет (катуу дисктен же башка алып жүрүүчүдөн), бул бөлүктү андан ары система ЫСТтын уландысы катары карайт. Жыйынтыгында компьютер чоң көлөмдөгү ЫСТ менен иштей алат.

Колдонмо программалык камсыздоо – бул колдонуучунун компьютердеги конкреттүү маселелерди аткаруусун камсыз кылуучу программалар.

Азыркы кезде кеңири тараган колдонмо ПКнын бири болуп, офистик документтер менен иштөөгө арналган OpenOffice.org эркин программалардын жыйындысы эсептелет. Ал тексттик документтер, электрондук таблицалар, презентациялар, вектордук сүрөттөр, маалыматтар базасы менен иштөөгө мүмкүндүк берет.

Программалоо системалары программаларды долбоорлоо жана иштеп чыгуу үчүн кызмат кылат. Мисалы, аларга Python тилинде программалоо үчүн арналган IDLE иштетүү чөйрөсү кирет.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Силер колдонгон колдонмо программаларды, алардын аткарган кызматтарын кошуп санап бергиле.*
2) *Компьютердин иштеши үчүн кайсы системалык программалар сөзсүз керек?*
3) *Жаңы принтерди компьютерге туташтыруунун алгортми кандай?*

2.2-тема:

Электрондук таблицалар

Чоң көлөмдөгү сандык маалыматтарды иштетүү, талдоо жана сорттоо үчүн электрондук таблицалар (ЭТ) колдонулат.

Алардын жардамы менен силер төмөнкүлөрдү аткара аласыңар:

- ар башка уячада эле эмес, ар башка бетте жайгашкан берилиштерди бириктирип эсептөөлөрдү жүргүзө аласыңар;
- график жана диаграммалардын жардамы менен маалыматтарды визуалдаштыра аласыңар.

Электрондук таблица төмөнкүлөрдү камтыган **Баракчалардан** турат: **Мамычалар** (латын алфавитинин тамгалары менен белгиленет) жана **Саптар** (цифралар менен белгиленет).

Сап менен мамычанын кесилиши **Уячаны** түзөт. Активдүү уячанын дареги аталыштар талаасында көрсөтүлөт, мисалы, A1. Уячаларда сандар, формулалар же текст жайгашышы мүмкүн.

Уячалардын тобу **Диапазонду** түзөт. Ал кош чекит менен бөлүнүп жазылат, мисалы, A1:B5. ЭТнын элементтерин бөлүп алуу ыкмаларын силер 2-тиркемеден көрүп алсаңар болот.

Электрондук таблицаларда берилиштерди киргизүү өзгөчөлүктөрү

Берилиштер активдүү уячага киргизилет. Ушул уячага маалыматтарды кайталап киргизүүдө буга чейин киргизилген маалымат өчүп калат, ошондуктан редакциялоо үчүн F2 баскычын басуу керек.

Эгерде киргизилип жаткан тексттин узундугу уячанын энинен чоң болсо, анда ал саптын бөлүгү гана көрүнүп турат.

БУЛ КЫЗЫКТУУ!

Эгерде бир нече бараткы бөлүп көрсөтсөк, силер бардык бөлүнгөн барактарга бир учурда маалыматты кошо аласыңар жана форматтасаңар болот.

	A	B	C	D	E	F
1	Эгерде кошуна уячалар бош болсо, анда сап толугу менен көрүнөт					
2	Эгер тол›	Саптын бөлүгү эле көрүнөт				
3	Ташымалды колдонсо болот.	Уячаны чоюп коюуга да болот				

Маалыматты толугу менен көрсөтүү үчүн уячанын энин өзгөртүү керек же сапты ташыганга уруксат берүү керек.

Электрондук таблицада берилиштерди сорттоо жана чыпкалоо

Эгерде мамычада бир типтеги берилиштер жайгашса (текст, сандар же даталар) аспаптар панелиндеги баскычтарды же **Берилиштер** менюсундагы **Сорттоо** командасын колдонуп, аларды өсүү же кемүү тартибинде сорттоого болот.

Фильтр

Фильтр (чыпка) экранга берилген шартка дал келген гана берилиштерди чыгарууга мүмкүндүк берет. Тандалган диапазонго чыпканы орнотууну **Берилиштер** менюсунун же Аспаптар панелиндеги баскычтын жардамы менен жүргүзүүгө болот.

Шарттуу форматтоо берилген шартка дал келген берилиштерди шрифттин гарнитурасын, өлчөмүн же түсүн өзгөртүү аркылуу бөлүп көрсөтүү үчүн колдонулат. Ал үчүн:

1. **Формат** менюсунан **Шарттуу форматтоо** командасын тандап алгыла.
2. Шартты уячанын маанисине же формулага колдонгула.
3. Жасалганын стилин орноткула.

Формулаларды киргизүү

Формуланы киргизүү ар дайым барабардык белгиси «=» менен башталат, андан соң формуланын өзү жазылат. Мисалы *=A4+16*. Мисалы, эгер *A4* уячасына биз 20 санын жазсак, анда *B4*кө *=A4+16* формуласын жазып, *Enter* баскычын бассак, *B4* уячасында 36 саны пайда болот.

Формулалар сабында учурда колдонуп жаткан формула жазылып турат жана аны редакциялоого болот. Татаал туюнтмаларды жазууда «+», «-», «*», «/», «^» белгилерин колдонсо болот. Мисалы: *=A1+C5*B4*

Салыштырмалуу жана абсолюттук шилтемелер

Формулаларда уячалардын дарегине шилтемелер колдонулат. Шилтемелерди салыштырмалуу, абсолюттук жана аралаш деп бөлүшөт.

Салыштырмалуу шилтемелерде формуланы көчүрүүдө баштапкы берилиштердеги шилтемелер өзгөрөт (мисалы A1, B3).

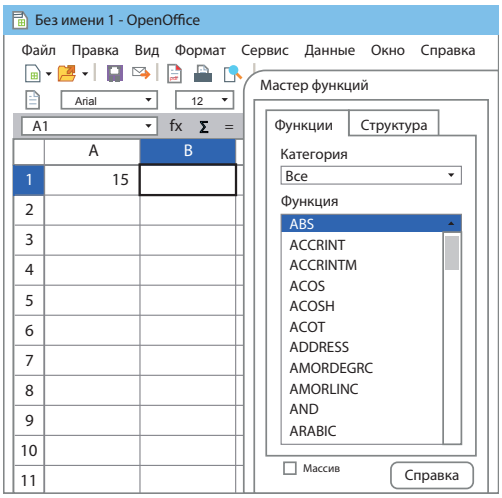
Абсолюттук шилтемелерде формуланы которууда же көчүрүүдө баштапкы берилиштери менен уячанын белгиленген дареги колдонулат. Ал шилтеме доллар белгиси менен белгиленет (мисалы A1).

Аралаш шилтемелерде болсо абсолюттук катары мамычанын дареги (мисалы, $A1) же сапчанын дареги (мисалы, A$1) эсептелет.

Функциялар

Электрондук таблицаларда төмөнкү категориялардагы функциялардын кеңири жыйындысы жеткиликтүү:

- маалыматтар базасы менен иштөөчү;
- дата жана убакытты иштетүүчү;
- финансылык;
- маалыматтык;
- логикалык;
- математикалык;
- массивдер менен иштөөчү;
- статистикалык;
- тексттик;
- кошумча.



Мисалы, «математикалык» категориясынан ***СУММ*** деген функцияны тандап, бөлүнүп алынган диапазондогу маанилердин суммасын эсептесек болот. «Статистикалык» категориясынан ***СРЗНАЧ*** функциясын тандап, белгиленген диапазондогу маанилердин орточо маанисин чыгарсак болот.

Диаграммалар

Электрондук таблицаларда жайгашкан сандык маалыматтарды көрсөтмөлүү кылуу үчүн диаграммаларды жана графиктерди колдонушат.

Open Office Calc колдонуучунун белгилүү маселелерин чечүү үчүн ар түрдүү типтеги диаграммаларды сунуштайт:

- айланма
- гистограмма
- сызыктуу
- чекиттик ж.б.

Диаграмманын негизги элементтери

- Диаграмма аймагы
- Түзүү аймагы
- X горизонталдык огу – категориялар огу
- Y вертикалдык огу – маанилер огу
- Берилиштер катары
- Легенда
- Диаграмманын аталышы
- Октордун аталыштары
- Берилиштердин жазылышы
- Берилиштер чекити

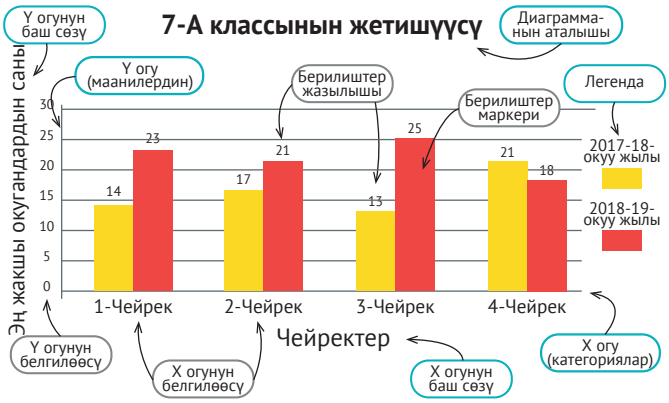


Диаграмма түзүүнүн мисалы:

Үлгү боюнча үй-бүлө мүчөлөрүнүн дем алыш күндөрдөгү чыгымдарынын таблицасын түзөлү.

1-кадам. Диаграмма түзүү үчүн берилиштер турган диапазонду бөлүп алышыбыз керек.

2-кадам. ***Вставить*** менюсунан ***Диаграммы*** командасын тандайбыз.

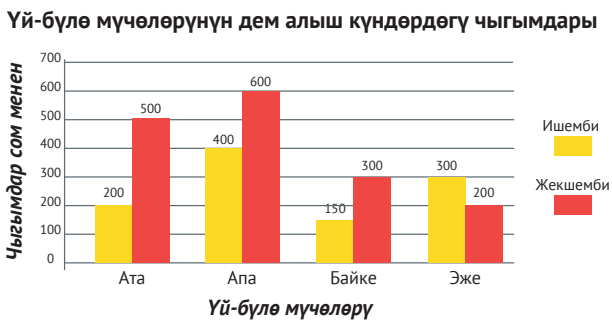
Дем алыш күндөрүндөгү үй-бүлө мүчөлөрүнүн чыгымдары

Аты	Ишемби	Жекшемби
Ата	200	500
Апа	400	600
Байке	150	300
Эже	300	200

3-кадам. Ачылган ***Мастер диаграмм*** терезесинде диаграмманы жайгаштыруу ордун, анын тибин (гистограмма) жана аталышын көрсөтөбүз.

4-кадам. Берилиштер катарынын аталыштарын, тор сызыктарын көрсөтүүнү тандап, легендалардын жайгашкан ордун көрсөтөбүз.

Диаграмманы алабыз ⟶



Диаграмманын параметрлерин өзгөртүү үчүн тиешелүү объектке чычкандын оң баскычын (ЧОБ) басып, контексттик менюдан керектүү команданы тандап алгыла.

Эгерде берилиштерди өзгөртсөңөр, анда диаграмма автоматтык түрдө өзгөртүлүп түзүлөт.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Таблицада учурдагы окуу жылына жылнаама түзгүлө.*
2) *Электрондук таблицада Пифагордун таблицасын түзгүлө.*
3) *Мектеп баскетболчулар командасы 12 окуучудан турат. Эгерде окуучулардын аттары жана алардын бойлорунун узундугу белгилүү болсо, окуучулардын боюнун өсүүсү боюнча диаграмма түзгүлө.*
4) *Музейдин бир жумалык кирешесин эсептегиле, эгер төмөнкүлөр белгилүү болсо:*
- ар бир күнү сатылган билеттердин саны;
- чоңдордун билетинин баасы – 40 сом;
- балдардын билетинин баасы чоңдордукуна караганда 25% га арзан.
Музейдин күнүмдүк кирешеси боюнча диаграмма (график) түзгүлө.
5) *«ЕСЛИ» функциясын колдонуу менен ар бир абонент электр энергиясы үчүн канча төлөөрүн аныктагыла. Төмөнкү шартты эске алгыла: биринчи 700 кВт/саат үчүн абонент 0,77 сом, ал эми андан ашса, 1 кВт/саат электр энергиясына төлөм 2,16 сомго чейин көбөйөт.*

	A	B	C	D	
1	*№*	*Кардар*	*Электр энергиянын саны*	*Төлөм*	
2	2	Асанова	135		
3	3	Давыдов	79		

2.3-тема:

Презентациялар

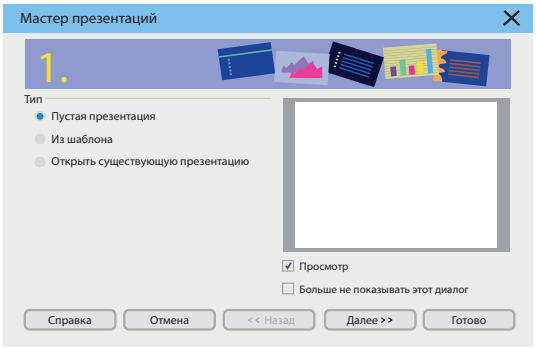
OpenOffice.org Impress программасы текстти, графикалык объектерди, диаграммаларды, анимацияларды, мультимедиа жана башка элементтерди колдонуп презентация түзүүгө мүмкүндүк берет.

OpenOffice.org Impress программасынын мүмкүнчүлүктөрү:

- эффекттер менен шаблондун негизинде слайддарды түзүү (анимация, өтүү эффекттери);
- ар түрдүү макеттердин негизинде слайддарды түзүү;
- диаграммаларды колдонуу менен презентация түзүү;
- презентацияларды автоматтык же кол режиминде демонстрациялоо.

«Мастер презентаций» терезесинде презентация түзүү

OpenOffice.org Impress программасын ишке киргизүүдө экранда «Мастер презентаций» терезеси пайда болот. «Мастер презентаций» шаблондорду жана редакциялоо үчүн ар кандай мүмкүнчүлүктөрдү колдонуу менен презентация түзүүгө жардам берет.

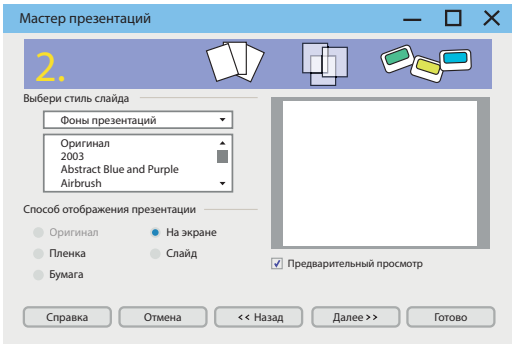


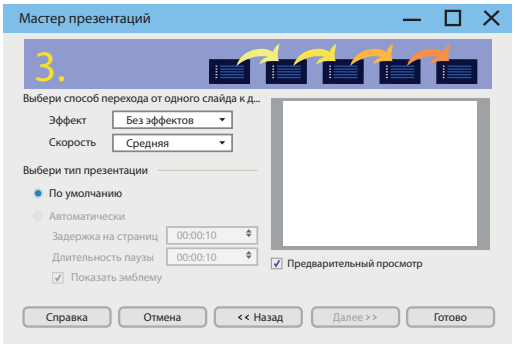
1-кадам. «Мастер презентаций» терезесинде презентациянын тибин тандагыла:

- бош презентация – жаңы презентацияны түзөт;
- шаблондон – шаблонду колдонуп жаңы презентацияны түзөт;
- бар презентацияны ачуу – мурда сакталган презентацияны ачат.

2-кадам. Кийинки терезеде «Выберите стиль слайда» группасында жогорку тизмеден дизайндын эки стилинин бирин тандагыла: презентацияны же презентация фонун.

3-кадам. «Способ отображения презентации» группасынан чагылдыруунун бир тибин тандагыла: *Оригинал, Пленка, Бумага, На экране* же *Слайд* жана «Далее» баскычын басуу керек.

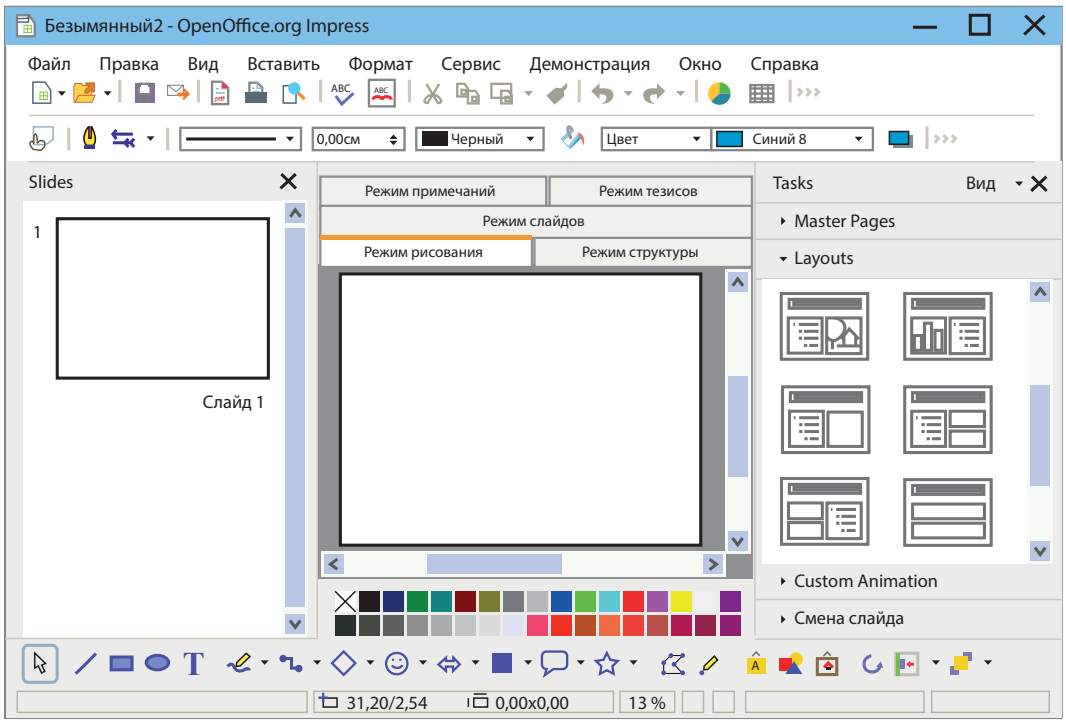




4-кадам. «Выберите способ перехода от одного слайда к другому» группасынын «Эффект» тизмесинен презентация үчүн эффектти тандагыла. «Скорость» тизмесине эффекттин иштөө ылдамдыгын орноткула, ал эми «Выберите тип презентаций» группасынан презентациянын убактысын аныктагыла.

Ошентип, мастердин жардамы менен презентация түзүү аяктайт. Аягына чыгуу үчүн «Готово» баскычына басуу керек.

Силердин алдыңарда презентацияны форматтоо үчүн терезе ачылат.



Слайддарды форматтоо

Слайддарда тексттерди (шрифтти, өлчөмүн өзгөртүү, түздөө ж.б.) жана графикалык объекттерди (өлчөмдөрүн, жарыктыгын, агуучулугун өзгөртүү) форматтоо каалагандай тексттик редакторлордо жүргүзүлгөндөй эле ишке ашырылат.

Титулдук слайдды жасалгалоо үчүн «Заголовок, слайд» макетин орнотуп, презентациянын баш сөзүн киргизгиле.

Кийинки слайдды кошуу үчүн «Вставка» менюсунан «Слайд» командасын тандоо керек же ошол слайддан ЧОБду басып, чыккан контексттик менюдан «Новый слайд» командасын тандап алуу керек.

Слайдга тексттик талаа коюу үчүн чычкан менен T пиктограммасын басуу керек. Андан соң талааны коё турган орунга ЧСБны басып коюу керек.

Графикалык объектти коюу үчүн «Вставка-Изображение-Из файла» менюсун колдонушат.

Презентация тексттерден анимация кошуу мүмкүнчүлүгү менен айырмаланат. Графикалык объекттерге жана текстке маселелер панелиндеги «эффекты» группасынан ар кандай эффекттерди кошууга болот.

Презентацияны демонстрациялоо үчүн Аспаптар панелиндеги Демонстрация баскычын, же F5 баскычын басуу керек.

Демонстрацияны каалаган жеринен бүтүрүү үчүн, же аягына чыгуу үчүн ESC баскычын же чычкандын баскычын басуу керек.

Презентацияны сактоодо файлдын тибине көңүл бургула. Эгерде силердин презентацияңар башка программаларда да мисалы, Microsoft Power Pointта ачылсын десеңер, файлдын тиби деген талаада Microsoft Power Point – версия –(.ppt) деп көрсөтүү керек.

Презентацияны видео форматта сактоо үчүн өзүңөрдүн ишиңерди башка форматка экспорттоо керек, ал үчүн «Файл» менюсунан «Экспорт» командасын тандоо керек.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Анимацияны колдонуп «Космонавтиканын тарыхы» деген темада презентация түзгүлө жана аны фильм түрүндө сактагыла.

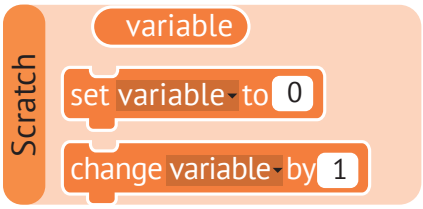
3 - бөлүм

Программалоо

3.1-тема:

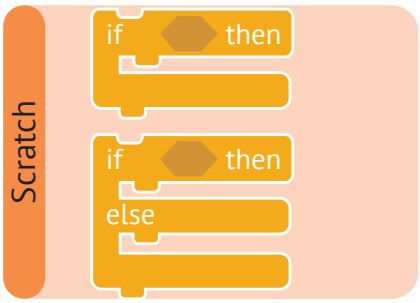
Python программалоо тили

Python программалоо тили – бул жаңы үйрөнүп жаткандар үчүн да жеткиликтүү болгон, ар түркүн багыттагы программаларды түзүү үчүн колдонулган аспап. Эгерде силер Scratchте блоктор менен программа түзүп жүргөн болсоңор, анда Python тилинде программа түзүү силерге оңой эле болот. Келгиле, командалар Python жана Scratch тилдеринде кандай жазыла тургандыгын салыштыралы.




```
variable = 0

variable = variable +1
```
Python

Шарттар ушундай көрүнөт:




```
if True :
 #бир нерсе жасоо керек

if True :
 #бир нерсе жасоо керек
else:
 #бир нерсе жасоо керек
```
Python

Python коду оңой окулат, ал эми интерактивдүү кабыкчасы болсо программаны киргизип, дароо жыйынтыгын алганга мүмкүндүк берет.

Бүгүнкү күндө бул программалоо тилинде банктар, телекоммуникациялык компаниялар үчүн программалар жазылат, көптөгөн аналитиктер маалыматтар менен дал ушул тилдин жардамында иштешет. Өтө жөнөкөй синтаксистин натыйжасында бул тилде программалоону баштоо жеңил.

Python тилинде программалоо үчүн өзүңөрдүн компьютериңерге акысыз программалоо чөйрөсүн орнотушуңар керек.

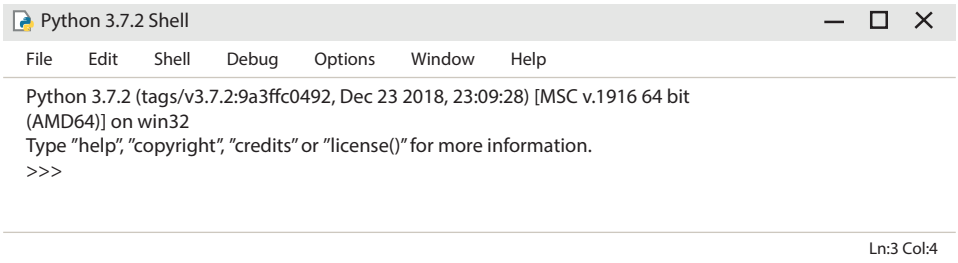
Аны ***https://www.python.org/downloads/*** сайтынан жүктөп алсаңар болот.

Python менен бирге IDLE – Python-программаларын жазуу үчүн иштетүү чөйрөсү да орнотулат.

Python-командаларын жазуу, сактоо жана аткаруу үчүн төмөнкү кадамдарды жасоо керек болот:

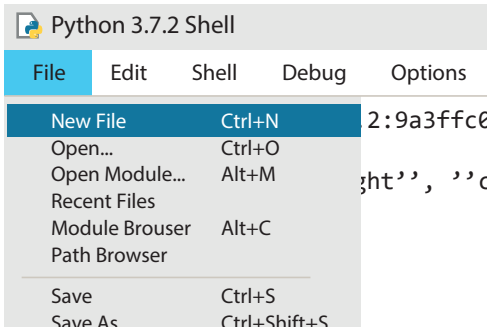
1 -кадам. IDLEни ишке киргизүү

Түзгөн программаңардын жыйынтыгын көрүп туруу үчүн консоль терезеси ачылат.



2 -кадам. Жаңы файлды түзүү

Жаңы программаны жазуу үчүн өзүнчө файл түзүү керек. Ал үчүн **File** менюсунан **New Fileды** тандап алабыз.



3 -кадам. Программаны киргизүү

Ачылган терезеге өзүңөрдүн программаңарды киргизгиле, мисалы:

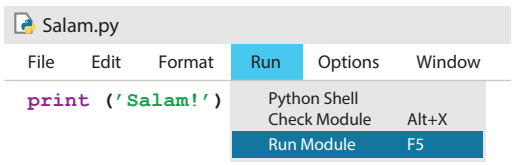
```
print('Salam!')
```

4 -кадам. Программаны сактоо

File менюсунан **Save As** дегенди тандап, файлыңар үчүн жаңы аталышты киргизгиле жана **Save** командасын баскыла. Биз алдын ала My scripts папкасын түзүүнү жана ошол жерге өз программаңардын бардыгын сактоону сунуштайбыз.

5 -кадам. Программаны ишке киргизүү

Run менюсунан **Run Module** командасын тандагыла (же тез иштетүү үчүн F5 баскычын баскыла)



Даяр! Консолдо силер төмөнкү маалыматты көрүшүңөр керек:

```
>>>
Salam!
```

Программа алгач файлга жазылып (адатта **.py** кеңейтилиши менен), андан кийин толугу менен аткарылган ыкма *программалык режим* деп аталат. Мындай программалык файл Pythondo скрипт деп аталат (англ. ***Script*** – сценарий).

Pythondo андан тышкары бир да операторду (команданы) камтыбаган бош программалар да болот. Көбүнчө булар комментарийлер (адатта # белгисинен кийин жазылат) – интерпретатор менен иштетилбеген түшүндүрмөлөр.

```
# бул бош программа
```

АНЫКТАМА

Интерпретатор – бул силердин программаңарды окуп, андагы камтылган нускамаларды аткаруучу программа.

Маалыматтарды киргизүү жана чыгаруу

Көп учурда программа колдонуучудан берилиштерди киргизүүнү сурайт, андан соң аларды иштетип, жыйынтыгын экранга чыгарат. Ал үчүн берилиштерди киргизүүчү жана чыгаруучу функциялар колдонулат:

`print()`	`input()`
`print` функциясы маалыматты экранга чыгарат. Ал өзгөрмөлөрдүн маанилерин да, туюнтмалардын маанилерин да чыгарат.	`input` функциясы болсо тескерисинче, колдонуучуга программа үчүн берилиштерди киргизүүгө мүмкүнчүлүк берет.

*Кашааларда бул функция иштей турган маанилер көрсөтүлөт.

Берилиштерди экранга чыгаруу жана программанын ишинин жыйынтыгын көрүүгө мүмкүндүк берүү үчүн `print()` функциясы колдонулат. Экранга функциянын кашаасында жазылган маалымат чыгарылат. Мында символдорду тырмакчага алып коюу керек.

```
>>> print (2020)
2020
>>> print ('Salam!')
Salam!
```

Кашаанын ичине математикалык туюнтма жазылса, анда экранга анын жыйынтыгы чыгарылат:

```
>>>print (5+6*3)
23
```

Өзгөрмөлөр

`print` функциясына сан жана символдордон тышкары өзгөрмөлөрдү да берүүгө болот. Өзгөрмө – бул аталышка жана кандайдыр бир мааниге ээ болгон эс-тутумдун уячасы (ячейкасы). Мисалы:

АНЫКТАМА

Өзгөрмө – бул аталышка ээ болгон эс-тутумдун эң кичине бөлүкчөсү. Андагы сакталган берилиштер өзгөрмөнүн мааниси деп аталат.

```
x=5
```

Өзгөрмөгө маанини ыйгаруу үчүн «=» ыйгаруу оператору колдонулат. Жогорудагы мисалда биз **x** өзгөрмөсүнө **5** деген маанини ыйгардык. Өзгөрмөнүн маанисин өзгөртсөк болот, мисалы, x=8 деп жазсак болот. Анда улам кийинки жазууларда ушул эң акыркы ыйгарылган маани колдонулат. Мисалы эгер **x=x+1** деп жазсак, анда **x** тин баштапкы маанисине 1 кошулат. Жаңы маани кайра эле ошол өзгөрмөгө жазылып калат.

ЭСИҢЕ ТУТ

Өзгөрмөнүн аты цифра менен башталбаш керек, анткени трансляторго кайсы жерден цифра, ал эми кайсы жерден өзгөрмөнүн аты баштала тургандыгын айырмалоо кыйын болуп калат.

Өзгөрмөлөрдүн атында латын тамгаларын (кичине жана баш тамгалар айырмалуу болот), цифраларды жана астына сызылган сызыкты («_») колдонсо болот.

Мүмкүн болушунча өзгөрмөлөрдү алар кандай роль аткара тургандыгын дароо түшүнө тургандай аташ керек. Мисалы: **adress, tel, summa** ж.б.

Клавиатура аркылуу **берилиштерди киргизүү** үчүн `input()` функциясы колдонулат. Эгерде программада бул функция кезиксе, анда программа ошол жерден өзүнүн иштөөсүн токтотуп, качан колдонуучу кандайдыр бир маалыматты киргизгенче күтүп турат. Мисалы:

```
>>> x = input()          #x тин маанисин киргизүүнү сурайт
35                       #35 деген маанини киргизели
>>> print(x)             #x тин маанисин экранга чыгарат
35
```

Бул мисалдан биз көрүп тургандай **x** өзгөрмөсүнө **35** деген маани ыйгарылды.

Колдонуучуга программа андан эмне талап кылып жаткандыгы түшүнүктүү болуш үчүн кашаанын ичине тырмакчага тексттик билдирүүнү жазып койсо болот. Анда программа төмөнкүдөй болот:

```
x = input('Бүтүн санды киргизгиле: ')
```

1-маселе. Колдонуучудан эки санды сурап, аларды кошуп, жыйынтыгын чыгарып берген программаны түзүп көрөлү. Алгач файл (скрипт) түзөлү жана аны task1.py аты менен сактайлы. Эми программаны жазалы:

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`x = input ('Бүтүн сан киргизиңиз:')`
`y = input ('Бүтүн сан киргизиңиз:')`
`z = x + y`
`print (z)`
	Бүтүн сан киргизиңиз: 3
	Бүтүн сан киргизиңиз: 5
	35

Биз көрүп тургандай эки сан кошулган жок: программа аларды биринин артына экинчисин жазып, бириктирип эле койду. Анткени мындай жазууда `input` оператору киргизилген маанилерди сан катары эмес, символ катары кабыл алат.

Бул катаны оңдоо үчүн маанилерди киргизүүдө алынган символдук сапты бүтүн санга өзгөртүп түзүү керек. Ал үчүн **int** (англ. *integer* – бүтүн) функциясы колдонулат:

```
x = int (input())
y = int (input())
```

Эки санды эки сапта эмес бир сапта бош орун менен ажыратып киргизүү үчүн программаны мындай жазса болот:

```
x, y = map (int, input().split())
```

Ал үчүн биз төмөнкү функцияларды колдондук:

`map()` – берилген функцияны тизменин бардык элементтерине тиешелүү кылат. Б.а. int функциясы эми колдонуучу киргизген бардык берилиштерге колдонулат. Ал эми жаңы маанилер тиешелүү түрдө **x** жана **y** өзгөрмөлөрүнө жазылат.

`split()` – бош орун аркылуу бир сапка киргизилген берилиштерди тизменин өзүнчө элементтерине бөлөт. Б.а. колдонуучу берилиштерди «**3 5**» деп киргизет. Программа бул эки цифраны эки өзүнчө элемент катары карайт.

Каралган функцияларды эске алуу менен программаны кайра жазалы:

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`x, y = map (int, input().split())`
`z = x + y`
`print (z)`
	3 5
	8

Эми программа туура иштеп жатат – ал колдонуучу киргизген эки санды кошту. Бирок мунун эки кемчилиги бар:

1) берилиштерди киргизүүдө колдонуучудан эмне талап кылынып жаткандыгы билинбейт (канча, кандай сандарды киргизүү керек);
2) жыйынтыгы эч нерсени билдирбеген сан түрүндө чыгат.

Ошондуктан, программа менен колдонуучунун ортосунда диалогду түзүү үчүн программаны мындайча өзгөртсө болот:

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`print ('Эки бүтүн сан киргизиңиз:')`
`x,y = map(int, input().split())`
`z = x + y`
`print (x,'+',y,'=',z,sep='')`
	Эки бүтүн сан киргизиңиз:
	3 5
	3+5=8

Көрүнүп тургандай биз `print` функциясында экранга чыгаруу керек болгон бардык элементтерди көрсөттүк: үч өзгөрмөнүн маанилерин жана «+», «=» эки символун. Адатта чыгарылуучу элементтердин арасына бош орун коюлат, ал экрандагы эсептөөнүн жазылышын чоюп салат. Бош орунду алып салуу үчүн анын ордуна каалагандай символду коё ала турган `sep` функциясын колдонобуз. Биз көрсөткөн, `sep=''` бардык бош орундарды алып салууга мүмкүндүк берет.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *30 окуучунун ичинен сабакта 24 окуучу эле отурат. Бул класстагы катышуу пайызын эсептөөчү программаны түзгүлө.*
2) *Убакыт интервалы саат, минута жана секунда менен берилген. Ошол эле интервалды секунда менен көрсөтө турган программаны түзгүлө.*

3.2-тема:

Маалыматтардын тиби жана алар менен болгон амалдар

Өзгөрмөлөрдү колдонуудан мурда берилиштердин типтерин аныктайлы.

Берилиштердин типтери

Python тилиндеги берилиштердин негизги типтерин карап чыгалы:

`int` – бүтүн сан, мисалы: 1; 6; 35;
`float` – чыныгы сан маанилери (бөлчөктүү сан), мисалы 2.5; 70.82;
`bool` – логикалык маанилер, **True** (чындык, «ооба») же **False** (жалган, «жок»);
`str` – бир же кош тырмакчага алынган символдордун жыйындысы.

Өзгөрмөгө ыйгарылган маанисине карата Python тилиндеги программа берилиштин тибин өз алдынча аныктай алат. Мисалы, төмөндөгүдөй жазсак:

```
a = 70.82   #программа берилиштин тибин float деп аныктайт
b = '5'     #программа берилиштин тибин str деп аныктайт
```

Ар кандай типтеги өзгөрмөлөрдү бириктирүү аракетинде (мисалы, c=a+b) программа катаны чыгарат (TypeError).

b = `'5'` – мында биз тырмакчаны коюу менен анын тибин өзгөрттүк. Эми программа бул маанини 5 цифрасы катары эмес, символ катары түшүнөт.

int(), float(), bool() методдору берилиштердин тибин өзгөртөт:

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`a = 10`
`b = 3`
`c = a/b`
`print ('c =', float (c))`
	c = 3.3333333333333335

`a = 10`
`b = 3`
`c = a/b`
`print ('c =', int (c))`
	c = 3

`a = 10`
`b = 3`
`c = a/b`
`print ('c =', bool (c))`
	c = True

Арифметикалык туюнтмалар жана амалдар

Pythondo сандар менен каалагандай арифметикалык амалдарды аткарса болот. Арифметикалык амалдардын белгилери:

+ кошуу,
- кемитүү,
***** көбөйтүү,
/ бөлүү,
****** даражага көтөрүү (x^2 мындай жазылат: x**2)

Арифметикалык туюнтмалар сапка жазылат:

```
x = (3 + y * 2) / 5
```

Амалдардын ирети математикада колдонулгандай эле: даражага көтөрүү, көбөйтүү жана бөлүү кошуу жана кемитүүдөн мурун, ал эми кашаанын ичиндеги амалдар анын сыртындагы амалдардан мурун аткарылат.

Өзгөрмөлөргө жаңы маанини ыйгарууда кыскартылган жазуулар көп колдонулат:

Кыскартылган жазуу	Толук жазуу
`a = b = 0`	`b = 0` `a = b`

`a += b`
`a -= b`
`a *= b`
`a /= b`
	`a = a + b`
	`a = a - b`
	`a = a * b`
	`a = a / b`

Python тилинде арифметикалык амалдарды аткаруу үчүн эки жаңы операторду (белгини) карайлы:

// – сандарды бөлүүдө сандын бүтүн бөлүгүн алуу үчүн колдонулат.
% – бөлүүдө сандын калдыктуу бөлүгүн алуу үчүн колдонулат.

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`a = 31`
`b = a // 7 #=4`
`print (b)`
`c = a % 5  #=1`
`print (c)`
	4
	1

1-маселе. Негизинин узундугу жана бийиктиги белгилүү болгон үч бурчтуктун аянтын табуучу программа түзгүлө.

Геометриядан белгилүү болгондой, үч бурчтуктун аянты үч бурчтуктун негизин (a) анын бийиктигине (h) көбөйткөндүн жарымына барабар:

$$S = \frac{1}{2} ah$$

Бул формуланы мындай жазсак да болот: ***s=(a*h)/2***

Эми программаны түзөлү жана үч бурчтуктун негизинин узундугун **6 см** жана бийиктигин **4 см** деп киргизели.

Python тилиндеги программа

```
a = float(input('Негизинин маанисин киргизиңиз: '))
h = float(input('Бийиктигинин маанисин киргизиңиз: '))
s = (a*h)/2
print ('Жообу: s=',s)
```

Экранга чыккан жыйынтыгы

```
Негизинин маанисин киргизиңиз: 6
Бийиктигинин маанисин киргизиңиз: 4
Жообу: s= 12.0
```

math модулу

Python тилиндеги көптөгөн стандарттык функциялар кызматтары боюнча топторго – **модулдарга** бөлүнгөн. Мисалы, математикалык функциялар **math** модулунда, сүрөт тартуу үчүн функциялар **turtle** модулунда, кокустук сандар менен иштеген функциялар **random** модулунда топтолгон. Керектүү функцияларды колдонордон мурда тийиштүү модулду жүктөп алуу керек. Ал үчүн `import` командасы колдонулат. Мисалы, математикалык модулду кошуу үчүн программага төмөнкү кодду киргизебиз:

```
import math
```

Ал эми функцияларга кайрылуу үчүн: алгач модулдун атын, чекиттен кийин функциянын атын көрсөтүү керек:

```
print (math.ceil(x))
```

Python тилиндеги программа

```
import math
x = 25.6
print (math.ceil(x))
```

Экранга чыккан жыйынтыгы

```
26
```

Төмөндө сандар менен иштөөчү кээ бир функциялар көрсөтүлгөн:

Команда	Аткарылуучу аракет
`int(x)*`	x чыныгы санын калдык бөлүгүн алып салуу менен бүтүн санга айландыруу
`round(x)*`	x чыныгы санын жакынкы бүтүн санга чейин тегеректөө
`ceil(x)`	жакынкы чоң бүтүн санга чейин тегеректөө
`floor(x)`	төмөнкү кичинекей бүтүн санга чейин тегеректөө
`abs(x)*`	сандын модулун эсептөө (абсолюттук чоңдугун)

* – *стандарттык функциялар,* ***math*** *модулун кошууну талап кылбайт.*

Кокустук сандар. random модулу

Кээде программаларда, мисалы, компьютердик оюндарда же лотереяларда кандайдыр бир кокустук санды алуу керек болот. Pythonдо кокустук сандарды алуу үчүн **random** модулу колдонулат. Бул модулдагы **randint** функциясы бүтүн кокустук санды генерациялоого жардам берет. Биз билгендей, алгач модулдун аты, андан кийин чекит менен бөлүнүп функциянын аты жана анын аргументтери жазылат.

```
import random
print (random.randint (1, 20))
>>>
7
```

`randint()` функциясы кашаадагы эки сандын арасындагы диапазондо кокустук санды тандап алды. Биздин мисалда ал 7 санын тандады.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) ***a*** *өзгөрмөсүнө 23,* ***b****га 7.5,* ***c****га «Hello» маанисин ыйгаргыла.*
2) *Мурунку* ***a*** *өзгөрмөсүнүн маанисин 2.5 эсеге көбөйтүү менен өзгөрткүлө, жыйынтыгын* ***d*** *өзгөрмөсүнө жазып, жакынкы бүтүн санга чейин тегеректегиле.*
3) *Температура Цельсий градуста берилген. Берилген температураны Фаренгейт градуста туюндуруучу программаны түзгүлө. Туюндуруу үчүн формула: t(F) = 9/5 *t(C) + 32.*
4) *Туюнтманын маанисин эсептөөчү программаны жазгыла:*

$$\frac{(a+b)*c}{x^2+y^2}$$

3.3-тема:

Шарттуу операторлор

Буга чейинки караган мисалдарда операторлор биринин артынан бири удаалаш аткарылган ***сызыктуу*** программаларды жазууга мүмкүндүк берген. Алардын аткарылышында коддордун бир сабы да калтырылып кеткен эмес.

Бирок, көп учурда тигил же бул шартка жараша программанын аткаруу жүрүшү өзгөрүлүшү мүмкүн. Программанын кээ бир бөлүктөрү аткарылбай калып, ал эми башкалары аткарылышы мүмкүн. Б.а. программада биз 5-класста өздөштүрүп баштаган тармактануу пайда болот.

Python тилинде тармактануу `if` (эгер) жана `else` (антпесе) шарттуу операторлору менен жазылат. Алардын ишин мисалда карайлы.

1-маселе. Айдоочулук күбөлүктү алуу үчүн тест тапшырууга жашы 18ден ашкандарга гана уруксат бере турган программаны түзөлү:

```
a = int(input('Өзүңүздүн жашыңызды киргизиңиз: '))
if a >= 18:
    print('Тестке кирүүгө уруксат')
else:
    print('Тестке кирүүгө уруксат эмес')
```

ЭСИҢЕ ТУТ

Python тилинде шарттуу операторлорлуу нускамалардын баш сөзүнүн аягына сөзсүз кош чекит коюлат.

Көрүп тургандай `if` операторунан кийин шарттын өзү жазылган: **a >= 18,** ал **шарттуу оператордун баш сөзү** деп аталат. Баш сөз ар дайым «:» кош чекит менен аяктайт. Эгерде берилген шарт туура болсо (`True`), анда оператордон кийинки турган код ишке ашат: `print('Тестке кирүүгө уруксат')`. Эгерде шарт аткарылбаса (`False`), анда кийинки сап да аткарылбайт. Программа ошол замат `else` операторунан кийинки турган командаларга, биздин мисалда `print('Тестке кирүүгө уруксат эмес')` командасына өтөт.

Кош чекиттен кийин жаңы сапта оңго жылдыруу менен жазылган командалар **шарттуу операторду тулкусун** түзөт. Жылдыруулар шарттын тулкусун анын баш сөзүнөн оңой айырмалоого жардам берет. `if` жана `else` сөздөрү бир катарда, ал эми шарттын тулкусунун командалары алардын катарынан оңду көздөй бирдей аралыкка жылдырылган.

Туура жылдыруу бош орун баскычын 4 жолу басуу же Tab баскычын бир жолу басуу менен аткарылат. Көпчүлүк программалоо чөйрөлөрдө сапты кош чекит менен аяктап, Enter баскычын басканда кийинки сап автоматтык түрдө оңго жылдыруу менен жазылат. Python тилинде жылдыруулар өтө маанилүү, алар **программанын иштешине таасир этет.**

Эгерде бир нече альтернативдүү шарттарды киргизүү керек болсо, анда кошумча `elif` (`else` – `if`тин кыскартылганы) блогун колдонуп дагы башка командаларды көрсөтсө болот. Мисалы:

```
a = int(input('Өзүңүздүн жашыңызды киргизиңиз: '))
if a >= 18:
      print('Тестке кирүүгө уруксат')
elif a >= 16:
      print('Сынамык тестке кирүү')
else:
      print('Тестке кирүүгө уруксат эмес')
```

Салыштыруу операторлору

Салыштыруу операторлору эки маанини бири-бири менен салыштырып, жыйынтыгында True же False деген маанини чыгарат.

Математикалык символ	Python оператору	Мааниси	Мисал	Жыйынтык
<	<	Кичине	1 < 2	True
>	>	Чоң	1 > 2	False
≤	<=	Кичине же барабар	1 <= 2	True
≥	>=	Чоң же барабар	1 >= 2	False
=	==	Барабар	1 == 2	False
≠	!=	Барабар эмес	1 != 2	True

2-маселе. 20дан 70 жашка чейинки адамдардын коомдук ой-пикирин сурамжылоо жүрүп жатат. Сурамжыланып жаткан адамдын курагын (n) кир-

гизгенде, ал сурамжылоо үчүн «ылайык келет» же «ылайык келбейт» деген жоопту берген программаны түзөлү:

```
if n >= 20 and n <= 70:
    print ('ылайык келет')
else:
    print ('ылайык келбейт')
```

Python тилинде кош барабарсыздыктар көп колдонулат. Мисалы:

```
if 20 <= n <= 70:
```

деген төмөндөгү менен бир мааниде:

```
if n >= 20 and n <= 70:
```

3-маселе. Шарттуу жана салыштыруу операторлорун колдонуп, файлды сактоо диалогун көрсөтүүчү программасын жазгыла.

```
ans = input('Сиз файлды сактагыңыз келеби? (ооба/жок)')
if ans == 'ооба':
    print('Сактоо үчүн папканы тандаңыз')
elif ans == 'жок':
    print('Маалыматтар өчүрүлөт, андан ары колдоно албайсыз')
else:
    print('Ката. Жооптун мындай варианты жок')
```

Чыгаруунун жыйынтыгы тандалган жооптон көз каранды болот: «ооба» же «жок»:

1-вариант:

Сиз файлды сактагыңыз келеби?
(ооба/жок) ооба
Сактоо үчүн папканы тандаңыз

2-вариант:

Сиз файлды сактагыңыз келеби?
(ооба/жок) жок
Маалыматтар өчүрүлөт, андан ары колдоно албайсыз

ЭСИҢЕ ТУТ

= өзгөрмө үчүн маанини ыйгарат (эгер a = b болсо, анда a b болуп калат);

== эки маанини салыштырат (эгер a==b болсо, анда бул салыштырууга сурам жана программа True же False деген жыйынтыкты чыгарат).

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Үч сан берилди. Алардын ичинен канча сан бирдей экенин аныктоочу программа түзгүлө.*
2) *Натуралдык сан берилди. Бул сан: а) жуп экенин, б) 3кө эселүү экенин аныктоочу программаны түзгүлө.*
3) *Киргизилген айдын номерине жараша жыл мезгилинин атын чыгарып бере турган программаны жазгыла.*
4) *Сомду доллар жана евро валюталарына алмаштыруучу программаны жазгыла.*

3.4-тема:

while жана for циклдери

Циклдер шарттуу операторлор сыяктуу эле программалоонун маанилүү бөлүгү болуп саналат. Алардын жардамы менен коддун кээ бир бөлүктөрүнүн кайталанышын уюштурса болот. Python тилинде циклдерди жазуу үчүн **`while`** жана **`for`** командалары колдонулат.

`while` цикли

«While» англис тилинен «азырынча» деп которулат, б.а цикл (командалар блогу) берилген шартты аткарууга мүмкүн болгончо кайталана берет. Ал үчүн ар бир циклдин кадамынын башында шартты текшерүү аткарылат. Ошондуктан ал алдын ала шарттуу цикл деп аталат. Ал блок-схема катары кандай көрүнөрүн эстейли:

Алдын ала шарттуу цикл

Блок схемада ромб менен **циклдин баш сөзү**, тик бурчтук менен командалардын сериясынан турган – **циклдин тулкусу** көрсөтүлгөн. Эгерде циклдин башталышында баш сөздөгү шарт туура (**`True`**) болсо, анда циклдин тулкусундагы командалар бир жолу аткарылат жана программа кайрадан негизги бутакка кайтып келет. Мында шарт кайрадан текшерилет, эгерде ал туура болсо, анда командалардын блогу экинчи жолу аткарылат. Андан кийин кайрадан баш сөзгө кайтат ж.б.у.с. Ошентип процесс циклдин шарты толук аткарылмайынча кайталана берет.

Цикл өзүнүн ишин качан баш сөздөгү логикалык туюнтма «жалган» маанисин кайтарганда гана, б.а. циклди аткаруу шарты сакталбай калганда токтотот. Мындан кийин программаны аткаруу циклдин сыртындагы командаларга өтөт. Аны биз «циклден чыкты» деп айтабыз.

1-маселе. 1ден 5ке чейинки бардык бүтүн сандарды экранга чыгаралы.

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`d = 1`
`while d <= 5:`
`  print (d)`
`  d += 1`
	1
	2
	3
	4
	5

`if` шарттуу операторунда колдонулгандай эле циклдин шарты да `while` сөзүнөн кийин жазылат. Биздин мисалда бул: d <= 5. Шарттын аягында сөз-сүз түрдө кош чекит «:» коюлуп, ал эми циклдин тулкусундагы командалар оңго жылдырылып жазылышы шарт.

Каралып жаткан программада **d эсептегич-өзгөрмөсү** колдонулган. Анын баштапкы мааниси 1ге барабар. Циклдин ар бир айлануусунда анын мааниси 1ге көбөйүп турат. Ошондуктан биз **d+=1** (же **d=d+1**) сапчасын жаздык. Өзгөрмөнүн мааниси 5ке жеткенде гана цикл токтойт.

2-маселе. Колдонуучудан эки орундуу кодду киргизүүнү сурап, андан соң аны туура код менен салыштыруучу программаны жазалы. Эгерде код туура эмес киргизилсе, анда программа жаңысын киргизүүнү сурайт. `while` циклин колдонуу менен программаны төмөнкүдөй түзөбүз:

```
code = 35
a = int(input('Эки орундуу кодду киргизиңиз: '))
while a != code:
    print ('Код туура эмес. Кайрадан киргизиңиз.')
    a = int(input('Эки орундуу кодду киргизиңиз: '))
    if a == code:
        print ('Код туура киргизилди')
```

Программанын катылган коду 35ке барабар. Шартта (`while a != code:`) көрүнүп тургандай, колдонуучу 35ке барабар болгон санды киргизмейинче программа кодду кайрадан киргизүүсүн сурай берет. Ал эми if камтылган шартына ылайык эгерде киргизилген сан катылган кодго барабар болсо (`if a == code:`), анда программа аяктайт.

`while` цикли үчүн эң негизгиси – анын баш сөзүндөгү логикалык туюнтма жалган (**False**) маанисин кайтарган учурдун болушу. Антпесе ал чексиз циклге, б.а. циклден чыкпай калган программага айланат. Чексиз циклди токтотуу үчүн консоль терезесинде Ctrl+C баскычтарын басуу керек.

`for` цикли

`for` цикли командаларды керектүү ирет кайталап, программаны кыскартууга мүмкүндүк берет. Жогорку мисалда `for` циклин колдонолу:

```
for i in range (5):
    print (i)
```

Бул жерде **i** өзгөрмөсү (аны циклдин өзгөрмөсү деп аташат) 0дөн 5ке чейинки (5 өзү кирбейт) диапазондо (`in range`) өзгөрөт.

Ошентип цикл туура 5 жолу кайталанат.

Python тилиндеги программа		Экранга чыккан жыйынтыгы

`d = 1`
`while d <= 5:`
`  print ( d )`
`  d += 1`
	`for i in range(1,6):`
	`  print ( i )`
		1
		2
		3
		4
		5

`for` циклинин иштөө «диапазонун» аныктоочу `range()` функциясы 1, 2 же 3 аргументти алышы мүмкүн. Эгерде бир аргумент берилсе, анда диапазон 0дөн көрсөтүлгөн санга чейин (сан өзү кирбейт) болот. Эки аргумент берилсе, анда диапазон биринчиден экинчи санга чейинки (ал сан өзү кирбейт) бардык сандарды камтыйт. Эгерде үчүнчү сан берилсе – ал циклдин өзгөрмөсүнүн өзгөрүү кадамы. Кадам берилбесе ал 1ге барабар болот.

Python тилиндеги программа	Экранга чыккан жыйынтыгы
`for i in range(1,10,2):` `  print ( i )`	1 3 5 7 9

Жыйынтыктагы сандарды кемүү тартибинде да чыгарса болот. Ал үчүн баштапкы мааниси акыркысынан чоң, ал эми кадам – терс болушу керек.

Python тилиндеги программа	Экранга чыккан жыйынтыгы
`for i in range(15,0,-3):` `  print ( i )`	15 12 9 6 3

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) ***a*** *жана* ***b*** *бүтүн сандарын алган жана* ***a****дан* ***b****га чейинки диапазондогу бардык натуралдык сандардын квадраттарын чыгара турган программаны түзгүлө.*

2) *Натуралдык сан берилген. Ал сандын цифраларынын суммасын чыгарып берген программаны жазгыла.*

3) *Узундугу 23 метр болгон тактай берилген. Бул тактайдан узундуктары 1,5 м жана 2 м болгон бүтүн сандагы канча минималдык кесиндини даярдоого боло турганын эсептөөчү программаны түзгүлө.*

4 - бөлүм

Компьютердик тармактар жана интернет

4.1-тема:

Татаал издөө сурамдары

Издөө системаларынан так маалыматты табуу үчүн атайын символдор же сөздөр (операторлор) колдонулат. Мисалы, Беляевдин авторлугундагы жаныбарлар жана канаттуулар жөнүндөгү китептерди табалы.

1 **жаныбарлар & канаттуулар & Беляев** же **+жаныбарлар +канаттуулар +Беляев**

Изделүүчү документте бардык берилген сөздөр камтылат. Ал сөздөр документте катар же ар кайсы бөлүгүндө жайгашуусу мүмкүн.

2 **«Кыргызстандагы жаныбарлар жана канаттуулар»**

Силер издөөнү кандай берсеңер так ошондой сөз тартибин камтыган веб баракчалар көрсөтүлөт. Мындай сурам так цитатаны, ырдын же фильмдин аталышын табуу керек болгондо колдонулат.

3 **Жаныбарлар жана канаттуулар –Африка**

Сурамдын алдына бош орунсуз «минус» (-) белгиси коюлса, анда издөөнүн жыйынтыгында ошол сөз кездешпеген веб баракчалар берилет.

4 **Жаныбарлар|канаттуулар** же **жаныбарлар OR канаттуулар**

Эки сурамдын ортосундагы OR оператору ушул сөздөрдүн жок дегенде бирөөсү кезиккен баракчаларды табууга мүмкүндүк берет.

5 **Жаныбарлар жана канаттуулар site:www.kyrgyzstantravel.net**

Берилген домендин же сайттын чегинде гана издөөнү ишке ашырат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Издөө критерийлерине качан «+», качан «-» белгисин киргизүү керек?*
2) *Үй мышыктарын багуу боюнча маалыматты издөөгө татаал сурамды түзгүлө. Издөөдөн чоң мышыктарды (мисалы арстан, жолборс) жана сатуу, сатып алуу жөнүндөгү сунуштарды алып салгыла.*
3) *«ЖЕ» логикалык амалды белгилөө үчүн сурамда «|» белгиси, ал эми логикалык «ЖАНА» амалы үчүн «&» белгилери колдонулат. Алынган сурамдарды издөө сервери таап чыккан баракчалардын санын өсүү тартибинде жайгаштыргыла.*

Код	Сурам
А	Конан Дойль & Г. Бичер-Стоу & Джером К. Джером
Б	Конан Дойль | (Г. Бичер-Стоу & Джером К. Джером)

4.2-тема:

Сайттардын конструкторлору

Дүйнөдөгү биринчи сайт пайда болгондон бери, аларды түзүү технологиялары олуттуу өзгөрдү. Эгерде мурда ар бир сайт «нөлдөн» баштап жазылса, азыркы учурда интернеттин өнүгүшү жана веб-сайттардын өтө тез көбөйүшү менен аларды түзүүгө атайын конструкторлор иштелип чыккан. Алардын жардамы менен HTML тилин билбеген колдонуучулар да өзүнүн сайтын түзө алат.

БУЛ КЫЗЫКТУУ!

Дүйнөдөгү биринчи веб-сайт **info.cern.ch** 1991-жылдын 6-августунда британиялык окумуштуу-ойлоп табуучу Тим Бернерс-Ли тарабынан түзүлгөн. Бул өтө жөнөкөй, эч кандай графиканы камтыбаган жана World Wide Web технологиясы эмне үчүн керектигин түшүндүргөн тексттик веб-сайт болгон.

Бул конструкторлор сайттын мазмунун башкаруу системалары же «кыймылдаткычтар» (content management system – CMS) деп аталышат. Алардын жардамы менен сайттын баракчаларын түзүүгө жана калыпка келтирүүгө, сайттын мазмунун редакциялоого: документтер, медиафайлдар ж.б. менен толтурууга болот.

Ар түрдүү CMSтерде көп учурларда даяр чечимдер бар: алар дүкөн-сайттардын, визитка-сайттардын, лэндингдердин (кайсы бир аракетти жасоого чакырган сайттар) шаблондору. Ал эми силер типтүү эмес сайтты түзүүнү кааласаңар, анда аны нөлдөн баштап жазуу керек.

Статистика боюнча азыркы сайттардын 99%ы ар түрдүү CMSтерди колдонуу менен түзүлгөн. Алардын кээ бирлери дүйнөдө абдан популярдуу: WordPress, Drupal, Joomla ж.б.

АНЫКТАМАЛАР

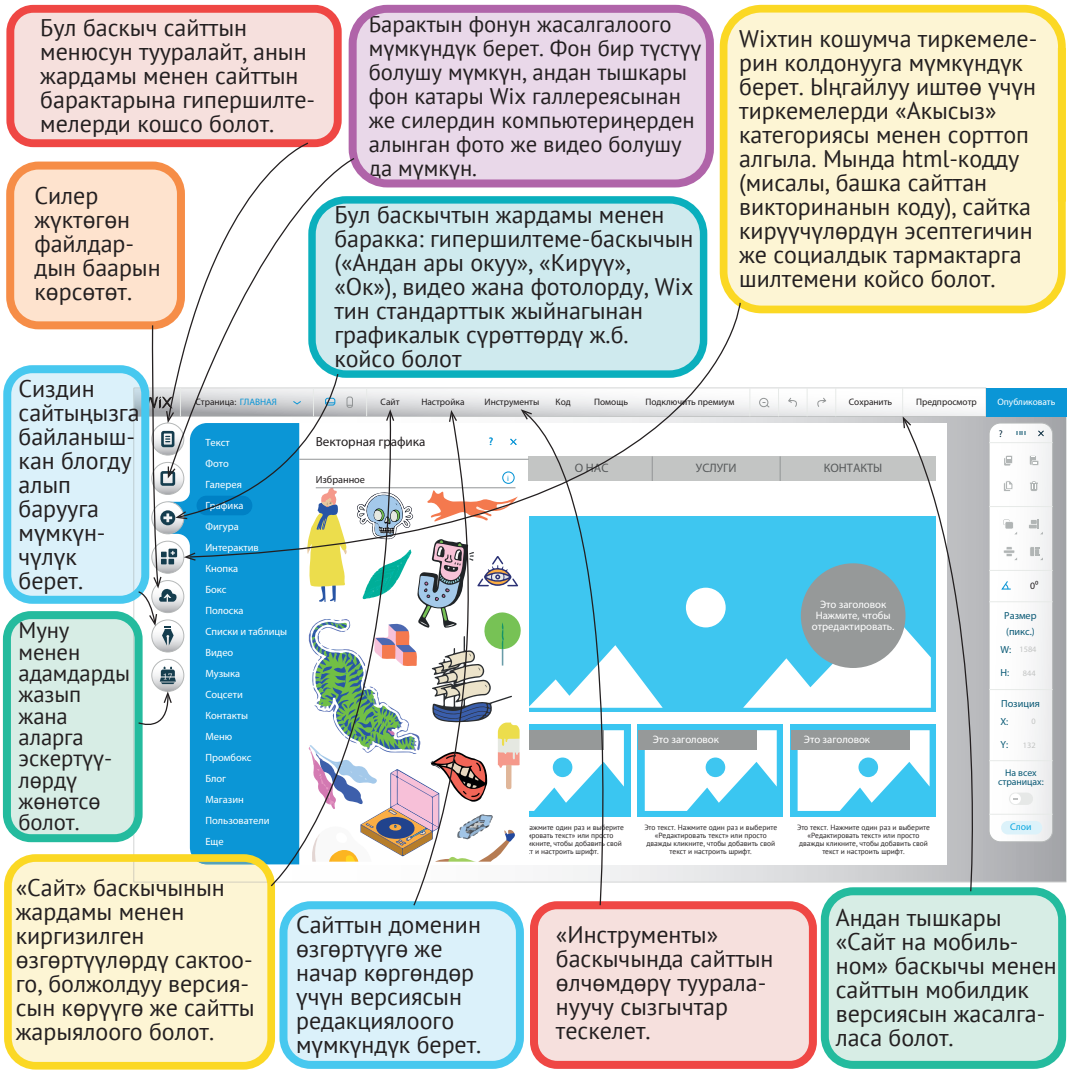
CMS – мазмунду (контентин) башкаруу системасы (англ. content management system, CMS) – бул сайттын мазмунун биргелешип түзүү, редакциялоо жана башкаруу процессин камсыздоо жана уюштуруу үчүн колдонулган программалык камсыздоо.

Лэндинг (англ. landing page) – негизги максаты сайтты колдонуучуга максаттуу аракетти жасатууга арналган веб-барак. Мисалы, бир нерсе үчүн добуш берүү, кайсы бир товар менен таанышуу жана аны сатып алуу, китепке буйрутма берүү ж.б.

Wix – сайттарды түзүү үчүн платформа

Анча чоң эмес визитка-сайтын, лэндинг же блогду тез арада түзүү үчүн эң популярдуу платформалардын бири болуп Wix сервиси эсептелет. Wixте сайттардын даяр көптөгөн кооз шаблондору бар. Ал жерден конструктордогудай эле силер баракчаларды кошуп же ашыкчасын алып салсаңар, стилин, тексттерди, фондук сүрөттөрдү, баскычтарды ж.б. өзгөртсөңөр болот.

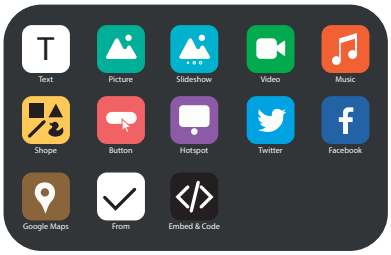
Веб-баракты түзүүдөн мурда силер сайттын контентин жана структурасын аныктап, ошого жараша сайттын шаблонун тандашыңар керек.



Интернетте Wix платформасын тереңдетип окутуучу көп материалдар бар. Эсиңерде болсун, бул платформанын жогорку деңгээлдеги кээ бир тейлөөлөрү акылуу болушу мүмкүн.

Сайттын мазмунун ар түрдүү визуалдык контенттерди, мисалы, лонгриддерди жана кооз презентацияларды кошуу менен да кызыктуураак кылсаңар болот. Аларды түзүү үчүн интернетте көптөгөн түрдүү платформалар бар.

Readymag деген конструктор лонгриддерди жасоо үчүн эң ыңгайлуу онлайн-платформалардын бири болуп саналат. Анын жардамы менен макалага текстти, сүрөттү, слайд-шоуну, видеофайлдарды жана музыканы оңой эле кошсо болот.

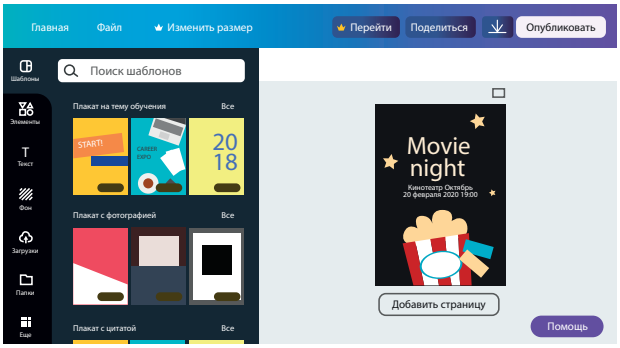


readymag.com – лонгриддерди түзүү үчүн платформа

АНЫКТАМАЛАР

Лонгрид (*англ.* longread) – ар кандай мультимедиалык элементтердин (фотография, сүрөттөр, видео, диаграмма ж.б.) жардамында блокторго бөлүнгөн узун текст (макала).

Контент (*англ.* мазмун) – бул колдонуучуга арналган маалымат жана тажрыйба. Сайттын контенти – бул анын ичинде камтылгандын бардыгы: текст, сүрөт, видео ж.б.



canva.com – графика жана дизайн үчүн сервис.

www.canva.com онлайн-сервиси силерге тез убакытта постерлерди, сайт үчүн баннерлерди, презентацияларды, соцтармактар үчүн сүрөттөрдү түзүүгө жардам берет. Текст, сүрөт жана фото менен болгон бардык аракеттер түздөн-түз сайтта жүргүзүлөт.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Графика жана дизайн үчүн платформаны колдонуп, мектебиңердеги бүтүрүү балынын афишасын түзгүлө.

4.3-тема:

Электрондук почта жана булуттук сервистер

Электрондук почта - интернетте эң көп колдонулган сервистердин бири. Электрондук почта менен иштөө үчүн почта тиркемелерин, мисалы, Outlook Express же жөн эле браузерди колдонсо болот.

Бүгүнкү күндө эң популярдуу почта сервиси болуп Gmail эсептелет. Gmailге катталуу (Google аккаунтту түзүү) Google аспаптарын бекер жана мыйзамдуу түрдө колдонууга мүмкүндүк берет. Алар: электрондук почта, Google диск, Google Classroom, Google Meet, Play Market, YouTube сайттары ж.б. Аккаунт түзүү үчүн www.gmail.com сайтына киргиле жана катталгыла.

Катталууда анкетада өзүңөрдүн чыныгы аты-жөнүңөрдү жана туулган датаңарды көрсөтүү керек. Анткени бурмаланган маалыматтар Googledун кээ бир сервистерине кирүүгө мүмкүнчүлүктү чектеп коюшу мүмкүн.

Электрондук почтанын дареги белгилүү формада жазылат жана «@» символу менен бөлүнгөн эки бөлүктөн турат:

Дарек_ээсинин_аты@сервердин_домендик_аты

Дарек ээсинин аты – бул колдонуучунун аты (логин).

Тамгалардын ар кандай регистрин, символдорду жана цифралардын айкалышын камтыган ишенимдүү паролду пайдалануу керек.

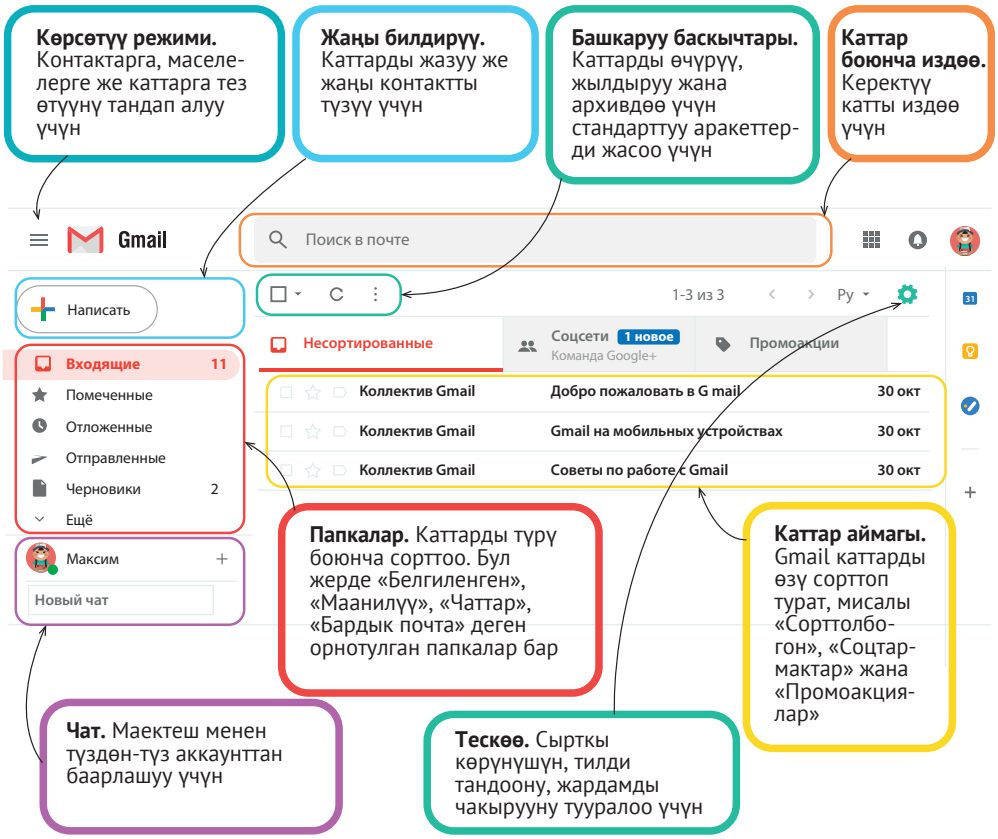
Google тиркемелери

Google Disk – маалыматты интернетте виртуалдык дискте сактоо үчүн сервис. ***Google диск***тин курамына онлайн тексттик документтерди, электрондук таблицаларды, презентацияларды ж.б. түзүүгө жана редакциялоого мүмкүндүк берүүчү тиркемелер да кирет. Документтин ээси башка колдонуучулар менен биргелешип иштөө максатында аны башкаларга жеткиликтүү кылса да болот ж.б.

Google Calendar – жолугушууларды пландоо үчүн ыңгайлуу сервис. Бул жерде жолугушуунун убактысын белгилеп, электрондук почта аркылуу башка катышуучуларга чакыруу жөнөтсө болот.

Google Translate – бир тилден башка тилге сөздөрдү, сүйлөмдөрдү жана веб-баракчаларды которууга мүмкүндүк берет. Бардыгы болуп тилдердин багыттары 1000ге жакын, мисалы, кыргыз тилинен орус, япон, серб, хинди тилдерине же тескерисинче.

Gmail почталык сервисинин терезеси:



Google Photo – санариптик фотосүрөттөрдөн коллаж жана слайд-шоуларды түзүү жана иштетүү үчүн атайын сервис.

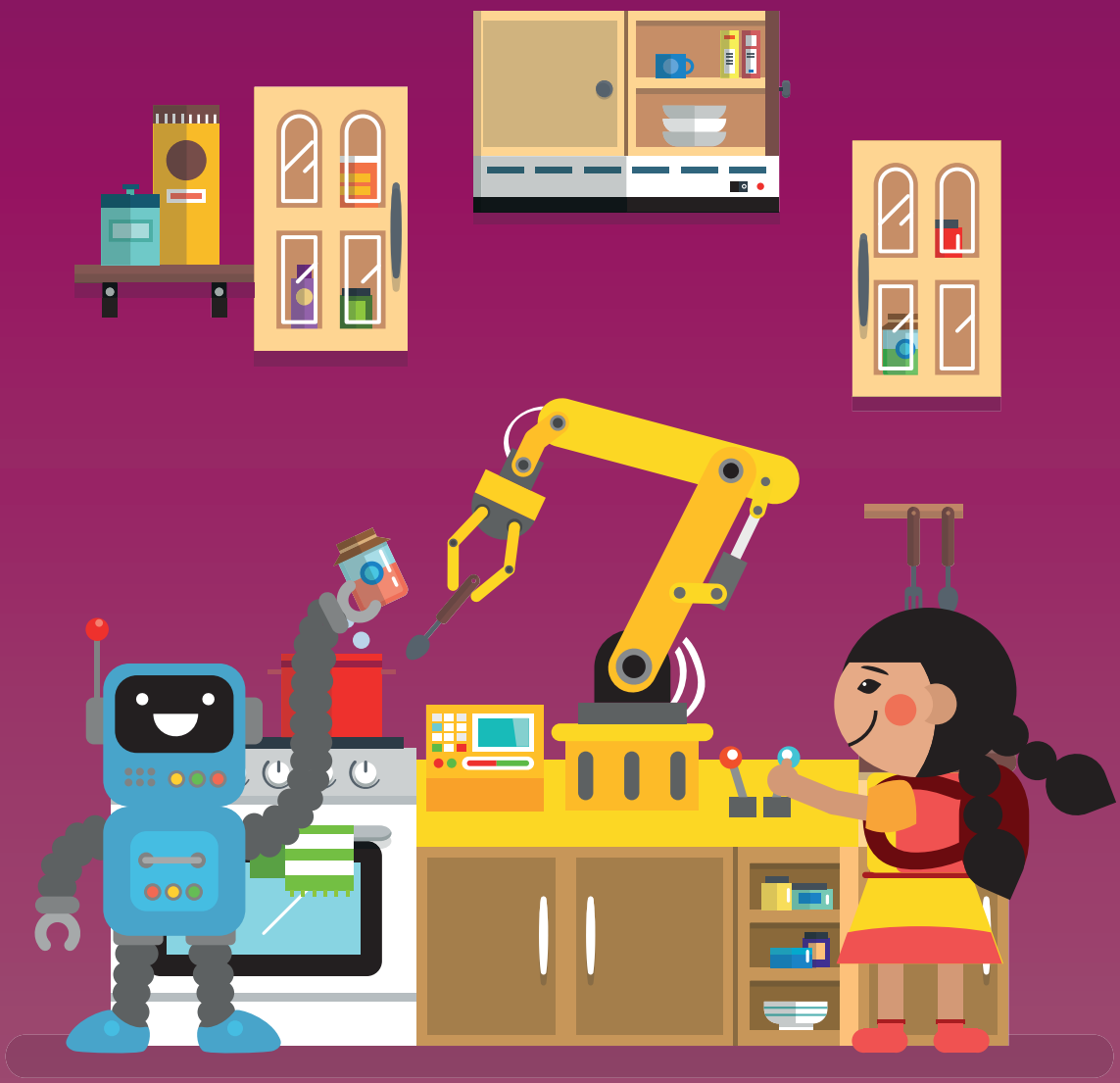
Google Maps – дүйнө жүзүнүн спутниктен тартылган сүрөттөрү жана карталары (Ай менен Марсты кошуп), маршруттарды издөө менен автомобиль жолдорунун картасы.

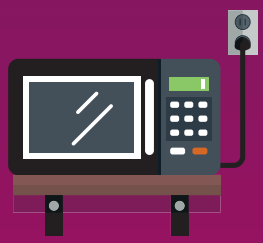
СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Google календарда мектеп каникулдары жана майрамдары боюнча эскертүүнү даярдагыла.*

2) *Google Maps аркылуу Кыргызстандын эң чоң көлдөрүн таап, алардын рельефтик сүрөттөрүн алгыла.*

3) *Google Maps аркылуу Эйфель мунарасынан Париждеги Майыптар үйүнө чейинки маршрутту тапкыла жана белгиленген жерге жөө жүрсө, автомобиль менен же велосипед менен барса, канча убакыт кетерин аныктагыла.*





6 - класс

1 - бөлүм

Информатика жана маалымат

1.1-тема:

Логикалык туюнтмалар жана амалдар

Күнүмдүк жашообузда «логикалуу», «логика» деген сөздөрдү биз көп колдонобуз. Байыркы грек тилинен которгондо «логика» – «ой жүгүртүү жөнүндөгү илим» деп которулат. Бул, туура ойлонуу, ырастоолорду далилдөө жана чечим чыгаруу жөнүндөгү илим.

Логика чындыкка жетүү ыкмаларын сезимдерге эмес, мурда алынган билимдерге таянуунун натыйжасында изилдейт. Логика математиканын бир бөлүмү болуп саналат жана көптөгөн илимдерде колдонулат, ал эми буль алгебрасы информатиканын негиздеринин бири болуп эсептелет.

БУЛ КЫЗЫКТУУ

1847-жылы англиялык математик Джордж Буль логикалык айтымдарды изилдөө үчүн татаал сөз түрүндөгү айтымдарды кыскача символдор менен белгилеген математикалык ыкманы колдонууну сунуштаган. Кийин математиканын бул бөлүмү логика алгебрасы же «буль алгебрасы» деген атка ээ болгон. Анын жардамында логикалык айтымдарды жазышат, эсептешет, жөнөкөйлөтүшөт жана өзгөртүп түзүшөт.
Буль алгебрасы өтө жөнөкөй, анткени ар бир өзгөрмө «чындык» же «жалган» деген эки маанини эле алышы мүмкүн.

Логика алгебрасы айтымдардын мазмунун териштирбестен, алардын чындык же жалган экенине гана карайт. Ал алгебралык эсептөөлөр жолу менен курама (татаал) айтымдардын чындык же жалган экенин аныктоого мүмкүндүк берет.

Логикалык айтымдарды латын алфавитинин баш тамгалары менен белгилешет:

***A* = «Жер тоголок»,**
***B* = «Күн – бул планета».**

АНЫКТАМАЛАР

Жөнөкөй логикалык айтым – бул маанисин жоготпостон, кичирейтүүгө же бөлүүгө мүмкүн болбогон айтым.

Татаал айтым логикалык амалдарга дал келген логикалык байламталар менен бириккен жөнөкөй айтымдардан турат.

Логикалык туюнтма – бул логикалык амалдар менен бириккен логикалык айтымдар.

Негизги логикалык амалдар

 Дизъюнкция (кошуу) – бул эгерде жөнөкөй логикалык айтымдардын бирөөсү гана чын болсо, анда чындык, жана экөөсү тең жалган болгондо гана жалган маанисин алган татаал логикалык туюнтма. Логикалык кошуу эки (же андан көп) айтымдын «же» байламтасы менен бирөөгө биригишинен түзүлөт.

Мисалы: *Апамдын маанайын көтөрүү үчүн идиш-аякты жуун же ага чай сунуштоо керек.*

 Конъюнкция (көбөйтүү) – бул жөнөкөй логикалык айтымдардын бардыгы тең чындык болгондо гана чындык маанисин алган, ал эми калган бардык учурларда жалган маанисин алган татаал логикалык туюнтма. Логикалык көбөйтүү эки (же андан көп) айтымдын «жана» байламтасы менен бирөөгө биригишинен түзүлөт.

Мисалы: *Максаттарга жетүү үчүн билим жана тырышчаактык керек.*

 Импликация (ээрчүү) – бул биринчи айтымы шарт, ал эми экинчиси натыйжасы болгон татаал логикалык туюнтма. Мындай логикалык туюнтма, чындыктан жалган келип чыгат дегенден башка бардык учурда чындык маанисин алат. Логикалык ээрчүү «эгерде…анда…» деген сөз айкашынын жардамы менен эки айтымдын бирөөгө биригишинен түзүлөт.

Мисалы: *Эгерде сан 6га бөлүнсө, анда ал 3кө да бөлүнөт.*

 Инверсия (тануу) – бул эгерде баштапкы логикалык айтым чындык болсо, тануунун жыйынтыгы жалган жана тескерисинче баштапкы логикалык айтым жалган болсо, тануунун жыйынтыгы чындык болгон татаал логикалык туюнтма.

Логикалык тануу баштапкы логикалык айтымга «эмес», же «мындай туура эмес» сөз байламталарын кошуу жолу менен түзүлөт.

Мисалы: *«Кыргызстандын борбору - Бишкек».*

Инверсиядан кийин: «Кыргызстандын борбору - Бишкек ЭМЕС».

 Эквиваленттүүлүк (барабардык) – бул качан жөнөкөй логикалык айтымдардын экөөсү тең бирдей чындык маанисин алганда гана чындык болгон татаал логикалык туюнтма. Логикалык барабардык «качан» жана «качан гана» сөз айкаштарынын жардамы менен эки айтымдын бирөөгө биригүүсүнөн түзүлөт.

Мисалы: *Жеңиш чокусу – Кыргызстандагы эң бийик тоо чокусу.*

Логикалык амалдар үчүн чындык таблицасы

Белгилейбиз: 0 – бул жалган айтым, 1 – чындык.

Амалдардын аталышы	Кошуу	Көбөйтүү	Тануу	Ээрчүү	Барабардык
	Дизъюнкция	Конъюнкция	Инверсия	Импликация	Эквиваленттүүлүк
Белгилениши	$\vee$	$\wedge$	$\neg$	$\rightarrow$	$\equiv$
Чындык таблицасы	$0 \vee 0=0$ $0 \vee 1=1$ $1 \vee 0=1$ $1 \vee 1=1$	$0 \wedge 0=0$ $0 \wedge 1=0$ $1 \wedge 0=0$ $1 \wedge 1=1$	$\neg 0=1$ $\neg 1=0$	$0 \rightarrow 0=1$ $0 \rightarrow 1=1$ $1 \rightarrow 0=0$ $1 \rightarrow 1=1$	$0 \equiv 0=1$ $0 \equiv 1=0$ $1 \equiv 0=0$ $1 \equiv 1=1$
	же	жана	эмес	эгерде ... анда	качан гана

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) Формула менен берилген татаал туюнтмаларды кадимки тилде жазып бергиле. А=«Тимурга тарых жагат», В=«Зояга информатика жагат», С=«Рита тарых боюнча видеоролик жасай алат».

а) А ∧В; б) В>С; в) А ∧С; г) (¬А)>(¬ С); д) А∨В∨С; е) (А ∧В)>С

2) Берилген айтым чындык болгон Х бүтүн санын атагыла: Х>6 ЭМЕС ЖАНА Х>4

3) Таблицада издөө серверине болгон сурамдар берилген. Сурамдагы «ЖЕ» логикалык амалын белгилөө үчүн «|» символу, ал эми «ЖАНА» логикалык амалын «&» менен белгилөө колдонулат. Ар бир сурам үчүн А дан Га чейинки тамгаларга туура келген анын коддору көрсөтүлгөн. Сурамдардын коддорун солдон оңго карай табылган беттердин сандарынын өсүү тартиби боюнча жайгаштыргыла. Мында табылган беттердин сандары ар башка.

Код	Сурам
А	Эльфтер | Гномдор | Хоббиттер| Орктор
Б	(Эльфтер & Гномдор) | Орктор
В	Эльфтер & Гномдор
Г	Эльфтер | Хоббиттер

1.2-тема:

Логиканын мыйзамдары

Татаал логикалык туюнтманын маанисин табууда логиканын мыйзамдарын билүү керек, алардын айрымдары арифметика мыйзамдарына окшош.

МЫЙЗАМ	БАЯНДООСУ (ФОРМУЛИРОВКАСЫ)
1. Теңдештик (окшоштук) мыйзамы $A = A$	Ар кандай айтым өзүнө барабар.
2. Орун алмаштыруу (коммутативдик) мыйзамы $A \wedge B = B \wedge A$ $A \vee B = B \vee A$	Айтымдардын үстүнөн жүргүзүлгөн амалдардын жыйынтыгы, бул айтымдардын кандай иретте алынгандыгынан көз каранды эмес. (Кошулуучулардын ордун алмаштыруудан сумма өзгөрбөйт. Көбөйтүүчүлөрдүн ордун алмаштыруудан көбөйтүндү өзгөрбөйт).
3. Топтоштуруу (ассоциативдик) мыйзамы $(A \vee B) \vee Z = A \vee (B \vee Z)$ $(A \wedge B) \wedge Z = A \wedge (B \wedge Z)$	Бирдей белгилерде кашааларды каалагандай жайгаштырса, же таптакыр алып салса болот.
4. Бөлүштүрүү (дистрибутивдик) мыйзамы $(A \wedge B) \vee C = (A \wedge C) \vee (B \wedge C)$ $(A \vee B) \wedge C = (A \vee C) \wedge (B \vee C)$	Жалпы айтымды кашаанын сыртына чыгаруу жана кашааларды ачуу эрежелерин аныктайт.

Кошумча мыйзамдар:

Тең күчтүүлүк мыйзамы
$A \vee A = A$
$A \wedge A = A$

Жутуу (сиңирүү) мыйзамы
$A \vee (A \wedge B) = A$
$A \wedge (A \vee B) = A$

Жалпы инверсия мыйзамы Де Моргандын мыйзамы
$\neg(X \vee Y) = \neg X \wedge \neg Y$
$\neg(X \wedge Y) = \neg X \vee \neg Y$

Карама-каршылык мыйзамы
$X \wedge \neg X = 0$

Константаларды чыгарып салуу мыйзамы
$A \vee 1 = 1, A \vee 0 = A$
$A \wedge 1 = A, A \wedge 0 = 0$

Жабыштыруу мыйзамы
$(A \wedge B) \vee (\neg A \wedge B) = B$
$(A \vee B) \wedge (\neg A \vee B) = B$

Эки жолку тануу мыйзамы
$\neg(\neg X) = X$

Үчүнчүсүн алып салуу мыйзамы
$X \vee \neg X = 1$

Логикалык маселелерди чыгаруу логикалык айтымдарды формалдаштыруу жолу менен жүргүзүлөт.

АНЫКТАМА

Формалдаштыруу – бул берилген айтымды символдордун жардамы менен формалдуу жазуу.

Маселе:

Ажара, **Диана**, **Ира** жана **Тамара** ар башка чет тилдерин окушат: англис, француз, кытай жана япон тилдери.

Андрей кыздардан ар бири кайсы тилдерди окушарын сураганда, **Тамара** мындай жооп берди: «Мен япон тилин».

Ал эми калган кыздар: «Бул сыр, эми өзүң таап көр эгерде:

Ажара кытай тилин окуса,
Диана кытай тилин окубайт,
Ира болсо француз тилин окубайт.

Ошондой эле, бул жерде айтылган үч ырастоонун бирөөсү чындык, ал эми экөө – жалган».

Андрейге туура жообун табуу үчүн ой талкуу ыкмасын колдонуп жардам берели.

3 ырастоо берилип жатат: Кыскартып жазалы:

1-ырастоо. **Ажара** кытай тилин окуйт, **А**=К

2-ырастоо. **Диана** кытай тилин окубайт, **Д**≠К

3-ырастоо. **Ира** француз тилин окубайт. **И**≠Ф

Шарт боюнча – булардын ичинен бирөөсү гана чындык.

1. Биринчи ырастоону туура деп эсептеп көрөлү:
*«**Ажара** кытай тилин окуйт»*

Анда, маселенин шарты боюнча башка эки ырастоо жалган болуп чыгыш керек. Мында эгер **Диана кытай тилин окубайт** деген жалган болсо, демек ал **кытай тилин окуйт экен да.**

А = *К=1*
Д ≠ *К=0*
И ≠ *Ф=0*

Демек **Ажара** да жана **Диана** да кытай тилин окуп калышат да. Ал эми баштапкы шартта кыздар ар түрдүү тилдерди окушат деп жазылган, натыйжада бул божомол туура эмес жана биринчи ырастоо чындык болушу мүмкүн эмес.

А = *К = 1*
Д ≠ К = 0
Демек Д=К
И ≠ *Ф= 0*

2. Эми экинчи ырастоо чындык деп эсептейли:

*«**Диана** кытай тилин окубайт».*

Анда маселенин шарты боюнча биринчи жана үчүнчү ырастоо да туура эмес. Мындан эч кимиси кытай тилин окубайт деген жыйынтык келип чыгат. Демек, мында да биздин божомол туура болбой калды.

А = *К=0*
Д ≠ *К=1*
И ≠ *Ф=0*

3. Маселенин шарты боюнча үч ырастоонун бирөөсү гана чындык болушу керек. Биринчи караган эки ырастоо жалган болгондуктан, үчүнчү: «**Ира** француз тилин окубайт» деген ырастоо чындык, ал эми биринчи экөө туура эмес экен да.

А = *К=0*
Д ≠ *К=0*
И ≠ *Ф=1*

Демек:

А ≠ *К=1* **Ажара** кытай тилин окубайт (француз же англис тили калат экен),
Д = *К=1* **Диана** кытай тилин окуйт.

Демек биз, **Диана** кытай тилин окуй тургандыгын аныктадык.
Анда **Ира** менен **Ажара** кайсы тилдерди окушат?
«**Ира** француз тилин окубайт» болсо, анда ал англис тилин окуйт.
Анда француз тилин **Ажара** окуйт экен да.

Жооп:

Тамара япон тилин окуйт, **Диана** кытай, **Ира** англис, ал эми **Ажара** француз тилин окушат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

Индира, Эмил жана Лена байыркы идишти таап алышты. Бул идиш жөнүндө алардын ар бири экиден божомолдорун айтышты:
Индира: Бул Скифтерден калган идиш жана ал V кылымда жасалган.
Эмил: Бул идиш Согдийлердики жана III кылымда пайда болгон.
Лена: Бул идиш скифтердики эмес жана IV кылымга таандык.
Тарых мугалими балдарга, ар бириңер эки божомолдун бирөөсүндө гана туура айтып жатасыңар деди.
Бул идиш качан жана кайда жасалган?

1.3-тема:

Логикалык туюнтмаларды чыгаруу

Маселелердин негизинде логикалык туюнтмаларды түзүүнү карайлы:

Элестеткиле, силерге суу астындагы кеменин сигнал берүү системасын программалоо маселеси берилди. Эгерде үч кыймылдаткычтын экөөсү иштен чыгып калса, авариялык сигналды бергендей кылуу керек.

A - « Биринчи кыймылдаткыч иштен чыкты».
B – «Экинчи кыймылдаткыч иштен чыкты».
C – «Үчүнчү кыймылдаткыч иштен чыкты».
X – «Авариялык кырдаал».

Сигнал берүү системасы «X» кырдаалында иштейт.

«X» абалын формула түрүндө көрсөтсөк болот:
X = (A жана B) же (A жана C) же (B жана C)

Биз формалдаштырууну аткардык.

Логикалык амалдардын символдорунун жардамы менен бул туюнтма мындай жазылат:
$X=(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$ Мындай сигнал компьютерге түшүнүктүү болот.

Логикалык туюнтмаларды айтымдар алгебрасынын жардамында чыгаруу

Логикалык амалдарды аткаруу ирети:

1. Кашааларда жайгашкан туюнтма;
2. Тануу;
3. Логикалык көбөйтүү;
4. Логикалык кошуу;
5. Логикалык ээрчүү;
6. Логикалык барабардык.

1-маселе. Татаал логикалык туюнтманы чыгаруу:

Жөнөкөй туюнтмалар берилген: **A={3=6}, B={2<3}, C={4<1}.**
Курама туюнтманын чындык экенин аныктагыла: **(¬A∨A∧B)∧¬C**

Чыгаруу алгоритми:

1) Жөнөкөй туюнтмалардын чындык экенин аныктайбыз:
A={3=6} жалган (0)
B={2<3} чындык (1)
C={4<1} жалган (0)

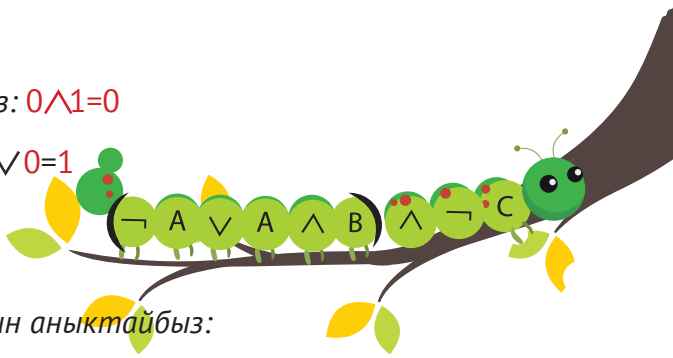
2) Курама туюнтмага маанилерин коёбуз:
(¬A∨A∧B)∧¬C=(¬0∨0∧1)∧¬0

3) Тануу аракетин эске алуу менен кашаанын ичинде жайгашкан туюнтманын маанилерин аныктайбыз:
(¬0∨0∧1)=1∨0∧1

4) Логикалык көбөйтүүнү аткарабыз: 0∧1=0

5) Логикалык кошууну аткарабыз. 1∨0=1

6) Ортодогу жыйынтыкты алабыз:
(¬0∨0∧1)=1



7) Баштапкы туюнтманын маанисин аныктайбыз:
1∧¬0=1∧1=1(чындык)

Жооп: (¬A∨A∧B)∧¬C курама туюнтмасы *чындык.*

2-маселе. Берилген сандардын кайсынысы үчүн төмөндөгү айтым чындык: **(сан<100) ЭМЕС ЖАНА (сан жуп) ЭМЕС? 115, 108, 35, 8.**

Чыгаруу алгоритми:

Туюнтманы (сан >=100) ЖАНА (сан жуп) түрүндө жазалы.
Текшеребиз:

1) Чындык, анткени эки айтым тең чындык: 115 саны 100 дөн кичине эмес жана так сан. Бул туура жооп.
2) Жалган, анткени экинчи айтым жалган: 108 –жуп сан.
3) Жалган, анткени биринчи айтым жалган: 35 саны 100 дөн чоң эмес.
4) Жалган, анткени экинчи айтым жалган: 8 – так сан

Чындык таблицасын түзүү

Логикалык туюнтманын чын экенин аныктоо үчүн чындык таблицасын колдонууга болот.

Чындык таблицасы татаал айтымга кирген жөнөкөй айтымдардын бардык маанилеринде кандай маанини ала тургандыгын көрсөтөт.

Чындык таблицасынын жардамы менен логикалык туюнтмаларды чыгаруу алгоритми:

1. логикалык туюнтмадагы n өзгөрмөлөрүнүн санын эсептеп чыгуу;
2. $m=2^n$ формуласы боюнча таблицадагы саптардын санын эсептөө, мында n өзгөрмөлөрдүн саны;
3. формуладагы логикалык амалдардын санын эсептөө;
4. приоритетти жана кашааларды эсепке алуу менен логикалык амалдардын ырааттуулугун аныктоо;
5. мамычалардын санын аныктоо: өзгөрмөлөрдүн саны + аракеттердин саны;
6. киргизилүүчү өзгөрмөлөрдүн топтомун жазып алуу;
7. 4-пунктта көрсөтүлгөн ырааттуулукта логикалык амалдарды аткаруу менен чындык таблицасын мамычалар боюнча толтуруу.

Мисал: ***чындык таблицасынын жардамы менен $\neg A \vee B$ туюнтмасынын чындык экенин аныктагыла.***

1. Логикалык туюнтмадагы өзгөрмөлөрдүн санын аныктайбыз: 2.
2. $m=2^n$ формуласы менен таблицадагы саптардын санын аныктайбыз: $2^2=4$.
3. Формуладагы логикалык амалдардын санын аныктайбыз: 2.
4. Логикалык амалдардын аткарылуусунун ырааттуулугун орнотобуз: биринчи – логикалык тануу, экинчиси логикалык көбөйтүү.
5. Таблицадагы мамычалардын санын аныктайбыз: өзгөрмөлөрдүн саны + амалдардын саны=2+2 = 4.

6 Киргизилген өзгөрмөлөрдүн топтомун жазып алабыз.

7 А жана В өзгөрмөлөрүнүн маанилеринин мүмкүн болгон бардык варианттарын толтуруу менен чындык таблицасын мамыча боюнча толтурууну жүргүзөбүз.

8 4-пункттa көрсөтүлгөн ырааттуулукка жараша логикалык амалдарды аткарабыз.

$\neg A \vee B$

A	B	$\neg A$	$\neg A \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

СУРООЛОР ЖАНА ТАПШЫРМАЛАР

1) *Логикалык туюнтманы логика алгебрасы тилинде жазгыла:*

- *Каникул учурунда биз театрга же циркке барабыз;*
- *15 саны 3кө же 5ке бөлүнөт;*
- *Эгерде сандын цифраларынын суммасы 3кө калдыксыз бөлүнсө, анда сан да 3кө бөлүнөт.*

2) *А айтылышы «х – так сан», В айтылышы «х 5ке бөлүнөт». $A \vee B$ логикалык кошуусу кандай жыйынтык берет?*

3) *$((1 \vee 0) \vee 1) \vee 0$ логикалык туюнтмасынын маанисин тапкыла.*

4) *Жөнөкөй айтымдар берилген:*

$A=\{3=6\}$, $B=\{2<3\}$, $C=\{4<1\}$

Курама айтымдын чындык экендигин аныктагыла:

$(A \vee \neg B) \wedge \neg (\neg A \vee C)$.

5) *a, b, c, - төмөнкүдөй маанилерге ээ болгон логикалык чоңдуктар болсун:*

a= чындык, b=жалган, c=жалган.

Эми төмөнкү логикалык туюнтмалар үчүн чындык таблицасын түзгүлө:

а) а же b жана с эмес; б) а жана b жана же с жана b.

6) *Силер окуп үйрөнгөн логиканын мыйзамдарын чындык таблицасынын жардамы менен далилдегиле.*

2 - бөлүм

Компьютер жана ПК

2.1-тема:

ПК жана лицензиянын түрлөрү

Силер көргөндөй программалык камсыздоосуз (ПК) эч бир компьютерде иштөөгө мүмкүн эмес. Ал компьютерге автоматтык түрдө орнотулушу мүмкүн (мисалы, операциялык системаны орнотконду, анын стандарттык ПКсынын курамына жөнөкөй тексттик редактор, калькулятор, графикалык редактор ж.б. кирет) же болбосо колдонуучунун каалоосу менен кошумча орнотулат.

Программалык камсыздоо

Проприетардык (эркин эмес) ПК

Эркин ПК

Белгилүү болгондой, каалагандай программа бул адамдын интеллектуалдык ишмердүүлүгүнүн жыйынтыгы, демек – анын менчиги.

Мисалы, сиз дүкөндөн уюлдук телефон сатып алдыңыз. Эми бул сиздин менчигиңиз, а бирок телефондун дизайны мурдагыдай эле телефонду иштеп чыгуучуларга жана дизайнерлерге таандык. Эгерде сиз бул телефондордун нускаларын (копияларын) чыгарып баштасаңыз, анда аны иштеп чыгуучулардын автордук укуктарын бузган болосуз.

ЭСИҢЕ ТУТ

Автордук укук бир эле программаларга эмес, фильмдерге, музыкага, ал эмес графикалык объекттерге (сүрөт, картина) да жайылат.

Ушундай эле кырдаал программаларга да тиешелүү. Программанын өзү – бул иштеп чыгуучунун интеллектуалдык менчиги, а сиз болсо бул программанын иштөөсүнүн жыйынтыгын гана сатып аласыз. Мыйзамдар боюнча сиз программаны колдонуу укугуна гана ээ боло аласыз. Берилген программаны колдонуунун баардык шарттары адатта программаны орнотуп жатканда чыгуучу лицензиялык макулдашууда көрсөтүлөт. Лицензиялык макулдашууга макулдугуңузду берүү менен сиз программаны иштеп чыгуучулардын бардык шарттарын кабыл аласыз.

Эгерде сиз лицензиялык макулдашуунун бир эле шартын бузсаңыз (андан ары таратууга тыюу салынгандыгын, өзгөртүп түзүүдө ж.б.), анда сиздин программа легалдуу болбой калат.

Эркин ПКнын өнүгүү тарыхы

Акы төлөнүүчү лицензиялык (проприетардык) ПКнын өнүгүшүнө каршы салмакта өткөн кылымдын 80-жылдарынан баштап дүйнө жүзүндөгү программалоочулардын арасында **эркин (Open-source software) ПК**ны өнүктүрүү кыймылы да пайда болгон.


Ричард Столлман

Эркин ПК - бул бир гана акысыз ПК эле эмес, андай программа ачык баштапкы коду менен берилип, каалоочулар үчүн аны андан ары өзүнүн муктаждыктары үчүн өзгөртүп түзүп алуу мүмкүнчүлүгү да каралган. Ачык баштапкы код ачык лицензиянын негизинде таралат.

Ушундай ачык лицензиялардын бири катары **General Public License (GPL)** эсептелет. Анын түзүүчүсү ачык ПК кыймылынын негиздөөчүсү Ричард Столлман болуп саналат.

Мындай лицензияны колдонуунун өзгөчөлүгү болуп, каалаган колдонуучу мындай ПКны өзгөртүүгө, толуктоого жана таратууга толук укугу бар экендиги эсептелет. Бардык жаңы өзгөртүүлөрдүн коду да башкалар үчүн ачык болушу керек.

Ушундай лицензия менен Linux, MySQL, Asterisk ж.б. өзөктөрү таралат. ПКны лицензиялоо үчүн экинчи популярдуу ачык лицензия катары **Apache License 2.0** эсептелет. Бул лицензия колдонуучу тарабынан өз алдынча жазылган коддун бөлүгүн жаап коюуга мүмкүндүк берет.

Эң популярдуу **Android** операциялык системасынын баштапкы коддорунун көбү ушул Apache 2.0 эркин лицензиясы менен таратылат. Учурда Android ОСде ар кандай түзүлүштөр, «интернет буюмдар» жана «акылдуу үйлөрдөн» баштап сыналгы, ноутбук жана автомобилдерге чейин иштейт. Бирок көп учурда Androidди смартфон жана планшеттерде кеңири колдонушат.

Интернеттеги маалыматты колдонуунун легалдуулугу (мыйзамдуулугу)

Интернеттеги материалдарды колдонууда жана файлдарды жөнөтүүдө тармактагы көпчүлүк маалыматтар автордук укук жөнүндө мыйзам (бул

жөнүндө веб-сайттарда түздөн-түз көрсөтүлбөсө да) менен корголгондугун эске алуу керек. Бул берилген маалыматты сиз болгону көрүүгө гана мүмкүнчүлүгүңүз бар дегенди түшүндүрөт. Аны сиз көчүрүп алууга, таратууга жана башкаларга берүүгө укугуңуз жок. Көпчүлүк учурларда мындай маалыматтар © ***[автордун аты-жөнү]*** копирайт белгиси менен белгиленет. Ал маалыматты колдонуу же иштеп чыгуу үчүн маалыматтын же сайттын ээсинен сөзсүз түрдө уруксат алуу керек дегенди түшүндүрөт.

Андыктан, интернеттен маалыматты көчүрүп алуудан мурда берилген материал анын автору тараптан эркин таратыларына толук ынануунуз зарыл. Андан тышкары, «акысыз» колдонуу жана «эркин» колдонуу бир маанини билдирбейт. Акысыз колдонууда материалды жөн гана акысыз көрүп жана окууга болот, бирок аны андан ары таратууга болбойт.

Ал эми эркин колдонуу деген түшүнүк кененирээк маанини берет. Ал материалды бир гана көрүүдөн жана окуудан тышкары, аны кайра иштетүүгө (мисалы, тексттин бөлүгүн доклад үчүн алууга же фильмдин кайсы бир бөлүктөрүн колдонууга) жана андан ары таратууга да мүмкүндүк берет.

Интернетте авторлору башкаларга толук колдонуу укуктарын берген жана эркин таратылган билим берүү материалдары **ачык билим берүү ресурстары** деп аталат. Эң ири ачык билим берүү ресурсуна Википедия кирет. Андагы бардык макалалар Creative Commons ачык лицензиясы менен берилет.

Бул лицензия менен түзүүчү-авторлор бардыгына анын эмгегин эркин колдонуп, таратууга жана кошумчалоого уруксат берет. Ал эмес силер кармап турган бул окуу китеби да ачык билим берүү ресурсу. Китептин биринчи барагынан силер ачык лицензиянын белгисин таба аласыңар:

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Эмне үчүн компьютердик каракчылык коомго зыян алып келет?*
2) *Open-source программаларга мисал келтиргиле жана алардын кызматын сүрөттөп бергиле.*
3) *Интернеттен алынган санариптик материалдарды толуктап иштетүүнүн кандай ыкмаларын билесиңер? Алардын кайсыларына авторлорунан уруксат алыш керек?*

2.2-тема:

Маалыматтар базасы

Азыркы коомдун эң маанилүү маалыматтык ресурстарынын бири болуп маалыматтар базасы (МБ) эсептелет. Азыркы мезгилде көптөгөн ири интернет-ресурстар МБсын маалыматтарды сактоо үчүн колдонушат, анткени алар бирдей касиеттер топтомуна ээ болгон объекттердин группасы жөнүндөгү берилиштерди уюштурулган тартипте сактоого мүмкүндүк берет.

АНЫКТАМА

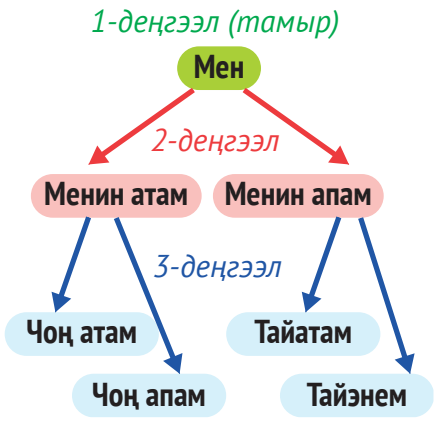
Маалыматтар базасы (мындан ары МБ) – бул эсептөө системасында эффективдүү издөө жана иштетүү максатында логикалык системалаштырылган маалыматтар топтому.

МБнын негизги параметрлери – бул анын физикалык жана логикалык көлөмдөрү. Физикалык көлөм байттар, Кб, Мб ж.б. менен туюнтулат, ал эми логикалык болсо жазуулардын санынан көз каранды. Эң ири маалыматтар базасын ондогон терабайт жана миллиарддаган жазуулар түзөт.

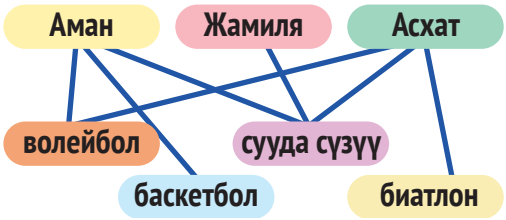
Берилиштерди иреттөөнү берилиштер арасындагы байланыштарды жана алардын үстүнөн мүмкүн болгон аракеттерди түшүндүргөн эрежелердин топтомун **маалыматтар базасын уюштуруу модели** деп аташат.

Маалыматтар базасынын негизги үч моделин айырмалашат:

1. Иерархиялык;
2. Тармактык;
3. Реляциялык.



Иерархиялык модель чокулары жазуулардан турган дарак сымал түзүлүштү элестетет. Мисалы, компьютерде сакталып турган файлдар жөнүндө маалыматтарды чагылдырган каталогдордун структурасы. Бул модель тамыры деп аталган биринчи деңгээлдеги бир чокуга гана ээ. Тамырдан тышкары ар бир деңгээлдин түйүндөрү жогорку деңгээлдин бир гана түйүнү менен жана төмөнкү деңгээлдин бир нече түйүнү менен байланышкан. Башкача айтканда, мындай байланыш «бирөөдөн – көптүккө» деп аталат.



Тармактык модель, вертикалдык байланыштан тышкары горизонталдык байланыштарга да ээ, б.а. бир деңгээлдин ар бир элементи башка деңгээлдин каалаган сандагы элементтери менен байланышкан болушу мүмкүн. Мисал катары бир класста окугандардын кызыгуусу жөнүндө маалыматтар сакталган МБ айтсак болот: бир окуучунун көптөгөн кызыгуулары болушу мүмкүн, ал эми бир кызыгууга көп окуучулар ээлик кылышы мүмкүн. Мындай байланыш «көптөн – көпкө» деп аталат.

Реляциялык модель (англ. relation - катыш) – бул бири-бири менен байланышта болгон таблицалардын жыйындысы. Реляциялык маалыматтар базасында байланыштардын бардык түрү колдонулушу мүмкүн: «бирөөдөн – бирөөгө», «бирөөдөн – көпкө», «көптөн-көпкө».

Реляциялык МБда сапчалар жазуулар деп, ал эми мамычалар – талаалар деп аталат. Ар бир таблица андагы ар бир жазууну идентификациялоого мүмкүндүк берген ачкычтык талааны (ID) камтуусу зарыл. Ошондуктан, көптөгөн азыркы маалыматтар базасын башкаруу системалары (мындан ары МББС) реляциялык МБна багыт алышат.

Эң популярдуу МББС: MySQL, Oracle, PostgreSQL, SQLite, Firebrid, DB2, Microsoft Access, Microsoft SQL Server.

Open Office.org Base маалыматтар базасын башкаруу системасы

Ар кандай категориялардагы маалыматтарды камтыган татаал бири-бири менен байланышкан берилиштерди иштетүү үчүн маалыматтар базасын башкаруу системалары (МББС) колдонулат.

Open Office.org Base мисалында МББС иштөө принцибин карап көрөлү. Мисалы, бизге сүрөттөрдүн онлайн галлереясын түзүү керек болду дейли. Ал үчүн ар бир сүрөт боюнча бардык актуалдуу маалыматтар сакталган маалымат базасын түзүү керек:

а) сүрөттүн автору;
б) тартылган жылы;
в) жанры (портрет, натюрморт же пейзаж);
г) сүрөттүн аталышы.

Эми, онлайн галереясынын көрүүчүсү сайтка кирип, көрүү үчүн конкреттүү автордун гана сүрөтүн же бир гана портреттик жанрды тандап алышы мүмкүн.

Программанын негизги терезесинин сол жагында негизги төрт баскыч жайгашкан: *таблицалар, сурамдар (запросы), формалар жана отчеттор*. Алар төмөнкүлөрдү түзүүгө мүмкүндүк берет:

ЭСИҢЕ ТУТ

МБда сапчалар жазуу, ал эми мамычалар талаа деп аталат. Ар бир таблица андагы ар бир жазууну бир маанилүү идентификациялаганга мүмкүндүк берүүчү ачкычтык талааны (ID) камтышы керек.

- **Таблицалар** – маалыматтарды сактоо үчүн;
- **Сурамдар** – берилген критерий боюнча маалыматты тандоо үчүн;
- **Формалар** – берилиштерди таблицага тез жана ыңгайлуу киргизүү үчүн;
- **Отчеттор** – колдонуучу үчүн маалыматты ыңгайлуу түрдө чыгарып берүү жана кагазга басып чыгаруу үчүн.

Маалыматтар базасын түзүү

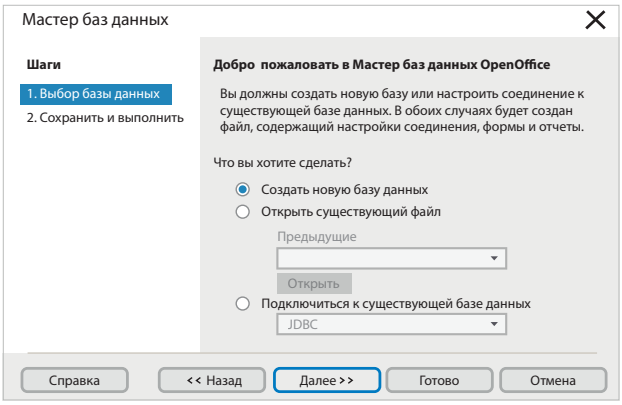
Программаны иштетип баштаганда автоматтык түрдө даяр МБсын ачуу жана түзүү мастери да ишке кирет. Жаңы МБсын түзүү үчүн:

1. «Жаңы МБ түзүү» пунктун тандап, «Кийинки» баскычын баскыла;
2. Экинчи кадам ар бирине экиден жоп болгон эки суроого ээ. 1-суроого жоп адатта «Ооба, МБсын каттоо» болот жана 2-суроого жоп «Базаны редакциялоо үчүн ачуу» болот. «Даяр (Готово)» баскычын баскыла;
3. Андан ары маалыматтар базасын кайсы бир ат менен, мисалы Gallery деген ат менен сактоо.

МБнын таблицасын түзүү

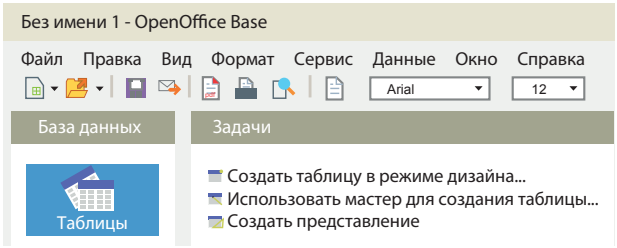
МБнын таблицасын түзүү үчүн «МБ (База данных)» терезесинен «Таблицалар» белгисин тандап жана таблица түзүүнүн кайсы бир ыкмасын тандап алуу керек:

- Таблицаны дизайн режиминде түзүү, б.а. өз алдынча талаанын атын коюу, маалыматтар тибин тандоо ж.б.



• Таблица түзүү үчүн мастерди колдонуу, б.а. даяр талаалары менен таблицалардан тандап алуу.

МБ түзүү үчүн «Дизайн режиминде таблица түзүү» дегенди тандайбыз, ачылган терезеде «Талаанын атын» жана «Талаанын тибин» киргизебиз:



Название поля	Тип поля
автор	Текст [VARCHAR]
название	Текст [VARCHAR]
жанр	Текст [VARCHAR]
год	Дата [DATA]

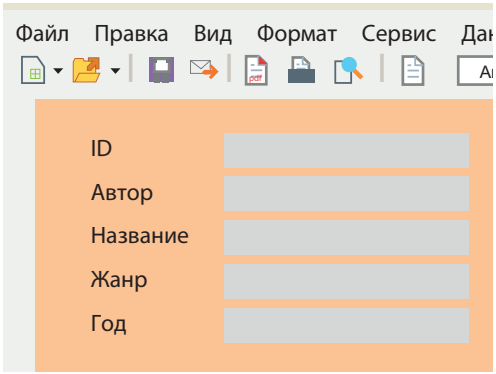
Таблицанын талаалары ар кандай типтеги берилиштерге ээ болушу мүмкүн, мисалы:

- текстик;
- сандык;
- логикалык;
- дата/убакыт.

Файлды биринчи жабууда жана сактоодо, программа «Баштапкы ачкычты түзөйүнбү?» деп сурайт. Мында «Ооба» дегенди тандоо керек, натыйжада сиздин таблицага «ID» деген ат менен дагы бир талаа кошулат.

Форманы түзүү

Эми колдонуучунун интерфейсин түзөлү, б.а. МБнын таблицасына берилиштерди киргизүүгө мүмкүн болгон форманы түзөбүз. Ал үчүн Форма баскычын басып «Формалар мастери» пунктун тандап алгыла. Түзүлгөн таблицанын негизинде «Формалар мастери» жөнөкөй кадамдар аркылуу форманы түзүп берет. Ушул жерден силер форманын сырткы келбетин да тандап аласыңар.



Сурамдарды түзүү

Маалыматтар базасынын таблицаларынын негизинде сурамдарды да түзсө болот, б.а. силерге керектүү берилиштерди гана чыпкалап алса болот. Сурамды ачып, силер сураган параметрлер менен берилиштерди таблица түрүндө көрө аласыңар.

Таблица жана формалар сыяктуу эле сурамды да «Сурамдар мастери» режиминде жана «Сурамдар дизайны» режиминде түзө аласыңар.

Сурамды аныктоо үчүн талааларды көрсөтүү шартын жана анда киргизиле турган маалыматтар базасынын талааларынын атын көрсөтүү керек. Мисалы, биздин галереянын маалыматтар базасында, конкреттүү авторлордун сүрөтүн жана портреттик жанрдагы гана түрүн тандап алууга болот.

Отчетторду түзүү

Отчет – бул силердин МБдагы берилиштерди чагылдырган тексттик документ. Силер отчеттун өлчөмүн, сырткы келбетин жана структурасын өзгөртсөңөр болот.

Таблицанын же сурамдын негизинде түзүлүп жаткан отчеттор бир же бир нече таблицадагы берилиштерди көрсөтүшү жана талаалардын каалагандай комбинацияларын камтышы мүмкүн.

Отчетту тез арада түзүү үчүн «Отчетторду түзүү мастерин» колдонсо болот. Ал жерде этап менен:

а) берилиштердин булагын тандоо (берилиштер алына турган таблица же сурам);

б) көрсөтүлө турган талааларды тандап алуу;

в) отчеттун сырткы келбетин аныктоо (талаалардын жайышы, элементтерди чагылдыруу тиби жана фон) керек.

Эми даяр отчетту кагазга басып чыгарсаңар болот.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Иерархиялык жана тармактык маалыматтар базасына мисал келтиргиле. Open Office каражаттарынын жардамы менен алардын схемаларын түзгүлө.*

2) *Талаалардын төмөнкү типтерин колдонуп «Менин досторум» маалыматтар базасын түзгүлө: тексттик, сандык, дата/убакыт, логикалык.*

3) *Төмөнкү маалыматтарды камтыган китепкананын маалыматтар базасын иштеп чыккыла: китептин аты, авторлордун аты-жөнү, басмакананын аты, чыккан жылы, беттердин саны, иллюстрациялардын саны, конкреттүү китептин нускаларынын саны, ушул китеп берилген окуучулардын саны. Маалыматтар базасы таблицаны, сурамды жана форманы камтысын.*

3 - бөлүм

Программалоо

3.1-тема:

Татаал шарттар: and, or, not

Программалоодо шартты туура коё билүү өтө маанилүү. Көпчүлүк учурда шарттар татаал болушат, б.а. «ЖАНА», «ЖЕ» жана «ЭМЕС» логикалык операторлору (байламта) менен бириккен бир нече курама шарттардан турушат. Python тилинде алар «and», «or», «not» деген англис сөздөрү менен жазылат.

`and` логикалык оператору (логикалык көбөйтүү)

Туюнтмада `and` байламтасы менен бириккен курама шарттардын баары тең `True` маанисине барабар болсо, анда татаал шарт `True` маанисин кайтарат. Эгер эки туюнтманын бирөөсү эле жалган болсо, анда шарт жалган:

```
x = 5
if x < 10 and x % 3 == 0:
      print('True')
else:
      print('False')
```

Бул жерде жооп False болот, анткени шарттын экинчи бөлүгүнө ылайык берилген 5 саны 3кө калдыксыз бөлүнбөйт. Эгерде биз туюнтманы x%3==0 `and` x<10 деп жаза турган болсок, ал кайрадан эле False маанисин кайтармак. Бирок, мындагы экинчи салыштыруу x<10 интерпретатор тарабынан аткарылмак эмес, анткени аны аткаруунун кажети жок. Анткени биринчи туюнтма (x%3==0) жалган болгондуктан `and` операторунун болушу бардык туюнтманы жалганга чыгарды.

`or` логикалык оператору (логикалык кошуу)

Эгерде жок дегенде бир туюнтма True маанисине ээ болсо, True маанисин кайтарат:

```
x = 5
if x < 10 or x % 3 == 0:
      print('True')
else:
      print('False')
```

Бул жерде жооп True болот, анткени шарттын биринчи бөлүгүнө ылайык берилген 5 саны 3кө калдыксыз бөлүнбөсө да 10 санынан кичине. Мына ушул үчүн эгерде эки туюнтманын бирөөсү эле True маанисин кайтарса, анда экинчи туюнтма бааланбайт, анткени `or` оператору баары бир True маанисин кайтарат.

`not` логикалык оператору (логикалык тануу)

`not` *унардык* оператору чындыкты жалганга, ал эми жалганды чындыкка айландырат. *Унардык* дегенибиз, анткени ал `and` жана `or` операторлорундай болуп анын оң жагында же сол жагында турган туюнтмаларга эмес андан кийин турган бир эле туюнтмага колдонулат.

1-вариант:

```
x = 8
print (not x < 15)

False
```

2-вариант:

```
x = 8
print (not x > 15)

True
```

Эгерде бир туюнтмада бир эле убакта бир нече же бардык логикалык операторлор колдонулса, анда аткаруу тартиби төмөнкүдөй болот:

1) катыштар (<, >, <=, >=, ==, !=);
2) not («ЭМЕС»);
3) and («ЖАНА»);
4) or («ЖЕ»).

Аракеттердин иретин өзгөртүү үчүн тегерек кашаалар колдонулат. Кашаалар пайда болгон учурдагы эсептөөлөрдүн иретинин өзгөрүшүн мисалда карап көрөлү:

1-вариант:

```
a=4
b=6
c=8
result = c==8 or b<a and not a < 7
print (result)
Жыйынтык:
True
Эмне үчүн экендигин түшүндүрөлү:
   c == 8 or b < a and not a < 7
    True   False       True
                       False
           False
           True
```

2-вариант:

```
a=4
b=6
c=8
result= (c==8 or b<a) and not a < 7
print (result)
Жыйынтык:
False
Эмне үчүн экендигин түшүндүрөлү:
  (c == 8 or b < a) and not a < 7
   True   False       True
          True        False
                      False
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) and *операторунун жардамы менен бирөөсү чындыкты, экинчиси–жалганды көрсөткөн эки татаал логикалык туюнтманы түзгүлө.*
2) *Жогорудагы маселени* ***or*** *операторун колдонуп аткаргыла.*

3.2-тема:

Тизмелер, кортеждер жана сөздүктөр

Көпчүлүк учурда бизге көптөгөн окшош маалыматтарды бир файлда чогултууга туура келет, мисалы, окуучулардын тизмеси же маалымдамадагы телефон номерлер. Python тилинде мындай маалыматтардын топтомун **тизмелерде, кортеждерде** жана **сөздүктөрдө** уюштурса болот.

Тизме (***list***) – бул белгилүү иретте жайгашкан элементтерден турган структура. Ар бир элементке ага кайрылууга мүмкүн болгон номер (же индекс) туура келет. Тизмени түзүү үчүн квадраттык кашаанын ичине үтүр менен ажыратылып, анын бардык элементтери тизмектелет.

Мисалга өзүбүздүн үй-бүлө мүчөлөрүбүздүн тизмесин түзөлү:

```
>>> myfamily = ['father', 'mother', 'sister', 'brother']
```

Тизме myfamily деген өзгөрмөдө сакталды жана 4 элементтен турат. Элементтердин номерлөөсү 0дөн башталгандыктан, ‘father’ 0чү, ал эми ‘brother’ элементи 3-индексте турат. Терс индекстерди да колдонсо болот. Анда номерлөө аягынан башталып эң акыркы элемент -1 индексине ээ болот.

Тизме бош болушу да мүмкүн:

```
>>> myfamily []
```

Тизме ар кандай типтеги объекттерди камтышы мүмкүн. Тизмеге бир убакта сапты, сандарды, башка типтеги объекттерди киргизүүгө болот:

```
objects = [1, 2.6, 'Hello', True]
```

Эми тизме менен иштей турган программаны карап көрөлү. Мисалга, үй-бүлөнүн ар бир мүчөсү менен саламдашуу программасын жазалы:

Python тилиндеги программа	Экранга чыккан жыйынтыгы

`myfamily = ['father', 'mother', 'sister', 'brother']`
`for item in myfamily:`
`  print('Hello', item)`
	Hello father
	Hello mother
	Hello sister
	Hello brother

Берилген мисалда for циклин колдонуп тизменин бардык элементтерине төмөнкү бир команданы иштеттик:

```
print ('Hello', item)
```

Тизменин өзүн да өзгөртсө болот: ага жаңы элементтерди кошуп же бар элементтерин өчүрүп же өзгөртсө болот. Жаңы элементти кошуу үчүн append() методун колдонуп көрөбүз:

```
myfamily.append ('uncle')
    print (myfamily)
>>>
['father', 'mother', 'sister', 'brother', 'uncle']
```

Элементти өчүрүү үчүн remove() методу колдонулат. Кашаага элементтин мааниси жазылат. Эгерде тизмеде мындай маанилер бир нече болсо, анда биринчиси гана өчүрүлөт.

```
myfamily.remove('sister')
    print (myfamily)
>>>
['father', 'mother', 'brother','uncle']
```

Тизменин элементтери менен иштөө үчүн дагы көптөгөн башка методдор бар. Аларды биз кененирээк «Тизмелерди иштетүүчү алгоритмдер» темасында карайбыз.

Тизменин өзүнчө элементине индекси боюнча кайрылуу үчүн өзгөрмөнүн атынан кийин чарчы кашаага анын индексин көрсөтүү керек:

```
print (myfamily[3]) #жыйынтыгы uncle
```

Тизмелерди бири-бирине кошсо болот, анда жаңы тизме эки тизмедеги тең элементтерди камтып калат:

```
x = [1, 2, 3, 4]
y = [5, 6, 7, 8]
z = x + y
print (z)  #жыйынтык [1, 2, 3, 4, 5, 6, 7, 8]
```

Тизмелер менен ар түрдүү көп амалдарды жасаса болот:

x in a	**x** элементи **a** тизмесинде бар же жок экенин текшерет.True же False маанисин кайтарат.	`a = [1, 2, 3, 4, 5, 6, 7, 8]` `print (2 in a)#жыйынтык True`
min(a)	**a** тизмесинен эң кичине элементти табуу.	`a = [1, 2, 3, 4, 5, 6, 7, 8]` `print (min(a)) #жыйынтык 1`
max(a)	**a** тизмесинен эң чоң элементти табуу.	`a = [1, 2, 3, 4, 5, 6, 7, 8]` `print (max(a)) #жыйынтык 8`

Кортеж (***tuple***) тизме сыяктуу эле элементтердин удаалаштыгынан турат. Бирок, андагы элементтерди өзгөртүүгө, кошууга же өчүрүүгө болбойт. Кортеждерди түзүү үчүн үтүр менен бөлүнгөн анын маанилери жайгашкан тегерек кашаалар колдонулат:

```
user = ('Timur', 23, 1/10/1998)
print(user)
```

Кортеждерде объекттердин касиеттерин сактоо ыңгайлуу, мисалы, атын, жашын, туулган датасын. Эгерде кортеж болгону бир элементтен турса, анда кортеждин жалгыз элементинен кийин үтүр белгисин коюу керек:

```
user = ('Tom',)
```

Сөздүктөр (***dictionary***) – бул ар бир элементи индекстин ордуна уникалдуу ачкычка ээ болгон маалыматтардын структурасы. Сөздүктүн элементтерин өзгөртө берсе болот. Сөздүктөрдү түзүү үчүн фигуралык кашаалар колдонулат ({}):

```
dictionary = {ачкыч1:мааниси1, ачкыч2:мааниси2, ...}
```

mychool атындагы сөздүктү түзөлү:

```
myschool = {'5 klass':'Anara, Kanat, Pavel', '6 klass':
'Chyngyz, Tina, Emil'}
```

Бул сөздүктө ачкыч катары класстын аттары, ал эми мааниси катары – ошол класстарда окугандардын аттары берилди.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *5 сандан турган тизме түзгүлө. Экранга алардын суммасын, максималдык жана минималдык элементтерин чыгаргыла. Чыгаруу үчүн Python тилинде каттылган sum(), max() жана min() функцияларын колдонгула.*

2) *5 элементтен турган тизмени түзгүлө. Тизменин биринчи жана акыркы элементин чыгаргыла.*

3) *a = [1, 2, 3, 5, 8, 13, 21, 34, 55, 89] тизмеси берилди. Экранга for циклин колдонуп тизменин так элементтерин чыгаргыла.*

4) *3 тапшырмада берилген тизмеден: а) 5 жана 34 элементтерин өчүргүлө; б) тизмеге 6 жана 120 сандарын кошкула.*

3.3-тема:

Циклдик алгоритмдер

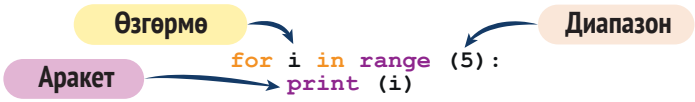
7-класста биз **while** жана **for** циклдерин үйрөнүп баштаганбыз.

for цикли **while** циклинен кайсы бир командаларды алдын-ала белгилүү санда кайталоо үчүн колдонулгандыгы менен айырмаланат.

ОШОНДОЙ ЭЛЕ КАРА:
7-класс *3.4-тема*
while жана for циклдери

Ал эми **while** цикли тескерисинче кайсы бир аракеттерди канча жолу кайталай тургандыгы белгисиз болгон учурларда колдонулат. Бирок анда циклди токтотуучу шарт белгилүү.

for циклинин колдонулушун кеңири карап көрөлү. **for** циклинин Python тилинде жазылышы төмөнкү схема боюнча жүрөт:



Берилген схемада **for** цикли көрсөтүлгөн диапазондогу бардык элементтерди терип чыгат. Циклде ар бир элементке **for** тулкусунда жазылган аракеттер колдонулат. **for** сөзүнөн кийин жазылган **i** өзгөрүлмөсүнө циклдин ар бир өтүүсүндө диапазондогу кезектеги элемент ыйгарылат.

Келгиле, мисалды карап көрөлү: **letter** өзгөрмөсүнө ар бир жолу ***Python*** сабынын жаңы элементи ыйгарылып турсун. `print` командасы экранга бул саптын ар бир тамгасын бирден чыгарат:

```
for letter in 'Python':
    print (letter, 'тамгасы', )
>>>
P тамгасы
y тамгасы
t тамгасы
h тамгасы
o тамгасы
n тамгасы
```

Төмөнкү мисалда ар бир кийинки өтүүдө өзгөрмөнүн мааниси берилген диапазондогу санга көбөйүп турат:

```
f = 12
for i in range(1,6):
    f = f + i
print (f)
>>>
27
```

«for i in range(1,6)» цикли беш жолу аткарылат (6 – кирбейт). Циклдин ар бир кадамында **f** өзгөрмөсү **i** санына өсүп турат. Баштапкы мааниси **f** = 12. Циклде маанилери өзгөрүп турат:

1-өтүү: f = 12+1=13
2-өтүү: f = 13 +2=15
3-өтүү: f = 15+3=18
4-өтүү: f = 18+4=22
5-өтүү: f = 22+5=27

Кыскача мындай кылып жазсак болот: f = 12+1+2+3+4+5 = 27

range() функциясынын аргументтери төмөнкүдөй берилет:

- **range (x)** – 0дөн х-ке чейинки маанилерди алат, бирок х – диапазонго кирбейт;
- **range (y, x)** – у-тен х-ке чейинки бардык маанилерди алат, мында да х диапазонго кирбейт;
- **range (y, x, s)** – у-тен х-ке чейинки бардык маанилерди s кадамы менен алат.

Мисалы:

```
for i in range(0,15,3):
    print(i)
```

Берилген мисалда for цикли 0дөн 15ке чейинки маанилерди 3 кадам менен алат, жыйынтыгында ал ар бир үчүнчү санды чыгарып берет:

```
>>>
0
3
6
9
12
```

Андан тышкары кадам үчүн терс сандарды да колдонсо болот, анда цикл маанилерди тескери багытта тандап ала баштайт:

```
for i in range(100,0,-20):
    print(i)
>>>
100
80
60
40
20
```

for циклинен айырмаланып **while** цикли саны менен эмес, логикалык шарты менен жетектелет. Ошондуктан кодду канча жолу аткарарынын так санын билүүнүн кажети жок.

while циклинин коду логикалык шарт чындык маанисинде (True) болуп турганга чейин кайталана берет.

1-маселе. while циклдин негизинде оюндун программасын түзүп көрөлү. Мында колдонуучу компьютер тарабынан катылган санды табышы керек:

```
import random #кокустук сандар модулун жүктөйбүз
number = random.randint(1, 25) #компьютер кокустук санды тандайт
choices = 0  #choices өзгөрмөсүнө аракеттердин санын жазабыз
while choices < 5: #циклди 5 аракетке чейин аткарат
    print('1ден 25ке чейинки санды тап:')  #колдонуучуга
санды киргизүүнү сунуштайт
    guess = input()
    guess = int(guess) #киргизилген сан бүтүн болуш керек
    choices = choices + 1 #ар бир аракетте эсептөө 1ге өсүп турат
    if guess == number:  #эгерде киргизилген сан катылган санга
барабар болсо
        break  #программаны токтот
```

choices өзгөрмөсүнө 0 мааниси ыйгарылган. Ал санды табуу боюнча жасалган ар бир аракет сайын көбөйө берет. Биз программа чексиз циклге түшүп калбашы үчүн аракеттердин санын бешөө менен чектейбиз.

Программа иштеп жатат, бирок ал колдонуучуга эч кандай жыйынтыкты билдирбейт: колдонуучу катылган санды таптыбы же тапкан жокпу, билбейт. Жыйынтыгы мындай көрүнөт:

```
1ден 25ке чейинки санды тап:
5
1ден 25ке чейинки санды тап:
16
```

```
1ден 25ке чейинки санды тап:
7
1ден 25ке чейинки санды тап:
18
1ден 25ке чейинки санды тап:
10
>>>
```

Ал үчүн колдонуучуга анын саны катылган сандан чоң же кичине экенин билдирип тургандай шарттуу операторлорду киргизели. Бул болсо санды тезирээк тапканга жардам берет:

```
import random
number = random.randint(1, 25)
choices = 0
while choices < 5:
    print('1ден 25ке чейинки санды тап:')
    guess = input()
    guess = int(guess)
    choices = choices + 1
    if guess < number: #эгер сан катылган сандан кичине болсо
        print('Менин саным сеникинен чоң')
    if guess > number: #эгер сан катылган сандан чоң болсо
        print('Менин саным сеникинен кичине')
    if guess == number: #эгер сан катылган санга барабар болсо
        break
if guess == number:
    print('Азамат! Сен санды ' +str(choices)+ ' аракеттен
кийин таптың!')
else:
    print('Тилекке каршы, сен санды тапкан жоксуң. Мен ' +
str(number) + ' санын каткам')
```

Эгерде программаны ишке киргизсек, анда колдонуучу менен баарлашуу варианты төмөнкүдөй болот:

```
1ден 25ке чейинки санды тап:
6
Менин саным сеникинен чоң
1ден 25ке чейинки санды тап:
17
Менин саным сеникинен кичине
1ден 25ке чейинки санды тап:
14
```

```
Менин саным сеникинен чоң
1ден 25ке чейинки санды тап:
15
Азамат! Сен санды 4 аракеттен кийин таптың!
>>>
```

Эми программа колдонуучуга санды табуу үчүн жардамдашат. Мисалы, компьютер 15 санын катса, ал эми колдонуучу 17 санын киргизсе, программа киргизилген сан катылган сандан чоң экендигин көрсөтөт.

`while` циклинин жыйынтыгы (тапты же тапкан жок) негизги программага берилет. Тапса, программа ал сан канча аракеттен кийин табылгандыгын, ал эми таппаса кайсы сан катылгандыгын экранга чыгарат.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *12, 34, 8, 15, 9, 26 сандарынан турган тизме берилген. Экранга төмөнкү сандарды чыгаргыла: а)* ***n*** *санынан кичине; б) биринчи саны* ***n****ден чоң; в)* ***n*** *санынан чоң бардык сандардын суммасын.*

2) *Төмөнкү программанын иштөөсүнүн жыйынтыгында алынган* ***y*** *өзгөрмөсүнүн маанисин жазгыла:*

```
y = 5
for i in range(2,6):
      y = y + 4 * i
print (y)
```

3) *Берилген код боюнча алгоритмдин ар бир кадамындагы өзгөрмөлөрдүн маанилерин таблицага жазгыла:*

k=4 p=1040 m=2

```
while p != m*m:
      k=k+1
      p=p-4
      m=m*2
print (k)
```

k	p	m	m*m

4) *Майнкрафттын каарманы Алекстин мүнөтүнө 4 минерал чыгара турган машинасы бар. Ар бир 100 минералга дал ушундай эле мүнөтүнө 4 минерал чыгарган машина курууга болот. Бир сааттан кийин Алекстин канча машинасы боло тургандыгын аныктоочу программаны жазгыла.*

3.4-тема:

Камтылган шарттуу амалдар жана циклдер

Көптүк тармактануу

7-класста биз силер менен `if` жана `else` операторлору кандай иштей тургандыгын өздөштүргөнбүз. Программада алар аткаруунун эки вариантын көрсөтөт. Бирок программанын алгоритми экиден көп жолду тандоону сунушташы мүмкүн, мисалы үчөөнөн, төртөөнөн же андан көптөн.

Муну бир нече шарттуу операторлорду же **көптүк тармактануу** колдонуп ишке ашырса болот. Бул учурда `if` жана `else` сөздөрүнөн кийин жаңы шарттуу оператор жазылат. Мисалды карап көрөлү:

```
if number > 0:
    print('Сан 0дөн чоң')
else:
    if number == 0:
        print('Сан 0гө барабар')
    else:
        print('Сан 0дөн кичине')
```

Барабардыкты текшерген экинчи `if` шарттуу оператору `else` блогунун ичинде жайгашкан жана камтылган шарттуу оператор болуп саналат. Качан `else` операторунан кийин `if` оператору кеткен учурларда аларды `elif` (`else-if` тин кыскартылышы) деген бир оператор менен алмаштырса болот. `else`ден айырмаланып `elif`тин баш сөзүндө `if` операторундагыдай сөзсүз шарт болушу керек. Мындай конструкцияны бир деңгээлдеги камтылуунун көптүк тармактануусу же каскаддык тармактануу деп аташат.

Туюнтмалардын тең күчтүү жазылыштары

```
else:
    if number == 0:
```

```
elif number == 0:
```

1-маселе. Көптүк тармактануу талап кылынган маселени карайбыз: берилген **x** жана **y** координаталары боюнча координаттык тегиздиктин чейрегин аныктоо керек. **x** жана **y** өзгөрмөлөрүндө клавиатурадан киргизилген бүтүн сандык маанилер сакталат.

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print('Биринчи чейрек')
elif x > 0 and y < 0:
    print('Төртүнчү чейрек')
elif y > 0:
    print('Экинчи чейрек')
else:
    print('Үчүнчү чейрек')
```

Келтирилген программада `if, elif, else` шарттары кезеги менен текшерилет жана биринчи чыныгы шартка дал келген блок аткарылат.

Камтылган циклдер

Эгер цикл башка циклдин ичинде жайгашса, ал камтылган цикл деп аталат. Б.а. циклдин ар бир кадамында ошондой эле циклдик алгоритмди туюндурган аракеттер аткарылат. Ал мындай иштейт: биринчи жүрүштө сырткы цикл ички циклди чакырат. Ал өзүнүн акырына чейин аткарылгандан кийин башкаруу кайрадан сырткы циклдин тулкусуна берилет. Экинчи жүрүштө сырткы цикл кайрадан ички циклди чакырат. Ушинтип бул процесс сырткы цикл аяктамайынча кайталана берет.

2-маселе. Экранга көбөйтүүнүн таблицасын чыгарабыз. Ал үчүн сырткы циклде 1ден 9га чейинки сандарды терип чыгуу керек. Ал сандардын ар бирине ички циклде 1ден 9га чейинки сандарды терип чыгуу керек.

```
1 2 3 4  5  6  7  8  9
2 4 6 8 10 12 14 16 18
```

Бир көк цифрага 9га чейинки цифралардын бир катары туура келет. 1 жана 2 көк цифралар сырткы циклде, ал эми кара цифралар ички циклде жайгашкан.

Мындагы ички циклде биринчи катардагы сырткы жана ички циклдердин өзгөрмө-эсептегичтерин көбөйтүш керек.

Ушундай жол менен сырткы циклдин бир аткаруусуна ички циклдин 9 аткаруусу туура келет жана көбөйтүүнүн таблицасынын бир сабы түзүлөт. Ар бир саптан кийин жаңысына өтүү керек: бул ички цикл аткарылып бүткөндөн кийин, сырткы циклде жүргүзүлөт.

Андан тышкары, таблицаны тургузуу үчүн форматталган киргизүүнү колдонуш керек, б.а. мамычалардын туурасын (\t) берүү керек, антпесе сандар жылышып калат, анткени ар бир саптагы цифралардын саны ар башка.

Биздин код мындай болуп көрүнөт:

```
for i in range(1,10):#1ден 9га чейинки биринчи көбөйтүүчү
    for j in range(1,10):#1ден 9га чейинки экинчи көбөйтүүчү
        print(i*j, end='\t')
    print()
```

Жыйынтык:

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

3-маселе. Экранда эки түрдөгү символдордун жардамы менен «тик бурчтук» тарталы. Тик бурчтуктун четтери «0» символу менен, ал эми анын ичи «1» символу менен тартылсын.

ЭСИҢЕ ТУТ

Цикл **n** жолу кайталанышы үчүн диапазондун акыркы саны **n+1** болуш керек.

Мейли тик бурчтуктун узундугу 20 символго жана туурасы 7 символго барабар болсун. Сырткы цикл саптарды терип жатып, биринчи жана акыркы цифраларга 0ду коюш керек. Эгерде сап биринчи же акыркысы болсо (эсептөө 0дөн башталгандыктан 0чү жана 6-саптар), 0 башынан аягына чейин тизилип чыгат. Калган бардык учурларда 1 цифрасын коёбуз. Программаны жазалы:

```
for i in range(7):              #сапты 7 жолу чыгарабыз
    if i==0 or i==6:            #эгерде сап 1чи же акыркы болсо
        for j in range(20):     #бардык 20 жолу
            print('0',end='')       #0ду чыгарабыз
    else:                           #антпесе
        print('0',end='')           #0ду чыгарабыз
        for j in range(1,19):       #1чи жана 19дан башкасына
```

```
            print('1',end='')     #1 цифрасы менен чыгарабыз
        print('0',end='')
    print()
```

Break жана continue операторлору

Кээде бизге, белгилүү бир шарттарда программаны мажбурлап циклден чыгаруу талап кылынат. Мындай учурлар үчүн **break** оператору колдонулат. Эгерде цикл кайсы бир белгилүү өтүүнү өткөрүп жана кайра улантылышын кааласак, анда **continue** операторун колдонобуз.

Операторлорду мисалдарда салыштыралы:

```
for i in [1,2,3,4,5]:
   if i == 3:
      break
   print ('тизменин элементи:', i)
>>>
тизменин элементи: 1
тизменин элементи: 2
```

```
for i in [1,2,3,4,5]:
   if i == 3:
      continue
   print ('тизменин элементи:', i)
>>>
тизменин элементи: 1
тизменин элементи: 2
тизменин элементи: 4
тизменин элементи: 5
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

__1)__ Колдонуучудан санды киргизүүнү сураган программа жазгыла. Эгер ал сан оң болсо, анда экранга 1 цифрасы чыгарылсын. Эгерде сан терс болсо, анда -1 цифрасы, эгер киргизилген сан 0 болсо, анда экранга 0 чыгарылсын. Коддо көп тармактанган шарттуу операторду колдонгула.
__2)__ 1ден 15ке чейинки сандардын тизмесинен 3кө бөлүнгөн бардык сандарды алып салган циклдик программаны жазгыла. __continue__ операторун колдонгула.
__3)__ 2-тапшырмадагы сандардын тизмесинен 7ден кичине болгон сандарды экранга чыгаргыла. __break__ операторун колдонгула.
__4)__ 5 кг, 10 кг жана 15 кг алма бата турган ящиктер берилди. 100 кг алманы бөлүштүрүү үчүн канча ар кандай өлчөмдөгү ящик керек боло тургандыгын аныктоочу программаны түзгүлө.

3.5-тема:

Функциялар

Программалоодогу функция – бул коюлган аты менен ага кайрылып, чакырууга мүмкүн болгон коддун обочолонгон аймагы. Чакырууда функциянын тулкусунда жазылган командалардын аткарылышы жүрөт. Функцияларды өз алдынча иштей албаган, бирок кадимки программаларга камтылып киргизилген анча чоң эмес программалар менен салыштырууга болот.

Мурда силер интерпретатордун өзүндө **орнотулган** функцияларды колдонуп келгенсиңер, мисалы:

`print()` – *экранга тегерек кашаанын ичиндегилердин бардыгын чыгарат;*
`str()` – *берилиштерди саптык типке өзгөртүп түзөт;*
`int()` – *берилиштерди бүтүн санга өзгөртүп түзөт;*
`float()` – *бүтүн сандарды чыныгы сан тибине өзгөртүп түзөт;*
`round()` – *чыныгы санды жакынкы бүтүн санга чейин тегеректейт.*

Булардан башка биз тигил же бул маселелерди аткартуу үчүн өзүбүздүн функцияларды түзүп алсак болот. Бул үчүн Python тилинде эгер кайсы бир алгоритм (же фрагменти) кайталанып жатса, аны функция катары жазууга боло турган мүмкүнчүлүк бар.

Ал үчүн жаңы функцияга ат берип жана анын алгоритмин баяндоо керек. Мындан кийин программада функциянын атына кайрылганда эле өзүнүн кирген жана чыккан берилиштери менен тиешелүү алгоритм ишке кирет. Функцияны аткаргандан кийин программанын иши функцияны чакырган командадан кийин кайра улана берет.

Мисалга, программанын бир нече жеринде экранга «Программада ката» деген билдирүүнү чыгарыш керек болуп жатат. Аны мындайча жасасак болот:

```
print ('Программада ката')
```

Бул чыгаруу операторун керек болгон жердин бардыгында коё берсек, анда бул эс тутумду толтуруп жибериши мүмкүн. Эгерде билдирүүнүн текстин өзгөртүш керек болуп калса, анда бул чыгаруу операторлорун бүткүл программанын ичинен издеш керек болот. Мына так ушундай учурлар үчүн кошумча алгоритм – функциялар колдонулат, аларга программанын каалаган жеринен кайрыла берсе болот. `error` функциясын жазалы:

```
def error():
      print ('Программада ката')
n = int (input())
if n < 0:
      error()
```

Биз `error` деген жаңы функцияны киргиздик.

Фукциянын аты `def` (англ. *define* - аныктоо) ачкыч сөзү менен башталат, андан кийин гана функциянын уникалдуу аталышы берилет (мисалы: `def sum`). Аталышынан кийин функциянын параметрлери киргизилген кашаалар жана кош чекит коюлат. Функциянын тулкусу жылдыруу менен жазылат. Функцияны программанын башка жеринде иштетиш үчүн, анын аталышы менен (кашааларын кошуп) чакырыш керек. Мисалы: **error()**.

Эгерде кандайдыр бир аракеттер программанын ар кайсы жерлеринде бир нече жолу кайталанса, анда функцияны колдонуу кодду бир топ кыскартууга мүмкүндүк берет. Кээде өтө чоң программаны жөнөкөйлөтүп жана ыңгайлуу кылуу үчүн бир нече функцияларга бөлүп алышат. Анын татаал алгоритминин өзүнчө этаптарын функциялар түрүндө беришет. Мындай ыкма бардык программаны түшүнүктүү кылат.

Функция жана анын аргументтери

Аргумент – бул функциянын чакыруусунда ага бериле турган маани. **Функциянын параметри менен аргументин алмаштырбаш керек.** Функциянын параметрлери аны түзүүдө бир гана жолу берилет, ал эми аргументтер ага ар бир колдонууда берилип турат. Б.а. аргументтер параметрлердин мааниси десек болот.

Мисалга, көптөгөн бирдей символдун жардамында бөлгүч (сызык) тарткан программаны жазалы:

```
n = 125 #ушунча жолу
s = '_' #символ
while n > 0:
      print (s, end = '')
      n -= 1
```

`print` функциясына эки аргумент берилген. Экинчи аргументи «`end = ''`». Негизи `end`, `print` функциясынын ар бир кийинки жыйынтыгы мурункусуна кандайча бириккенин аныктайт. Биздин

ЭСИҢЕ ТУТ

Функциянын аты кичинекей латын тамгаларынан туруш керек, ал эми сөздөр бири-биринен төмөнкү сызык символу менен ажыратылышы керек. Бул кодду окуу үчүн ыңгайлуу кылат (snake_case).

учурда «**end = ''**» бардык жыйынтыктарды бир сапка жазып берет. Эгерде «**end = ','**» деп жазылса, анда бардык жыйынтыктар бир сапчага үтүр менен ажыратылып жазылат.

Бөлгүч сызыкты тартуу алгоритмин s жана n параметрлери менен өзүнчө функцияга өзгөртүп түзсө болот. Биринчиси кандай символ чыгарыш керектигин, экинчиси – канча символ чыгаруу керектигин аныктайт. Программаны жазалы:

```
def print_char(s, n): #функциянын аты жана параметрлери
      k = n
      while k > 0:
            print (s, end = '')
            k -= 1
print_char ('-', 10)  #негизги программа аргументтери
```

Негизги программа **print_char** функциясын чакыруунун болгону бир командасын гана камтыйт. Кашаанын ичинде тире («-») символун 10 жолу чыгарыш керектигин көрсөтүүчү функциянын аргументи көрсөтүлгөн.

Глобалдык жана локалдык өзгөрмөлөр

Функциялардагы өзгөрмөлөр глобалдык же локалдык болушу мүмкүн. **Локалдык өзгөрмөлөр** функцияга аргументтер аркылуу берилет. Локалдык өзгөрмөлөр ошол функциянын гана «көрүнүү зонасында» жайгашат жана программанын калган бөлүгүнө жеткиликтүү эмес. Ал эми **глобалдык өзгөрмөлөр** программанын бардыгында жеткиликтүү. Аларга аты боюнча кайрылса болот жана аны менен байланышкан маанилерди алса болот.

1-маселе. Колдонуучуну кайсы бир геометриялык фигураны тандап, ал фигуранын аянтын эсептөө үчүн берилиштерди киргизүүнү суранган программанын мисалында өзгөрмөлөрдүн типтерин карайлы:

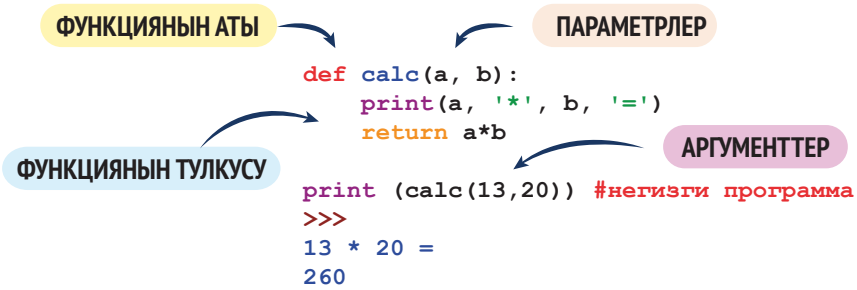
```
def rectangle():
      a = float(input('Туурасы: '))
      b = float(input('Бийиктиги: '))
      s = a*b
      print('Аянты: ', s)
def triangle():
      a = float(input('Негизи: '))
      h = float(input('Бийиктиги: '))
      s = 0.5*a*h
      print('Аянты: ', s)
```

```
figure = input('1-тик бурчтук, 2-уч бурчтук:')
if figure == '1':
      rectangle()
elif figure == '2':
      triangle()
```

Бул маселеде 5 өзгөрмө бар, анын ичинде **figure** гана глобалдуу. **rectangle()** функциясындагы **a** жана **b** жана **triangle()** функциясындагы **a** жана **h** өзгөрмөлөрү – локалдык. Ошону менен бирге бул эки функциядагы **a** локалдык өзгөрмөсү – эки башка өзгөрмө.

Функциядан маанилерди кайтаруу

Өзүнүн жыйынтыктарын функция программанын негизги бутагына берет, башкача айтканда – **маанилерин кайтарат**. Бул үчүн **return** (англ. *кайтаруу*) оператору колдонулат: ал функцияны токтотуп жана анын өзгөрмөлөрүнүн маанисин негизги программага кайтарат. Ал кантип иштерин мисалда карайлы:



Биз көрүп тургандай **return** функциясы маанилердин өзүн кайтарбастан, көбөйтүүнүн жыйынтыгын кайтарып жатат. Бул функция түзүлгөндөн кийин, андан ары негизги программада функцияга эки аргументти гана киргизүү жетиштүү. Ошол замат программа эки сандардын көбөйтүндүсүн чыгарып берди.

1-маселе. Эгерде берилген сандын акыркы эки цифрасынан түзүлгөн эки орундуу сан 4кө бөлүнүүчү сан болсо, анда ал берилген сан 4кө бөлүнөрүн математикадан окуганбыз. Колдонуучу тараптан киргизилген сандын эки акыркы санын бөлүп алган, андан кийин ал сан туура 4кө бөлүнөрүн аныктоочу **div_four** функциясын жазалы:

```
def div_four (n):
    d = n % 100 #сандын акыркы эки цифрасын бөлүп алабыз
    if d%4 == 0:
```

```
        return (True) #4кө калдыксыз бөлүнсө Truену кайтарат
    else:
        return (False)
a = int(input('Санды киргизгиле: ')) #негизги программа
print (div_four(a))
```

Ар бир функция белгилүү бир жыйынтыкты берерин эстен чыгарбаш керек. Эгер силер жыйынтык катары маанисин кайтарууну көрсөтпөсөңөр да, ал **None** (эч нерсе) деген жыйынтыкты берет. Мисалы, функцияны төмөндөгүдөй жазсак, анда колдонуучу 4кө бөлүнбөгөн санды киргизсе, программа экранга **None** деген жыйынтыкты чыгарат:

```
def div_four (n):
    d = n % 100
    if d%4 == 0:
        return (True)
```

2-маселе. Сандын цифраларын кошууну эсептөөчү функцияны түзөлү (мисалы, 147 саны үчүн: 1+4+7=12). Цифраларды кошууну акыркысынан баштайлы, биздин мисалда бул 7 цифрасы.

1. Сандын акыркы цифрасын алуу үчүн, санды 10го бөлгөндөгү калдыгын алуу керек (147 % 10 = 7).
2. Алынган калдыкты баштапкы мааниси нөлгө барабар болгон «суммага» (total = 0) кошобуз. Эми сумма 7ге барабар.
3. Андан соң биз бүтүн сандык бөлүү операторун колдонуп, сандын акыркы цифрасын «бөлүп салабыз» (147 // 10 = 14).
4. 14 != 0 болгондуктан биз циклдин башына кайрылабыз. Цикл n маанисси нөлгө барабар болгонго чейин уланат.
5. Мындан кийин кайрадан 4 цифрасын бөлүп салабыз (14 % 10 = 4) жана аны суммага кошобуз (4 +7 = 11).
6. Аны бардык сандан бөлүп алабыз (14 // 10 = 1).
7. Акыркы бир орундуу санды суммага кошобуз (11 + 1 = 12).

Жыйынтыгы: 12

Ушундай жол менен биздин программа төмөнкүдөй жазылат:

```
n = int(input('Санды киргизиңиз: '))
def digits_sum (n):
    total = 0
    while n != 0:
```

```
        total += n%10
        n = n // 10
    return total
#негизги программа
print (digits_sum(n))
```

Кээде программада бир эле жолу колдонулуп жана бир нече аргументи менен анча татаал эмес аракеттерди аткарган функцияны түзүү үчүн **lambda-функцияларды** колдонушат. lambda-функция бул анонимдик функция, б.а. **def** сыяктуу өзүнүн атына ээ эмес. Функциянын жазылышы **lambda** сөзүнөн башталат жана бош орундан кийин функциянын аргументтери көрсөтүлөт. Андан соң кош чекиттен кийин жыйынтыгы функцияда кайтарылган амалдар көрсөтүлөт. Мындай функцияны эки санды көбөйтүү мисалында карайлы:

```
multiple = lambda x, y: x * y #2 аргументи менен lambda-функция
print (multiple (2, 5)) #жыйынтык 10
```

Жогорку мисалда биз lambda-функцияны **multiple** өзгөрмөсүнө ыйгардык. Бирок, мындай функцияны бир сап менен эле өзгөрмөнү колдонбостон да жазсак болот:

```
print ((lambda x, y: x * y)(2, 6))
```

Ошентип, программалоодогу негизги көндүмдөрдүн бири – бул бир эле кодду бир нече жолу жазбоо болуп саналат. Качан код кайталана баштаса, бул кайталанган коддун бөлүгүн функцияга айлантуу керектигин түшүнүш керек.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *3 берилген санды өсүү тартибинде экранга чыгаруучу функцияны жазгыла.*
2) *Эки натуралдык сандын эң чоң жалпы бөлүүчүсүн табуучу функцияны жазгыла.*
3) *Эки натуралдык сан берилген. Алардын кайсынысында: а) цифралар көп экендигин; б) цифралардын суммасы чоң экендигин аныктоочу программа түзгүлө.*
4) *Эки үч бурчтуктун жактары берилген. Алардын: а) периметрлеринин; б) аянттарынын суммасын таап берүүчү программаны түзгүлө.*

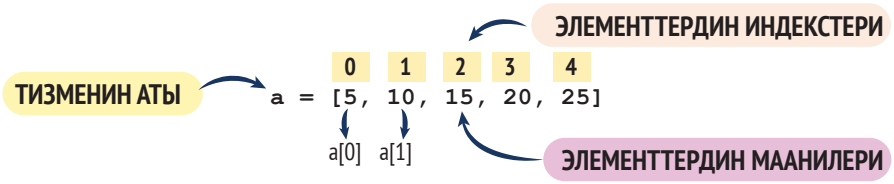
3.6-тема:

Массивдер

ОШОНДОЙ ЭЛЕ КАРА:
8-класс *3.2-тема*
Тизмелер, кортеждер, сөздүктөр

Азыркы компьютерлердин эң негизги кызматы – бул көп көлөмдөгү маалыматты иштетүү. Аларды иштетүүгө ыңгайлуу болуш үчүн **массивдер** деп аталган топторго бириктиришет. Массивдер бир өлчөмдүү (эгер элементте бир гана индекс болсо) жана эки өлчөмдүү (эки индекс) болушат. Эки өлчөмдүү массивдерди матрицалар деп да аташат. Алар менен иштөөнү биз 9-класста үйрөнөбүз.

Python тилиндеги бир **өлчөмдүү массивдер** – бул биз мурунку темаларда карап кеткен элементтердин тизмеси. Тизме өзүнүн атына ээ, ал эми тизменин ар бир элементи – өзүнүн индексине ээ. Мисалга «**a**» атындагы тизме берилсин:



Тизменин ар бир элементине мындай кайрылса болот: **a[0]** – «a тизмесиндеги 0 индексиндеги элемент». Мисалы:

```
print (a[0]) #жыйынтык 15
print (a[3]) #жыйынтык 20
```

Pythondo элементтерди аягынан баштап да санаса болот, анда индекстер терс сан менен көрсөтүлүп, -1ден башталат:

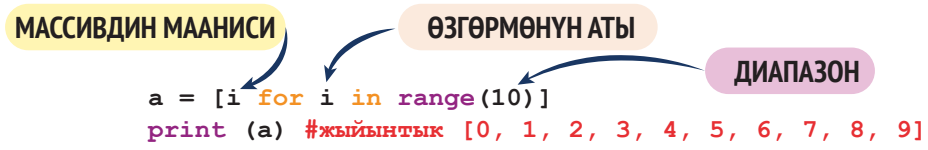
```
print (a[-1]) #жыйынтык 25
print (a[-3]) #жыйынтык 15
```

Тизменин узундугу (андагы элементтердин саны) **len()** функциясынын жардамы менен аныкталат:

```
d = len (a)  #жыйынтык 5
```

Тизмени бир нече ыкма менен түзсө болот: элементтерди чарчы кашаада үтүр менен ажыратып жазып, тизмени толтуруучу генератордун же кокустук сандардын генераторунун жардамы менен толтуруп ж.б.

Тизмени толтуруучу генераторлор жаңы тизмени циклди колдонуп элементтер менен толтурат. Мисалы, белгилүү бир санга чейин натуралдык сандар менен толтурулган жаңы тизмени түзүү керек дейли. Анда программа мындай болот:




```
a = [i for i in range(10)]
print (a) #жыйынтык [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Ушундай эле тизмени диапазону берилген **list()** функциясынын жардамы менен алса да болот:

```
a = list(range(10))
```

Тизмени ошол эле сандардын квадраттары менен толтуруу үчүн мындай генераторду колдонсо болот:

```
a = [i*i for i in range(10)] #[0,1,4,9,16,25,36,49,64,81]
```

Ал эми тизмени кокустук сандар менен толтуруп көрөлү. Ал үчүн бизге **random** модулунан **randint** (кокустук бүтүн сандардын генератору) функциясы керек:

```
from random import randint
a = [randint(1,100) for i in range(10)]
```

Коддон биз **a** тизмеси 1ден 100гө чейинки диапазондо 10 кокустук сан менен толтуруларын көрүп турабыз.

Берилиштерди тизмеге киргизүүнүн дагы бир ыкмасы - **append()** методун колдонуу. Ал үчүн **a[ ]** бош тизмеси түзүлөт, андан кийин ар бир жаңы киргизилген элемент тизменин аягына кошула берет. Алгач келечектеги **n** тизмесинин элементтеринин жалпы санын, андан кийин ар бир элементтин маанилерин бир-бирден өзүнчө сапчага сураган программаны жазалы. Төмөндө бул программанын мисалы көрсөтүлгөн:

```
a = []  #бош тизме
n = int(input())  #элементтердин санын беребиз
for i in range(n):
    a.append(int(input())) #кошуу үчүн элементтерди киргизебиз
print(a)
```

Үзүмдөр

Тизмелерди иштетүүдө анын белгилүү бир бөлүгү же үзүмү (англ. *slice* – үзүм) менен иштөө Pythondo көп колдонулат.

Үзүм мындайча жазылат: **[x:y]**. Мында **x** – үзүмдүн биринчи элементинин индекси, ал эми **y** – үзүм аяктаган элементтин индекси. Бирок үзүмгө **y** индексиндеги элемент кирбейт.

Үзүмдөрдү тизмеден бөлүп чыгаруунун жолу:

```
a = [1,2,3,4,5,6] #тизменин мисалы
b = a[1:3] #1ден 3-элементке чейин үзүм (3 кирбейт)
print (b)  #жыйынтык [2, 3]
```

Үзүмдү аныктоодо 3чү **z** саны да болушу мүмкүн. Ал кадамдын өлчөмү же б.а. элементтер үзүмгө кайсы кадам менен тандаларын көрсөтөт. Биздин тизменин мисалында ал мындай иштейт:

```
b = a[1:6:2] #жыйынтык [2, 4, 6]
```

Эгерде үзүмдүн биринчи санын көрсөтпөсөк (мисалы [:4]), анда программа элементтерди үзүмгө тизменин башынан баштап чыгарат. Эгерде экинчисин көрсөтпөсөк (мисалы [2:]), анда үзүм тизменин акыры элементине чейин уланат. Эгерде үзүмдө үчүнчү мааниси жок болсо, анда үзүмдө бардык элементтер ирээти менен тандалат. [:] үзүмү бардык тизмени толугу менен көчүрөт.

Үзүмдөрдү колдонуп, тизменин элементтерин тескери ирээтте да чыгарса болот:

```
a[::-1] #жыйынтык [6, 5, 4, 3, 2, 1]
```

Үзүмдөрдү белгилеп, бириктирип тизмени өзгөртсө болот. Төмөнкү мисалда башкы 2 элементтен турган үзүм жана 5-элементтен баштап аягына чейин алынган үзүм «+» операторунун жардамы менен биригишти:

```
b = [1, 2, 3, 4, 5, 6]
b = b[:2] + b[4:]
print (b) #жыйынтык [1, 2, 5, 6]
```

Берилген тизменин бир элементин гана эмес, үзүмдөгү бардык элементтерди өзгөртсө да болот:

```
a = ['p','y','t','h','o','n']
a [0:2] = [10,20]
print (a) #жыйынтык [10, 20, 't', 'h', 'o', 'n']
```

Элементтерди иргөө

Тизмени иштетүүдө андагы белгилүү элементтерди таап, алар менен кандайдыр бир аракеттерди жасоо көп колдонулат. Мисалы, тизменин эң максималдуу же минималдуу элементтерин табуу, белгилүү элементтерин керектүүгө алмаштыруу же өчүрүү. Ал үчүн программа тизмедеги ар бир элементти иргейт жана анын берилген шартка дал келишин текшерет.

Элементтерди иргөө үчүн төмөнкүдөй циклди колдонуу ыңгайлуу болот:

```
a = [1,2,3,4,5]
for i in a:
    print(i)
```

Бул жерде `print(i)` операторунун ордуна каалагандай команданы колдонсо болот. Мисалы, тизмедеги керектүү элементтердин санын эсептөөгө болот. Ал үчүн ар бир керектүү элементти табууда 1ге көбөйүп тургудай кылып эсептегич-өзгөрмөнү киргизүү керек.

Мисалы, **a** тизмесинде класстагы балдардын бою жөнүндө маалыматтар жазылган дейли. Бойлору 120 смден 150 смге чейин болгон окуучулардын санын табуу керек. **count** эсептегич-өзгөрмөсүн колдонуп программаны түзөлү:

```
count = 0
for i in a:
    if 120 < i < 150:
        count += 1
print (count) #көрсөтүлгөн бойдогу окуучулардын саны
```

Маселени татаалдаталы: тандалган окуучулардын бойлорунун орточосун табалы. Ал үчүн бардык табылган маанилердин суммасын эсептей турган **summa** деген жаңы өзгөрмөнү түзөлү. Андан кийин аны маанилердин жалпы санына бөлүп коёбуз. **summa** өзгөрмөсүнүн баштапкы мааниси да нөлгө барабар болушу керек:

```
count = 0
summa = 0
for i in a:
    if 120 < i < 150:
        count += 1
        summa += i
print(summa/count)
```

Тизменин элементтерин суммалоо үчүн Pythondo атайын `sum` функциясы каралган:

```
a = [1,2,3,4,5]
print (sum(a))
```

Бул функцияны колдонуп, мурунку маселени бир топ кыскартып жазса болот. Ал үчүн биз өзүнчө тизмеге керектүү элементтердин маанилерин чыгарып алалы. Андан кийин алардын суммасын жалпы санына (тизменин узундугуна) бөлөбүз.

Жаңы тизмени түзүү үчүн циклдеги шарттуу операторду колдонобуз:

```
b = [i for i in a if 120 < i < 150]
print (sum(b)/len(b))
```

Тандоо шарты **a** тизмесинен жаңы **b** тизмесине шартты канааттандырган гана элементтерди берди. Ал эми балдардын орточо боюн чыгаруу үчүн биз жаңы тизмедеги элементтердин суммасын алардын санына бөлүп койдук.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Тизмени толтуруу генераторун колдонуп, 1ден 30га чейинки сандардын тизмесин түзгүлө. Экранга* ***a*** *массивиндеги жуп жана так сандардын санын чыгаргыла.*
2) *Мурунку* ***a*** *тизмесинен экранга* ***b*** *тизмесине төмөнкүлөрдү чыгаруучу программа түзгүлө: а) 3кө бөлүнө турган бардык сандарды; б) так сандарды.*
3) *Тизменин эки кошуна элементинин (****a[0]*** *жана* ***a[1]****,* ***a[2]*** *жана* ***a[3]*** *...ж.б.) ордун алмаштыра турган программаны түзгүлө. Эгерде элементтердин саны так болсо, анда акыркы сан өз ордунда калышы керек.*
4) *Массивди 100 кокустук сан менен толтургула. Алынган тизмени экранга бир катарда 10 элементтен кылып чыгаргыла. Тизмени чыгаруу үчүн өзүнчө функцияны түзгүлө, аргумент катары ал тизмени кабыл алыш керек.*
5) *Дене тарбия сабагынан окуучулардын бойлорун тизмеге жазышты. Бул тизмеден бою эң узун жана эң кыска окуучуларды тапкыла.*

3.7-тема:

Саптар жана алар менен болгон амалдар

Адегенде компьютерлер эсептөөчү машиналар катары колдонулган, азыр болсо алардын негизги кызматы тексттик маалыматтарды иштетүү болуп бара жатат. Python тилинде текст менен иштөөчү негизги тип – бул саптар (**str** тиби) англ. ***string***.

Сап – бул бир же кош тырмакчанын ичине алынган символдордун удаалаштыгы: `'Бул сап'` = `"Бул сап"`

Тизмелерден (массивдерден) айырмаланып, саптар берилиштердин структурасына кирбейт. Ошону менен бирге саптарды иреттелген элементтердин удаалаштыгы катары карап, алар менен тизменин элементтериндей эле иштесе болот.

```
>>> s = 'Бул сап'
>>> s[0] #көрсөтүлгөн  индекси менен элементтерди кайтарат
'Б'
>>> s[5:] #5-индекстен акыркысына чейинки элементтер
'ап'
```

Бирок Python тилинде саптар өзгөртүлбөйт. Башкача айтканда, берилген саптын кайсы бир бөлүнгөн элементин башкасына алмаштырууга болбойт. Мындай учурда программа ката деп чыгарат. А бирок берилген саптын символдорунан керектүү өзгөртүүлөрдү киргизип **жаңы сапты** түзүүгө болот.

Сапты клавиатурадан киргизип, андагы бардык «а» тамгаларын «б» тамгаларына алмаштырып, экранга чыгаруунун толук программасын көрөлү:

```
s = input('Сапты киргизиңиз: ')
s1 = '' #өзгөртүүлөрү менен жаңы сап
for i in s:  #s сабындагы бардык символдорду иргеп чыгат
    if i == 'а': #өзгөрмөнүн мааниси «а»га дал келсе
        i = 'б' #анда аны «б» тамгасына алмаштырабыз
    s1 = s1 + i
print (s1)
```

Бирок бул ыкма өтө жай иштейт. Символдорду алмаштырууну жеңилдетүү үчүн Pythonдо даяр **replace** методу каралган.

Мындан тышкары көпчүлүк учурда тексттин сабын иштетүү керек болот: бул саптын бөлүгүн (сапча) алуу, эки сапты бир сапка бириктирүү, же саптын бөлүгүн өчүрүү. Мисалы, саптарды **бириктирүү** үчүн «+» оператору колдонулат. Бул амал конкатенация деп аталат:

```
s1 = 'Кутман'
s2 = 'күн'
s = s1 + ' ' + s2 + '!'
print (s)
>>>
Кутман күн!
```

ЭСИҢЕ ТУТ

Саптын узундугу саптын касиети **len** (*англ. length – узундук*) дин жардамы менен аныкталат. Ал үчүн **n** өзгөрмөсүнө **s** сабындагы символдордун саны жазылат, бош орун дагы символ болуп эсептелет:

n = len(s)

Саптарды иштетүү үчүн үзүмдөрдү колдонуу

Саптын белгилүү бир бөлүгүн бөлүп алып иштетүү үчүн тизмелер менен иштеген сыяктуу эле үзүмдөр колдонулат. Үзүмдөр мындайча жазылат: **[x:y]**. **x** – үзүмдүн башталышы, **y** – үзүмдүн аягы. **y** индексиндеги символ үзүмгө кирбейт. Алгач көрсөтүлбөсө, биринчи индекс 0гө, ал эми экинчиси – саптын узундугуна барабар.

Мисалы, s[3:8], s сабындагы 4-символдон 7-символго чейин (б.а. 8-ге чейин бирок 8 өзү кирбейт) үзүмдү түшүндүрөт.

Мааниси	s = 'китепканачы' сапчасы үчүн мисалдар

Саптын бөлүгүн (сапча) белгилеп алуу үчүн	`s1 = s[2:8]`
	`#s1 сабына 3-элементтен баштап 7-элементи кошулган маани жазылат`
	`>>> тепкан`

Саптын бөлүгүн өчүрүү үчүн	`s1 = s[:3] + s[9:]`
	`#башынан баштап 3-элементке чейин, жана 10-элементтен аягына чейин кесип алабыз жана аларды s1 сабында сактайбыз`
	`>>> китчы`

Саптын ичине жаңы фрагмент коюу	`s1 = s[:3] + 'ABC' + s[3:]`
	`#3-элементтен кийин ABCны кошуп кетебиз`
	`>>> китABCепканачы`

Сапты реверстөө (аны тескерисинче буруу)	`s1 = s[::-1] #сөзду тескери жазып чыгарат` `>>> ычанакпетик`

Берилген кадам аркылуу элементтерди тандоо	`s1 = s[::2]`
	`#ар бир экинчи символду кайтарат`
	`>>> ктпааы`

Саптар методу

Python тилинде саптар менен иштөө үчүн көптөгөн камтылган методдор бар. Алардын ичинен кызыктууларын карап көрөлү.

1 **upper** жана **lower** методдору сапты тиешелүү түрдө жогорку жана төмөнкү регистрлерге өткөрөт. **title** методу болсо биринчи эле тамгаларды жогорку регистрге, калганын төмөнкү регистрге өткөрөт:

```
s = 'aAbB cC'
s1 = s.upper() #'AABB CC'
s2 = s.lower() #'aabb cc'
s2 = s.title() #'Aabb Cc'
```

2 **split** методу сапты бош орундар боюнча бөлүүгө мүмкүндүк берет. Жыйынтыгында сөздөрдөн тизме алынат. Эгерде колдонуучу программада ар бири өзүнчө (тизмедегидей) иштетилсин деп, бир сапта катар сөздөрдү же сандарды киргизүү керек болсо, анда split методун колдонсо болот:

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.split () #['Дүйшөмбү','Шейшемби','Шаршемби','Бейшем-
би','Жума']
```

3 **join** методу тескерисинче тизмеден сапты курайт. Ал үчүн алдына сап-бөлгүч коюлат, ал эми кашаанын ичинде тизме берилет:

```
s = ['Дүйшөмбү', 'Шейшемби', 'Шаршемби', 'Бейшемби','Жума']
s1 = '-'.join (s) #Дүйшөмбү-Шейшемби-Шаршемби-Бейшемби-Жума
```

4 **find** методу саптын бөлүгү (сапча) менен иштейт. Ал саптагы сапчаны издейт жана табылган сапчанын биринчи элементинин индексин кайтарат. Эгерде сапча табылбаса анда -1ди кайтарат.

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.find ('Шаршемби') #жообу: 18, сапчанын 1-элементи - «Ш»
тамгасынын индекси
```

Мындан тышкары, бул метод менен берилген үзүмдөгү саптын элементинин индексин табууга болот. Үзүмдү көрсөтүү үчүн, анын башталышын жана аягын үтүр менен ажыратылган цифралар менен берүү керек. Эгерде экинчи цифра көрсөтүлбөсө, анда издөө саптын аягына чейин жүргүзүлөт:

```
s2 = s.find ('ү', 4) #8, 4-индекстен аягына чейинки бөлү-
гүндөгү биринчи «ү» нүн индекси.
```

```
s1 = s.find ('ү', 0, 9) #2, башынан 9-индекске чейинки бөлүгүндөгү биринчи «ү» нүн индекси
```

5 **replace** методу бир сапчаны башкасына алмаштырат. Бул учурда баштапкы сап өзгөрбөйт, болгону жаңы сапка модификацияланат (өзгөртүп түзүлөт), ал болсо жаңы s1 өзгөрмөсүнө ыйгарылат:

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.replace ('б', 'Б') #ДүйшөмБү ШейшемБи ШаршемБи БейшемБи Жума
```

Кээде бизге бөлүнгөн бөлүгү жаңы саптан башталгыдай кылып, сапты бөлүш керек болот. Бул учурда биз \n белгисин колдонобуз.

```
print ('Дүйшөмбү \n Шейшемби \n Шаршемби \n Бейшемби \n Жума') #бардык сөздөр мамыча түрүндө чыгат.
```

Эгерде жаңы сапты жылдыруу менен чыгаруу керек болсо, анда \t белгисин колдонобуз.

```
print ('Дүйшөмбү \n\t Шейшемби \n\t Шаршемби \n\t Бейшемби \n\t Жума') #бардык сөздөр мамыча түрүндө чыгат, ар бир кийинки саптын алдына орун таштalат.
```

Жогорудагы үйрөнгөн командаларды колдонуп, сапты иштетүүнүн мисалын карап көрөлү.

1-маселе. Клавиатурадан фамилиясын, атын жана атасынын атын камтыган сап киргизилет, мисалы:

Айтматов Чыңгыз Төрөкулович

Ар бир эки сөз бири-биринен бош орундар менен ажыратылган, саптын башында бош орун жок. Бул сапты иштетүүнүн натыйжасында фамилия жана инициалдарын эле камтыган жаңы сап пайда болуш керек:

Айтматов Ч.Т.

Чыгаруу:

1 Сапты клавиатурадан киргизебиз:

```
s = input('Фамилия, атыңыз жана атаңыздын атын киргизиңиз: ')
```

2 Киргизилген сапты бош орун менен ажыратылган өзүнчө сөздөргө бөлүп чыгабыз. Ал үчүн **split** методун колдонобуз. Бул массивде үч элемент болот: **fio[0]** – фамилиясы, **fio[1]** – аты, **fio[2]** – атасынын аты:

```
fio = s.split()
```

3 Инициалдары менен фамилияны чогултабыз:

```
fioshort = fio[0]+' '+fio[1][0]+'.'+ fio[2][0]+ '.'
```

Толук программа мындай көрүнөт:

```
s = input('Фамилия, атыңыз жана атаңыздын атын киргизиңиз: ')
fio = s.split ()
fioshort = fio[0] + ' ' + fio[1][0] + '.' + fio[2][0] + '.'
print (fioshort)
```

Саптарды салыштыруу жана сорттоо

Биз алфавит боюнча сорттоону сөздү тезирээк табуу үчүн колдонобуз. Эгерде сөздүктөрдө сөздөр алфавит менен жайгашпаганда, анда биз бир сөздү издөө үчүн эле бир топ убакыт коротмокпуз. Python тилинде саптарды сорттоо кандай принцип менен түзүлгөн?

Көрсө, сандар сыяктуу эле тамгалар да өзүнүн салмагына ээ экен. Алфавитте биринчи турган тамга жеңилирээк, б.а. «а» «б» га караганда жеңил, ошондуктан сорттоодо ал биринчи чыгат. Мындан саптарды да сандар сыяктуу эле салыштырууга болот деген жыйынтык келип чыгат.

Сөздөрдү салыштырууда, алгач биринчи тамгалары салыштырылат, эгерде алар айырмаланса, анда салыштыруунун жыйынтыгы аныкталат. Андан кийинки тамгалар салыштырылбай калат. Ал эми эгерде биринчи тамгалары барабар болсо, анда кийинки 2 элементи салыштырылат жана ушул сыяктуу аягына чейин кетет. Мисалы, «банкет» сөзү «банкир» сөзүнө караганда жеңил: алар 5-тамгада айырмаланышат жана «е» < «и».

Эгерде силерде текшерүү үчүн символдор бүтүп калса, анда кыска сап узун сапка караганда кичине, ошондуктан «банк» < «банкир». Символдор таблицасында баш тамгалар кичине тамгадан биринчи турушат, ошондуктан, кичине коддорго ээ: «А» < «а», «Б» < «б». Бирок компьютер «алфавиттик тартипти» кайдан «билет»? Көрсө, саптарды салыштырууда символдордун ASCII жана Unicode коддору колдонулат экен.

«БАНК» жана «Банк» деген сөздөрдү салыштыралы. Биринчи символ эки сөздө тең бирдей, ал эми экинчиси айырмаланат – биринчи сөздө баш тамга, ал эми экинчисинде ошол эле тамга, бирок кичине. Ошондуктан:

«БАНК» < «Банк» < «банк»

Ал эми башка символдор (цифралар, латын тамгалары) менен кандай болот? Коддук таблицада цифралар ирети менен жайгашкан жана латын

тамгаларынан мурун турат; латын тамгалары – орус тамгаларынан мурун, орус жана латын тамгаларынын баш тамгалары тиешелүү тилдердин кичине тамгаларынан мурун турат. Демек:

«5BANK» < «BANK» < «Bank» < «bank» < «БАНК» < «Банк» < «банк»

Мисалы, сөздүн ичиндеги тамгаларды сорттоо үчүн, программаны мындай жазуу керек:

```
s = 'Дүйшөмбү'
s1 = ' '.join (sorted (s))
print (s1)
>>> Д б й м ш ү ү ө
```

1-маселе. Клавиатурадан бир нече сөздөрдү (мисалы фамилияларды) киргизүү жана аларды экранга алфавиттик тартипте чыгаруу керек.

Бул маселени чыгаруу үчүн, үтүр менен ажыратылып киргизилген фамилияларды тизмеге кайра жазып алуу ыңгайлуу, андан соң sorted методу менен сорттоп коюу керек:

```
s = input('Фамилияларды киргизиңиз: ') #Абакиров Муканова
Бебинов
s1 = s.split() #сапчалар менен тизме түзөт ['Абакиров', 'Му-
канова', 'Бебинов']
s2 = ' '.join (sorted (s1))
print (s2)
>>> Абакиров Бебинов Муканова
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) Өзүңөрдүн атыңарды киргизгиле: а) экранга 1чи жана 3-символду чыгаргыла; б) силердин атыңардын 1чи жана акыркы символдору дал келерин аныктагыла.
2) Клавиатурадан бир нече сөз киргизүүнү жана ошол сөздөрдүн ичинен эң кыска сөздүн узундугун аныктоочу программа түзгүлө.
3) isdigit() саптык методу сап жалаң гана цифралардан турарын текшерип берет. Киргизүүдө программа эки бүтүн санды сураган жана алардын суммасын чыгарып берүүчү программаны түзгүлө. Туура эмес киргизүү учурунда программа ката деп токтоп калбастан, кайрадан эле сандарды сурап тургандай кылуу керек.

3.8-тема:

Саптарды форматтоо

Python тилинде саптарды форматтоо шаблондун кайсы бир ордуна башка текстти коюу дегенди түшүндүрөт. Ордуна коюу ошол замат жүргүзүлөт. Мисалы, форматтоону колдонуу менен даяр шаблон аркылуу мындай чакыруу билеттерин жасоого болот:

Урматтуу Алина!
Сизди Ачык эшиктер күнүнө чакырабыз.
Датасы: 1-май
Урматтоом менен Тимур.

Бир жолу ордуна коюу мындай жазылат:

```
print ('Урматтуу {}!'.format ('Алина'))
```

Башкача айтканда, шаблондун текстинен кийин бош фигуралуу кашаалар коюлат, андан кийин **.format ()** командандын кашаасынын ичине коюу керек болгон маанилери көрсөтүлөт. Программа ишке киргенде, ордуна коюлуучу текст фигуралуу кашаалардын ордуна коюлат.

Бир нече коюулар болсо, фигуралуу кашаага сөздөрдүн индекстери, ал эми сөздөрдүн өзү **.format**тан кийин кортежде коюлат. Ал шаблон мындай жазылат:

```
print ('Урматтуу {0}! \n Сизди {1} чакырабыз.\n Датасы:
{2} \n Урматтоом менен {3}.'.format ('Алина', 'Ачык эшиктер
күнүнө', '1-май', 'Тимур'))
```

Текстти **.format**ты колдонбостон башка ыкма менен да форматтаса болот. Бул ыкма бир аз туура эмес жана эскирип калган деп эсептелет. Бирок, эгер силер коддон % белгисин кезиктирсеңер, анда бул жерде форматтоо колдонулду деп эсептесеңер болот. Биздин чакыруу шаблонубузду % операторунун жардамы менен жазып көрөлү:

```
print ('Урматтуу %s! \n Сизди %s чакырабыз.\n Датасы: %s \n
Урматтоом менен %s.' % ('Алина', 'Ачык эшиктер күнүнө',
'1-май', 'Тимур'))
>>>
Урматтуу Алина!
Сизди Ачык эшиктер күнүнө чакырабыз.
Датасы: 1-май
Урматтоом менен Тимур.
```

Силер байкадыңарбы, кээ бир жерде биз %d деп, кээ бир жерде %s деп жаздык? Алар ордуна эмне коюларын аныктайт:

%s – сапты коёт;
%d – бүтүн санды коёт;
%f – бөлчөк санды коёт.

Санды сапка жана сапты санга өзгөртүп түзүү

Практикалык маселелерде көбүнчө символдордун катары түрүндө жазылган сандарды сандык мааниге жана тескерисинче айландырууга туура келет. Бул үчүн Python тилинде атайын стандарттык функциялар бар:

int – сапты бүтүн санга айландырат

```
s = '123'
N = int ( s ) #N = 123
```

float – сапты чыныгы санга (бөлчөк) айландырат

```
s = '123.456'
X = float ( s ) #X = 123.456
```

str – бүтүн жана чыныгы сандарды сапка айлантат

```
N = 123
s = str ( N ) #s = '123'
X = 123.456
s = str ( X ) #s = '123.456'
```

Эгерде сапты санга айландыра албай калса (мисалы, сапта тамгалар камтылса), анда ката пайда болот да программа аяктайт.

Бөлчөк сандар үчүн (float тиби), бөлчөк бөлүгүнөн канча белгини чыгарууну көрсөтүп койсок болот, мисалы:

```
X = 34.8589578
print('{:.2f}'.format(X))   #34.86
print('{:.3f}'.format(X))   #34.859
```

Эгерде чоң сандарда үтүрдү биз разряддарды бөлүү үчүн колдонгубуз келсе, анда мындай жазабыз:

```
print('{:,.2f}'.format(10001.23554))   #10,001.24
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Колдонуучудан логин жана паролду сураган шаблонду жазгыла. Эгерде логин же пароль туура эмес болуп калса, «мындай логин жана пароль табылган жок» деген билдирүүнү чыгарсын. Ал эми логин жана пароль туура болсо, анда колдонуучунун атын көрсөткөн саламдашууну чыгаргыла.

3.9-тема:

Python тилинде графика менен иштөө

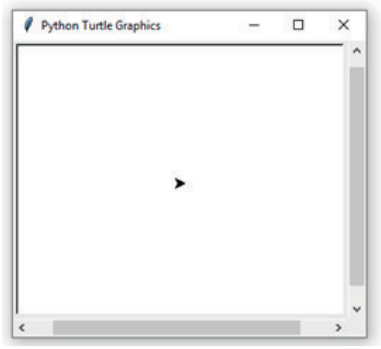
Turtle модулунун жардамы менен сүрөт тартуу

Python программалоо чөйрөсүндө жөнөкөй сүрөт тартуу үчүн **Turtle** (ташбака) модулу колдонулат, ал төмөнкү команда менен кошулат:

```
from turtle import* #ташбака менен иштөөчү командаларды жүктөйт
reset () #позицияны нөлгө айлантат жана калемди күйгүзөт
```

Программаны ишке киргизүү менен силер борборунда калем (ташбака) жайгашкан графика үчүн терезени көрө аласыңар. Мисалы, **forward** командасы ташбаканы алдыга жылдырат. Кашаанын ичинде пиксель менен кадамдардын саны көрсөтүлөт. **right** жана **left** бурулуу командалары үчүн кашаанын ичинде бурууга керек болгон градустардын саны көрсөтүлөт. Эми квадратты тартып көрөлү:

```
from turtle import*
reset ()
forward (100) #алдыга карай 100 пик-
селге кыймылда
right (90)  #90˚ка оңго бурул
forward (100)
right (90)
forward (100)
right (90)
forward (100)
```



Кодду кыскартуу үчүн **for** циклин жана фигурага дагы кошумча параметрлерди берели:

```
from turtle import*
width (5)      #сызыктын жоондугу - 5 пиксель
color('blue','green') #1-түс – сызыктын, 2-түс – боёктун түсү
begin_fill() #аймакты толтуруу үчүн ташбакага байкоо жүргүзүү
for i in range (4):
    forward (100)
    right (90)
end_fill() #begin_fill()ден баштап, аймакты түс менен толтуруу
>>>
```

up() командасын колдонуп, калемди көтөрүп, башка орундан баштап сүрөттү тартып (же текстти жазып) баштасаңар болот. Башка сүрөттү тартуудан мурун кайра калемди **down()** командасы менен түшүрүү керек.

Айлананы тартуу үчүн **circle(r,n)** командасы колдонулат, мында r – айлананын радиусу, n – биз тарткан айлананын бөлүгү, градус менен. Эгер n = 180 градус болсо, анда калем жарым айлана сызат, n = 360 градус болсо, толук айлана сызат.

Чекитти тартуу үчүн **dot(r, color)** командасы колдонулат, мында r – чекиттин радиусу (пиксель менен), color – чекиттин түсү.

1-маселе. Айлана чийели, анын узундугу боюнча берилген сандагы чекиттер бирдей бөлүштүрүлүп жайгашсын:

```
from turtle import*
def circ(d, r, rBig): #параметрлери менен circ функциясы: чекиттердин саны, чекиттин радиусу, айлананын радиусу
    for i in range(d):
        circle(rBig, 360 / d)#чекиттерди санын айлана боюнча бөлүштүрөбүз
        dot(r, 'red')
up()
goto(150, 0) #калемди 150 пикселге оңго жылдырабыз
setheading(90) #калемди 90° ка бурабыз
down()      #калемди сүрөт тартууну баштоо үчүн түшүрөбүз
circ(15, 10, 150) #параметрлерге маанилерин беребиз
screen.mainloop() #программанын аткарылышын токтотобуз
```

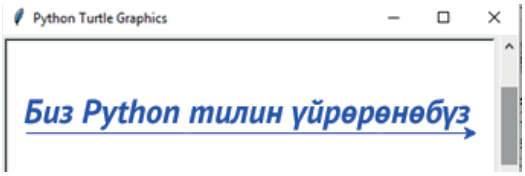
Фигураларды чыгаруудан башка графикалык терезеде текстти да тартса болот. Ал үчүн параметрлери менен **write()** командасы колдонулат:

```
write(text, move, align, font = (fontname, fontsize, fontstyle))
```

- **text** параметрине тырмакчага алынып текст өзү жазылат;
- **align** параметри «left», «right», «center» маанилерин алат жана текстгин абалын ташбакага салыштырмалуу өзгөртөт, маанилери тырмакчада жазылат;
- **font** параметри fontname, fontsize, fontstyle маанилерин алат:
  - **fontname**ге тырмакчада шрифттин аты жазылат;
  - **fontsize** шрифттин өлчөмү үчүн жооп берет;
  - **fontstyle** текстгин стилине жооп берет (**normal** – кадимки, **bold** – кара, **italic** – курсивдүү, **bold italic** – кара, курсивдүү текст).

2-маселе. Форматталган текстти чыгаруучу программа түзөлү:

```
from turtle import*
color ('blue')
write('Биз Python тилин үйрөнөбүз', 'center', font=('Roboto', 24, 'bold
italic'))
>>>
```



clear() жана **reset()** командалары сүрөттөрдөн экранды тазалайт жана калемди борборго жайгаштырат.

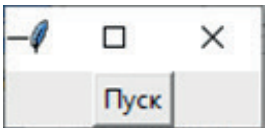
Графикалык объекттерди түзүү үчүн Tkinter менен иштөө

Tk графикалык китепканасы менен иштөө үчүн Tkinter модулу жогорку деңгээлдеги графика жана анимация менен иштөөгө мүмкүндүк берет. Turtle модулу сыяктуу эле tkinter модулун да кошуп алабыз:

```
from tkinter import*.
```

3-маселе. Алгач кадимки эле баскычты (кнопка) түзүп көрөлү. Ал үчүн Button виджетин колдонобуз, кашаанын ичинде баскычтын параметрлерин беребиз. text маанисинин жардамы менен баскычтын атын жазабыз. Төмөнкү программаны иштетели:

```
from tkinter import*
tk = Tk() #tk ат менен Tk
объектисин түзөбүз
btn = Button(tk, text='Пуск')
#«Пуск» тексти менен баскыч
түзөбүз
btn.pack() #баскычты терезе-
нин ичинде жайгаштырабыз
>>>
```



Бирок бул баскычты баскандан эч нерсе чыкпайт. Баскычты басуу кандайдыр бир аракет менен коштолсун үчүн программага жыйынтыгы **command** командасы аркылуу чыга тургандай функция киргизели:

БУЛ КЫЗЫКТУУ!

Виджеттер – бул кээ бирлери бардык программалоо тилдеринде стандарттык болгон, программанын графикалык интерфейсин түзүү үчүн атайын базалык блоктор. Мисалы, бул кнопкалардын, желекчелердин же жылдырып көрүү тилкелеринин виджеттери. Алар аттары менен гана айырмаланышы мүмкүн: мисалы, классикалык желекчелер (check box) Tkinterде check button деп аталышат.

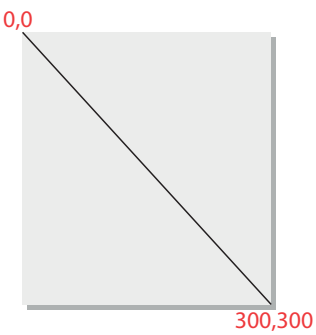
```
from tkinter import*
def btn_act(): #консолдо жыйынтык чыгара турган функция
    print('Оюн башталды!')
tk = Tk()
btn = Button (tk, text='Пуск', command=btn_act) #баскычка
басканда функциядагы билдирүү чыгарылат
btn.pack()
```

Эми баскычка баскан сайын консолдо ушул маалымат чыгып турат:

```
>>> «Оюн башталды!»
```

Башка виджет – Canvas (англ. холст) берилген аянтта сүрөт тартууга мүмкүндүк берет. Сүрөт тартуу холсттун өлчөмдөрүн: холсттун туурасын (width) жана бийиктигин (height) берүү менен башталат.

Холсттогу сүрөттүн баштапкы чекитин белгилөө үчүн X, Y координаталары колдонулат. Координаттар горизонталь боюнча (X) сол четинен, ал эми вертикаль боюнча (Y) жогорку четинен канча пикселге жылышкандыгын аныктайт.



Сызыкты түзүү үчүн **create_line()** методу колдонулат, кашаанын ичинде 4 сан көрсөтүлөт. Биринчи экөөсү сызыктын башталыш координаталары, кийинки экөөсү – акыркы координаталары болот. Төмөнкү программаны жазалы:

```
from tkinter import *
tk = Tk()
canvas = Canvas(tk, width=300, height=300)
canvas.pack()
canvas.create_line(0, 0, 300, 300)
```

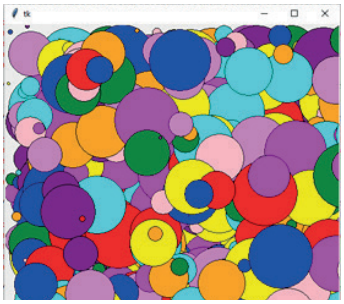
Айлана сызуу үчүн **create_oval()** методу колдонулат:

```
canvas.create_oval(10, 10, 80, 80, outline= 'red', fill=
'green', width=2)
```

Берилген жазууда биринчи 4 параметри фигураны чектөө координаталарын аныктайт. Б.а., бул ичинде айлана сызыла турган квадраттын жогорку сол бурчтун жана төмөнкү оң жак бурчтарынын x жана y координаталары.

4-маселе. Холстто диаметрлери жана түстөрү кокустук сан менен алынган тегеректерди сызуучу программаны түзөлү.

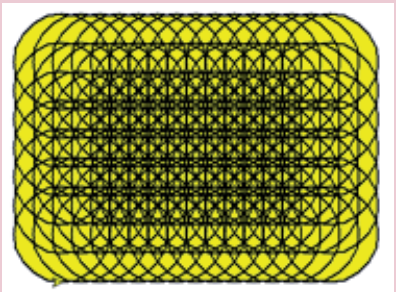
```
from random import* #random модулунан randint жана choice функцияларын жүктөйбүз
from tkinter import*
size = 500 #size өзгөрмөсүн киргизебиз
tk = Tk()
diapason = 0 #тегеректердин санын чектөө үчүн diapason өзгөрмөсүн киргизебиз
my_canvas = Canvas (tk, width=size, height=size) #size өзгөрмөсүнүн маанисин колдонуп холст түзөбүз
my_canvas.pack()#холстту терезенин ичине жайгаштырабыз
while diapason <1000: #цикл ушул шартка чейин кайталанат
    color = choice(['green', 'red', 'blue', 'orange', 'yellow', 'pink', 'purple', 'violet', 'magenta', 'cyan'])
#тегеректердин түстөрүн кокустан тандоо үчүн тизме түзөбүз
    x1 = randint(0,size) #x, y коорд-ды кокустан тандоо
    y1 = randint(0,size)
    d=randint(0,size/5) #тегеректердин диаметрлерин каалагандай тандоо, бирок size/5 тен чоң эмес
    my_canvas.create_oval(x1,y1,x1+d,y1+d,fill=color)#тегеректерди түзөбүз жана кокустан тандалган түс менен ичин боёйбуз
    tk.update()
    diapason+=1 #циклдин кадамы, эсептегич
```



КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) Turtle модулунун жардамында айланалардан килем жасагыла. Мында айланалар бир түс менен, ал эми килемдин фону башка түс менен боёлсун.

2) Tkinter модулунун жардамында холстко жоондугу жана түсү кокустан тандалган сызыктарды чыгаруучу программаны түзгүлө.



4 - бөлүм

Компьютердик тармактар жана интернет

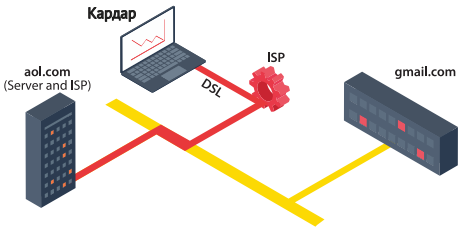
4.1-тема:

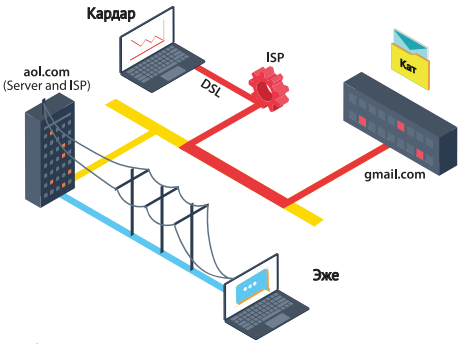
Компьютердик тармактар

Компьютердик тармак – бул маалыматты берүү каналдары менен байланышкан компьютерлердин системасы. Компьютерлер өз ара атайын өткөргүчтөр менен (Ethernet кабелдери, була оптикалык кабелдер) же аларсыз деле (Bluetooth, Wi-Fi ж.б.) байланышы мүмкүн. Компьютердик тармактар локалдуу (бир имараттын чегинде) же глобалдуу (интернет) болушу мүмкүн.

Интернет глобалдуу тармагы кантип иштерин карап көрөлү.

Мисалы, биз **aol.com** сайтына кирип, акыркы жаңылыктарды окугубуз келип жатат дейли.





Алгач биз провайдердин тармагына туш болобуз, андан кийин провайдер бизди интернет тармагы аркылуу тиешелүү сайтка багыттайт. Ошондон кийин гана биз **aol.com** сайтынын баракчарын көрө алабыз.

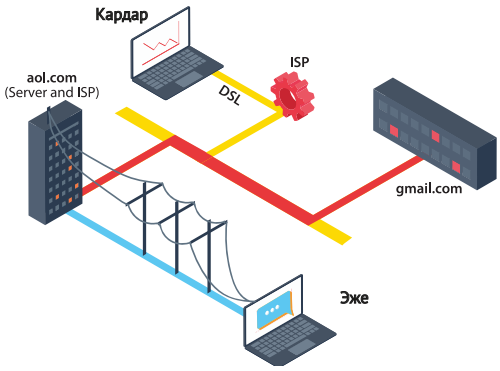
Башка мисал: эжеме биз электрондук катты жөнөткүбүз келип жатат. Эжемдин интернети телефондук байланыш кабели боюнча **aol.com** провайдерине кошулган. Эжемдин почтасы **aol.com** сайтында жайгашкан. А биздин – **gmail.com**до. Алгач **gmail.com** сайтында өзүбүздүн почтабызга киришибиз керек (мисалы, *my@gmail.com*).

БУЛ КЫЗЫКТУУ!

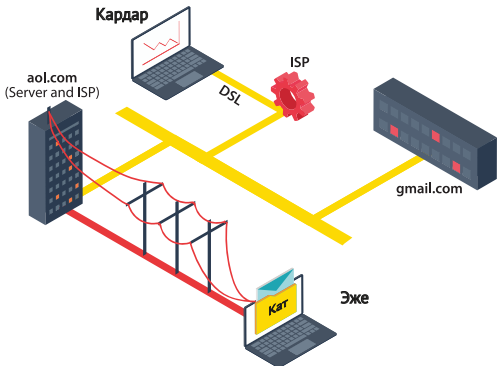
IP–даректерди домендик аттарга өзгөртүп түзүү үчүн атайын **DNS** (*домендик аттардын Сервери, Domain Name Server*) серверлер колдонулат.

Бул серверлер домендик аттардын IP–даректер менен дал келген атайын таблицаларын сактайт. Сурам боюнча алар домендик аттарды IP-даректерге, кайра IP-даректерди домендик атка өзгөртүп түзөт. Домендик аттарды биз браузерге веб-баракчарды табуу үчүн киргизебиз. Алар **URL** (*Universal Resource Locator*) деп да аталышат.

Андан соң катты жазып, эжемдин дарегин киргизебиз (sister@aol.com) жана «жөнөт» баскычын басабыз.



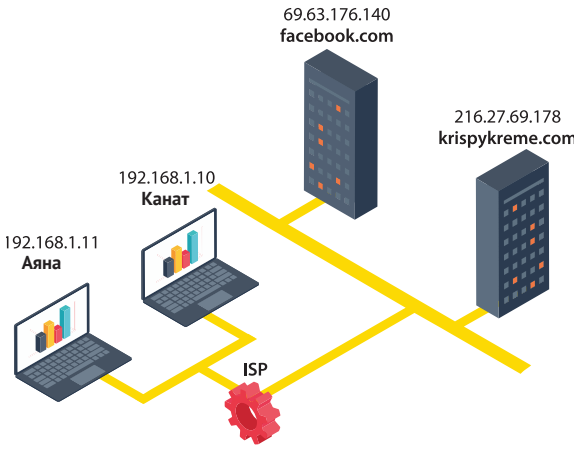
Gmail.com aol.com серверине катты жөнөтөт.



Андан кийин эжем аны өзүнүн компьютерине жүктөп алат.

Интернетте билдирүүнү жөнөтүүдө компьютер аны **пакеттер** деп аталган кичинекей бөлүктөргө бөлүп чыгат. Акыркы серверде билдирүү кайрадан керектүү тартипте чогулуп калат. Билдирүү бир эле электрондук кат түрүндө болбостон, веб-баракча, Твиттердеги билдирүү же каалагандай башка маалымат болушу мүмкүн.

Мисалга, үйдөгү интернетти сиз жана сиздин ата-энеңиз да колдонуп жатасыңар. Мындай учурда силер Wi-Fi аркылуу үй маршрутизаторго туташкан болосуздар.

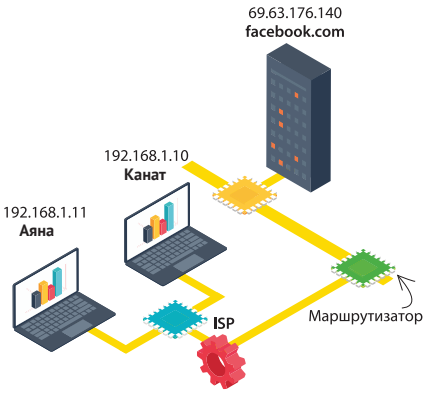


Эгерде сүрөткө карасак, анда сиздин пакеттер жана ата-энеңиздин пакеттери интернетке бир чекит аркылуу кетип жаткандын көрүүгө болот. Башкача айтканда, сиздин ата-энеңизге тиешелүү маалымат жана сиздин досторуңуз менен болгон маалымат алмашуу да, бардыгы бир орунда чогулат.

Интернеттен сизге тиешелүү каттар сиздин ата-энеңизге эмес өзүңүзгө гана келиш үчүн эмне кылуу керек?

Ушул учурда бизге IP-даректер жана маршрутизаторлор жардамга келет. Интернет тармагында бардык колдонуучулар автоматтык түрдө IP-дарек алышат. Компьютерлер, уюлдук телефондор, планшеттер жана интернетте бири-бирине маалымат жөнөтүп жаткан түзүлүштөр өзүнүн IP-даректерине ээ болот. Интернет тармагынын түрдүү бөлүктөрүн бириктирген түзүлүштөр маршрутизаторлор деп аталат.

Маршрутизаторлор интернетте пакеттерди жөнөтөт. Ар бир пакет жакынкы маршрутизаторго барат. Мисалы сайтка кирүүдө силердин пакетиңер адатта 10дон 20га чейинки маршрутизаторлордон өтөт.



Сүрөттө сигналдын маршрутизаторлор арасындагы кыймылы берилген.

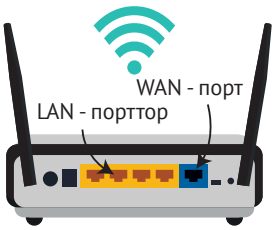
Тармакка берилген пакеттер бир нече катмарлардан турат: 1-катмар – компьютериңиздин IP-дареги; 2-катмар – маршрутизатордун IP-дареги; андан кийин пакет өткөн бардык маршрутизаторлордун IP-даректери жазылат.

ЭСИҢЕ ТУТ

- **IP-дарек** – бул интернет тармагына кошулган ар бир түзүлүшкө ыйгарылган атайын номер.
- **DNS** (***Domain Name Service***) – бул сайттын аттарын IP – дарекке которуучу сервис.
- **URL** (***Universal Resource Locator***) – бул сайттын универсалдуу көрсөткүчү, эстеп калууга жеңил дарек (мисалы, www.code.org)/
- **Интернет** (***Internet***) – бул бири-бири менен жалпы тармак аркылуу биригишкен компьютерлердин жана серверлердин группасы.
- **Серверлер** (***Servers***) – бул берилиштерди жөнөтүп жана сурамдарга жооп берип турган кубаттуу компьютерлер.
- **Була оптикалык кабелдер** (***Fiber optic cable***) – бул маалыматты берүү үчүн жарык колдонулган атайын кабелдер.
- **Wi-Fi** –бул берүү радиотолкундарды колдонуп санариптик сигналды берүү ыкмасы.
- **DSL** – бул телефондук же телевизиондук кабелди колдонуп маалыматты берүү ыкмасы.
- **Пакеттер** (***Packets***) – бул сигналды өтө тыкандык менен берүү үчүн чоң маалымат пакеттеринен түзүлгөн маалыматтын кичинекей үлүштөрү.

Локалдык тармакты жана Wi-Fi тармагын тескөө (орнотуу)

Локалдык тармактар бир имараттын чегинде бири-биринен өтө алыс эмес жайгашкан компьютерлердин ортосунда маалымат алмашууну ишке ашырат.



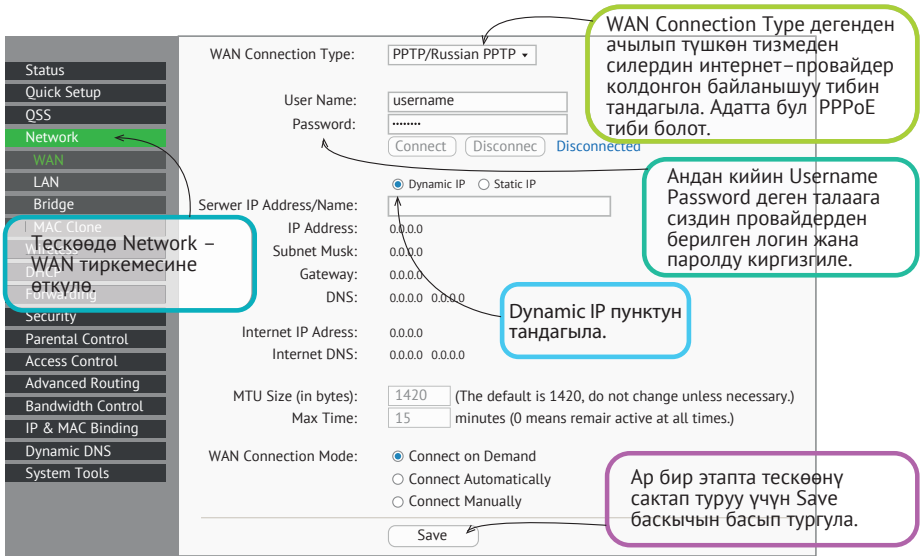
Үйдөгү (локалдык) Wi-Fi тармагын орнотуп көрөлү. Ал үчүн интернет-провайдерге туташылган роутер керек. Аны кабель же Wi-Fi аркылуу тескөөгө болот.

Кабель аркылуу кошууда: кабель бир учу менен компьютерге кошулат, 2- учу менен – роутердин LAN сайгычына кошулат. Андан кийин роутердин WAN сайгычына силердин интернет провайдердин кабелин кошуу керек.

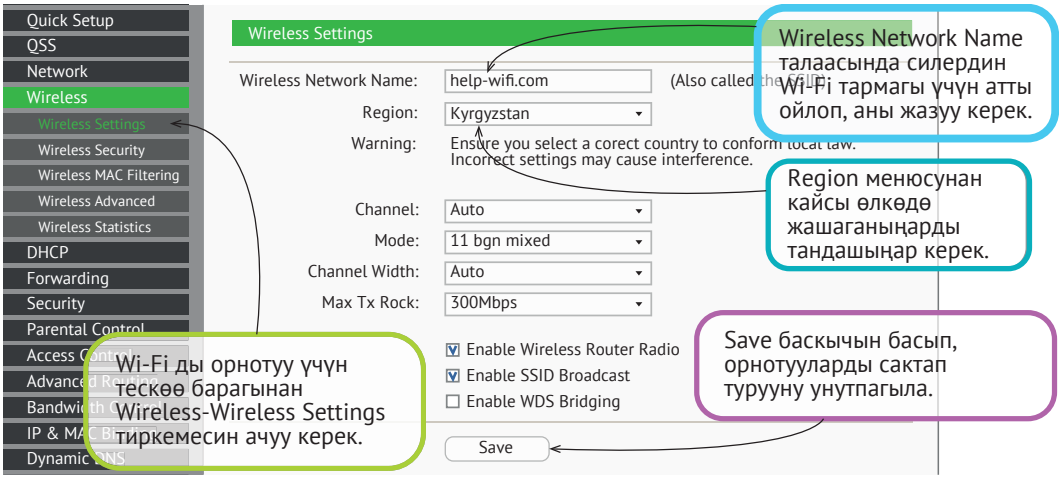
Роутерди Wi-Fi аркылуу орнотууда роутерди күйгүзгөндөн кийин (адатта тармактын аты сиздин роутердин маркасындай болот) пайда болгон Wi-Fi тармагына компьютерди кошуп коюу жетиштүү болот. Бул тармак ачык болот. Бул тармакка планшет же телефон аркылуу кошулуп компьютерсиз эле маршрутизаторду орнотсо да болот. Бул учурда роутер өзү WAN–кабели аркылуу провайдердин тармагына туташкан болуш керек.

Андан ары силер браузерден 192.168.0.1. же 192.168.1.1. даректери аркылуу роутердин тескөөсүнө киришиңер керек. Мында колдонуучунун атын жана паролун сураган терезе чыгат. Ага роутердин алдында чапталган кагазда көрсөтүлгөн логин жана паролду киргизесиңер.

Роутердин тескөө терезеси ачылат.



Wi-Fiды тескөө үчүн Wireless – Wireless Settingsти ачуу керек.



Wi-Fi тармагын пароль менен коргоо үчүн Wireless - Wireless Security тиркемесине өтүңүз. Ал жерден коопсуздуктун тибин WPA/WPA2 – Personal тандоо керек. PSK Password талаасында силердин Wi-Fi тармакты коргогон паролду ойлоп таап жазгыла.

Save баскычын басып, орнотууларды сактап турууну унутпагыла.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Качан силер аны браузерде ачканда силердин пакетиңер google.com серверине жеткенге чейинки өткөн бардык серверлерди жазып чыккыла.*

2) *Роутер кошулганда силердин провайдериңер кандай DNS – серверлерди беререн жазып чыккыла.*

3) *Алмаз кагазга IP-дарегин жазып алып, кийин кокустан ал баракты майдалап айрып алды. Жыйынтыгында IP-даректердин фрагменттери менен 4 айрынды алынды. Бул айрындылар А, Б, В жана Г тамгалары менен белгиленди. IP-даректи калыбына келтирип көргүлө. Жообунда IP-даректин иретине туура келгендей кылып айрындыларды белгилеген тамгалардын удаалаштыгын көрсөткүлө.*

А.	Б.	В.	Г.
.64	3.13	3.133	20

4.2-тема:

Интернет протоколдорунун түрлөрү

Интернет тармагындагы протоколдор жана тармакта иштөө

Бир компьютерди экинчисине жөнөкөй туташтыруу – бул тармакты түзүү үчүн зарыл болгон кадам. Бирок бир эле бул кадам аздык кылат. Маалыматты бере башташ үчүн, компьютерлер бири-бирин «түшүнөрүнө», алар өз ара «маалымат алмаша ала турганына» ынаныш керек. Компьютерлер тармакта кантип «маалымат алмашышат»? Мына ушул мүмкүнчүлүктү камсыз кылуу үчүн «протоколдор» деп аталган атайын каражаттар иштелип чыккан.

АНЫКТАМА

Протокол – бул тармак аркылуу маалыматты берүүнү жүргүзгөн тиешелүү эрежелер.

Протокол түшүнүгү бир эле компьютердик индустрияга гана тиешелүү эмес. Ал эмес интернет менен эч качан тааныш болбогондор да протоколдорду колдонуп иштеген түзүлүштөр менен кезигишет. Мисалы, кадимки телефондук канал да өзүнүн протоколуна ээ болот. Ал аппараттарга байланыш каналынын экинчи учундагы абоненттин жооп бергендиги же сигналдын үзүлгөндүгү жөнүндө фактыны тактайт. Телефондук трубкадагы үн сигналы – бул телефондордун тили.

Компьютерлердин да бирдиктүү тили бар, бул – **протокол**.

Интернет ишинде колдонулуучу негизги протоколдор:

- ***TCP/IP*** – пакеттерди берүү;
- ***POP3/SMTP*** – электрондук каттарды алуу/берүү;
- ***FTP*** – файлдарды берүү;
- ***HTTP/HTTPS*** – веб-баракчаларды берүү;
- ***IMAP4*** - электрондук каттарды алуу/берүү.

TCP/IP протоколу

IP протоколу TCP/IP протоколдорунун негизин түзөт. Ал туташтырууну орнотпогон протоколдордун тибине кирет. IP протоколу билдирүүлөрдү жеткирүүнүн ишенимдүүлүгүнө кепилдик бербейт, анткени маалыматты жеткирүү тастыкталбайт.

Берилиштердин агымын IP протоколу белгилүү бөлүктөргө – **IP-пакеттерге** же **дейтаграммаларга** бөлөт жана ар бир IP-пакетти көз карандысыз пакет катары карайт. IP протоколунун негизги кызматы тармактар арасында IP-пакеттерди берүү болуп саналат.

IP протокол билдирүүлөрдү ишенимдүү жеткирүүгө кепилдик бербегендиктен, бул маселени чечүү үчүн TCP протоколун колдонушат. TCP протоколу компьютерлер арасында логикалык байланышты орнотот. Маалыматтарды берүү алдында сеансты баштоого сурам жөнөтүлөт. Кабыл алуучу тарабынан тастыктоо жөнөтүлөт, эгер белгилүү убакытта тастыктоо болбосо, анда сурам кайрадан жөнөтүлөт.

TCP протоколу берилиштердин агымын **сегменттерге** бөлөт. Ар бир сегменттин баш сөзүндө контролдук суммасы берилет.

- Эгерде жөнөтүүдө берилиштер бузулган болсо TCP протоколу ошол контролдук сумма аркылуу муну аныктай алат. Бузулган сегмент жок кылынат, ал эми булакка эч нерсе жөнөтүлбөйт.

- Эгерде берилиштер бузулбаса, анда алар тиркемеде чогултулат, ал эми булакка тастыктоо жөнөтүлөт. Сегменттерди транспорттоо үчүн TCP/ IP протоколу аларды IP-пакетинин кабыгына жайгаштырат.

HTTP протоколу

HTTP протоколу ***(Hypertext Transfer Protocol – Гипертекстти берүү протоколу)*** интернеттен веб-барактарды натыйжалуу берүү үчүн иштелип чыккан. HTTP негизинде тармактагы барактарды алардын укмуштуудай кооздугу менен көрүүгө мүмкүнчүлүк түзүлдү. HTTP протоколу World Wide Web системасынын негизи болуп саналат.

Силер браузердин интерфейсин колдонуп, HTTP командаларын бересиңер. Электрондук чычкандын баскычы менен шилтемеге басууда, браузер веб-серверден ошол шилтеме көрсөткөн ресурстун берилиштерин – мисалы, веб-барактын маалыматтарын сурайт.

ЭСИҢЕ ТУТ

Интернет тармагынын протоколдору деңгээлдерге бөлүнөт, бир протоколдордун иштеши үчүн башка протоколдор керек болот. Мисалы, HTTPS/HTTP, FTP, SMTP, POP3, IMAP протоколдору компьютерлер арасында маалыматтарды берүү үчүн TCP/IP протоколун колдонушат.

Веб-баракты мазмуну браузерде ойдогудай чагылдырылышы үчүн, ал өзгөчө тексттик белгилердин – гипертекстти белгилөө тили (HyperText Markup Language, HTML) менен берилет.

Мындан тышкары, учурда **HTTP-S (HTTP Secure)** протоколу – гипертекстти кошумча коргоо менен берүүгө колдонулат. Бул учурда веб-сервер жана колдонуучунун ортосундагы бардык берилип жаткан маалыматтар шифрленет. Бул болсо колдонуучуга интернетте маанилүү маалыматты (кредиттик карталардын аттарын, паролдорун, номерлерин ж.б.) коопсуз алмашууга жардам берет.

HTTP протоколу боюнча силер кайрылган интернет ресурстарынын даректери болжолдуу түрдө төмөнкүдөй болот: http://www.code.org.

FTP протоколу

FTP (File Transfer Protocol – Файлдарды берүү протоколу) – файлдар менен алмашууга мүмкүндүк берет. FTP протоколу боюнча файлдарды берүү **FTP-сервер** жана **FTP-кардардын** ортосунда жүрөт. FTP-сервер FTP-кардардан келген сурамдарды – файлдарды жүктөө үчүн программаларды иштетет.

FTP-кардарга файлды берүү процессинин башында серверге колдонуучунун атын жана паролду жөнөтүү талап кылынат. Ал колдонуучуну идентификациялоого мүмкүндүк берет. FTP-серверди колдонуучу идентификациясыз эле киргендей кылып орнотуп койсо да болот. Бул учурда колдонуучу **анонимдүү** (anonymous) деп аталат, башкача айтканда, серверге колдонуучунун бир эле атын (anonymous) билдирип турат. Пароль мындай учурда эске алынбайт.

Эми биз электрондук почталардын иштөөсүнө зарыл болгон SMTP, POP3 жана IMAP протоколдорун карап көрөлү. Бул протоколдор бир гана максатта – электрондук билдирүүлөр менен алмашууну уюштуруу үчүн колдонулат. Мисалы, билдирүүнү жөнөткөн протокол аларды кайра кабыл ала албайт жана тескерисинче. Ошол себептен мындай протоколдор түгөйү менен иштешет.

SMTP протоколу

SMTP (Simple Mail Transfer Protocol – Жөнөкөй почталык берүү протоколу) электрондук билдирүүлөрдү жөнөтүүнү камсыз кылат.

Бирок SMTP сервер билдирүүлөрдү кабыл алуучу тарапта чогултууга жөндөмдүү эмес. Ошондуктан, почтаны кабыл алууда дагы бир почталык протокол - POP3 же IMAP протоколу керек.

POP3 протоколу

POP3 (Post Office Protocol 3 – Почталык кызмат протоколу) адресат тарабынан электрондук билдирүүнү кабыл алууну камсыз кылат. Бул протоколдун негизинде почта сервер аркылуу кабыл алынат жана ал жерде топтолот. Почталык кардар программасы почтаны мезгил менен текшерип турат жана билдирүүлөрдү локалдык компьютерге жүктөйт.

Ошентип, почтаны жөнөтүү SMTP менен ишке ашат, ал эми кабыл алуу – POP3 жардамы менен жүрөт. Мына ошондуктан почталык кардарды орнотууда SMTP серверинин аты менен бирге POP3 серверинин атын да киргизүү талап кылынат.

IMAP протоколу

IMAP (Internet Message Access Protocol – интернеттин электрондук почтасына кирүүгө мүмкүндүк алуу протоколу) POP3кө окшош, келген каттар менен иштөө үчүн кызмат кылат, андан тышкары кошумча функцияны камсыз кылат. Тактап айтканда, почтаны локалдык компьютерге сактабастан туруп, ачкычтык сөз менен издөөгө мүмкүндүк берет. Бул протоколду колдонгон почталык программа сервердеги каттарды кабыл алуучунун компьютерине жүктөбөстөн эле иштетүүгө мүмкүндүк берет. Демек каттар, алардын толук мазмундагы файлын серверден улам жөнөтүп жана кабыл алып турбастан эле, колдонуучунун (кардардын) компьютеринде жеткиликтүү болот. Каттарды жөнөтүү үчүн SMTP протоколу колдонулат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Силер электрондук почтаңарда катты ачып окуганда жана жөнөткөндө браузер кандай протоколдорду колдонот?*
2) *Силер колдонуп жүргөн сайттардын ичинен кайсылары HTTPS протоколун колдобойт?*

4.3-тема:

Стилдердин каскаддык таблицалары (CSS)

Кадимки HTML документ форматтоо тегдеринин жардамы менен текстин түсүн жана өлчөмдөрүн орнотууга мүмкүндүк берет. Эгерде сайттагы бир типтүү элементтердин параметрлерин өзгөртүш керек болуп калса, анда тегдерди табыш жана өзгөртүү үчүн бардык баракталарды карап чыгуу керек болот. Анын ордуна текстин түсүн, өлчөмдөрүн ж.б параметрлерин стилдерде сактай турган башка – CSS технологиясын колдонсо жеңил болот.

Стиль деп HTML документтин элементтерине, анын сырткы келбетин тез өзгөртүү үчүн колдонулган форматтоо эрежелеринин жыйындысын айтышат. Стилдер бир аракет менен эле форматтоону окшош элементтердин бүткүл группасына колдонгонго мүмкүндүк берет, мисалы, бардык баш сөздөрдүн көрүнүшүн өзгөртүү ж.б.

Стилдер жөнөкөй HTMLге караганда форматтоо үчүн көп мүмкүнчүлүктөрдү берет, андан тышкары өзүнчө сырткы файлда сактала алат. Браузер мындай документтерди колдонуучунун компьютеринде локалдуу сактайт (кэштейт), ошондуктан сайттын жүктөлүүсү тез болот.

HTML документтеги элементтерди форматтоонун бардык жеткиликтүү касиеттери 9 категорияга бөлүнөт:

- **Шрифт** – шрифттердин типографиялык касиеттерин орнотот;
- **Түс жана фон** – тексттин түсүн жана фонду аныктайт, андан тышкары сүрөттү фон катары колдонот;
- **Текст** – текстти түздөө, форматтоону аныктайт;
- **Блок** – блоктуу элементтерди форматтоо касиети;
- **Визуалдык форматтоо** – элементтерди чагылдыруу блоктору, аларды позициялоо жана тизмелерди чагылдыруу менен байланышкан касиеттери;
- **Фильтрлер жана өтүүлөр** – мультимедиялык эффекттерди жана графикалык сүрөттөрдү өзгөртүп түзүүнү аныктайт;
- **Басып чыгаруу (печать)** – баракты бөлүнүшүнүн спецификацияларын аныктайт;
- **Псевдокласстар** – @import, cursor, important касиеттери ишке киргизилет;
- **Калган башка касиеттери.**

CSSтин синтаксиси

CSS стили ар дайым стилди жарыялоо блогунун сол бөлүгүнөн орун алып, чарчы кашаага алынган селектордон (HTML документтин элементинен) турат.

Берилген мисалда H1 тегине жасалгалоонун жаңы стили колдонулду, эми ал көк түстө жана анын шрифти Verdana болду.

Селектор – бул берилген стилдер кайсы элементтерге таандык экендигин аныктоочу конструкция.

Селекторлордун 4 тиби эң көп колдонулат:

- Элементи боюнча;
- Классы боюнча;
- ID боюнча;
- Контексттик селектор.

Элементи боюнча селектор

Эгерде стилди биринчи деңгээлдеги бардык баш сөздөргө колдонуш керек болсо, анда селектор катары <H1> элементи колдонулат. Мисалы, биринчи деңгээлдеги бардык баш сөздөр күңүрт-боз түстө, Arial шрифти менен жана тамгалардын арасындагы аралык 10 пикселден болуп чыгыш үчүн стиль мындай жазылат:

```
h1{color:#666;font-family:Arial;letter-spacing:10px}
```

Класс боюнча селектор

Белгилүү тегге конкреттүү касиеттерди колдонуш үчүн каалаган элементте пайдаланууга боло турган class деген атайын атрибут бар.

Мисалы, менюдагы шилтемелерди өзгөчө кылуу үчүн жаңы классты киргизебиз: class=«menu».

```
a.menu {font-style: italic;font-weight: bold}
```

HTML код болсо мындай болот:

```
<a href="about.html" class="menu">Мектеп жөнүндө</a> (Өзгөчө шилтеме)
<a href="links.html">Шилтемелер</a> (Кадимки шилтеме)
```

ID боюнча селектор (идентификатор боюнча)

Эгерде класс боюнча селектордо элементтин аты менен классты бөлүү үчүн чекит колдонулса, ID боюнча селектор үчүн # белгиси колдонулат. Элементтин атын калтырып кетсе да болот, анда берилген стилди first идентификаторундагы бардык элементтерге колдоно берсе болот.

```
a#first {font-style: italic;font-weight: bold}
<a href="about.html" id="first">Мектеп жөнүндө</a>
```

Контексттик селектор

Контексттик селекторлор башка элементтин ичинде жайгашкан элементтердин группасы үчүн бирдиктүү стилди колдонууга мүмкүндүк берет. Мисалы, таблица – бул элемент, ал эми бардык мамычалар – бул элементтер группасы.

Берилген мисалда div теги менен бөлүнгөн бардык группадагы шилтемелер үчүн

```
<div id="first">
 <a href="index.html">Негизги барак</a>
 <a href="about.html">Мектеп жөнүндө </a>
</div>
```

first идентификатору менен стиль колдонулат.

```
a#first {font-style: italic;font-weight: bold}
```

Стилдерди документке кошуу

Бардыгы болуп стилдерди кошуунун төрт ыкмасы бар:

- Стилдерди HTML документтин ичине камтуу;
- HTML документтен сырткы Стилдер документине шилтеме берүү;
- Стилдер документин HTML документине импорттоо;
- Стилдерди түздөн-түз HTML документинин саптарына кошуу.

Стилдерди HTML документтин ичине камтуу

CSS менен байланышкан бардык маалымат HTML документтин башында жайгашат жана <BODY> тегинин ички мазмуну менен байланыш болбойт.

Мисал:

```
<html>
 <style type="text/css">
  h1{color:green;font-family:Impact}
  p{background:yellow;font-family:'Courier New'}
 </style>
  <head>
  <title>Камтылган стилдер</title>
  </head>
 <body>
  <h1> Мисал 1</h1>
  <p>Мисал 2</p>
 </body>
</html>
```

Камтылган стилдер ушул гана HTML документке колдонулат. Эгер силер Стилдерди жалгыз барак үчүн колдоном десеңер, анда бул ыкма эң ыңгайлуу болуп саналат.

Стилдерди импорттоо

Импорттолгон стилдер өзүнчө файлда жайгашкан болот. Аларды документтин ичинде жайгашкан өзүңөрдүн стилиңер менен бириктирсеңер да болот.

Мисал:

```
<html>
 <style type="text/css">
  <!--
       @import url(mytyles.css)
   h1{color:green;font-family:Impact}
  -->
 </style>
 <head>
  <title>Камтылган стилдер</title>
  </head>
 <body>
```

```
  <h1>Мисал 1</h1>
  <p>Мисал 2</p>
 </body>
</html>
```

Mystyles.css файлы мындай көрүнөт:

```
h1{color:orange;font-family:Impact}
p{background:yellow;font-family:'Courier New'}
```

Бул мисалда браузер биринчи Mystyles.css (@import сапчасы ар дайым биринчи болуш керек) файлынан стилдерди импорттойт, андан кийин буга стилдердин камтылган командаларын кошот. Веб-бараkта тигил да, бул да стилдер колдонулат.

Стилдерди түздөн-түз HTML документинин тулкусуна кошуу

Бул учурда стилди HTML документтин баш сөзүнө кошуунун кереги жок. Элементтин стилинин орнотулган атрибуттары браузерде чагылдыруу үчүн бүткүл маалыматты өзүндө камтыйт. Бул стиль берилген элемент үчүн гана колдонулат.

Мисал:

```
<html>
 <head>
  <title>Камтылган стилдер</title>
 </head>
 <body>
  <h1 style="color:green;font-family:Impact"> Мисал 1</h1>
 <p style="background:yellow;font-family:'Courier New'">
Мисал 2</p>
 </body>
</html>
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

«Менин хоббим» деген веб-баракты түзгүлө жана баш сөз, шилтемелер үчүн аныкталган өзүңөрдүн CSS стилиңерди импорттогула.


</CODE>
PYTHON
void setup() { inMode (13, OUTPUT);
// жылтылдоолорун пиндерди чыгууга орнотолу
pinMode (12, OUTPUT); pinMode (11, OUTPUT); pinMod

9 - класс

1 - бөлүм

Информатика жана маалымат

1.1-тема:

Маалыматтык сабаттуулук

Бүгүнкү күндө, маалыматтын агымы күндөн-күнгө көбөйүп жатканда, керектүү маалыматты тандоо жана баалоо ар бир адам үчүн эң маанилүү көндүм болуп тургандыгы шексиз. Бул көндүм маалыматтык сабаттуулук деп аталат жана коюлган маселени чечүү үчүн компьютердик тиркемелерди (программаларды) колдоно алууну түшүндүргөн компьютердик сабаттуулуктан айырмаланат.

Маалыматтык сабаттуулук төмөнкүлөргө мүмкүндүк берет:

- конкреттүү маалымат муктаждыктарды аныктаганга;
- маалымат булактарын табууга;
- маалыматты кабыл алууга же тандоого;
- маалыматтын сапатын баалоого жана талдоого;
- маалыматты сактоого жана иреттөөгө;
- маалыматты натыйжалуу жана этика менен колдонууга.

АНЫКТАМА

Маалыматтык сабаттуулук – бул адамдын маалыматка муктаж экендигин аңдап-түшүнүү, аны издеп, тандап, баалап жана пайдалана билүү жөндөмү.

Маалыматты издөөдө аны сын көз менен баалоо жөндөмү маанилүү ролду ойнойт. Интернеттен маалыматты алып жатып, өзүңөргө беш суроо бергиле:

- Макаланын автору ким, ал жазып жаткан тема боюнча эксперт болуп саналабы?
- Макаланын мазмуну жалпы сайттын тематикасына дал келеби?
- Сайт ким тарабынан түзүлгөнү жөнүндө маалымат барбы?
- Макаланы жарыялоо датасы барбы?
- Эмне үчүн мен бул маалыматка ишенүү керек деп эсептейм?

Өзгөчө азыркы кездеги «жаңы медиа» деп аталган интерактивдик массалык маалымат каражаттарынан (социалдык тармактар, жаңылыктар каналы ж.б.) алган маалыматты туура баалай билүү керек.

Мындай медиалардын негизги максаты болушунча көп колдонуучулардын көңүлүн бурдуруу жана аудиториянын көз карашына олуттуу таасир этүү болуп саналат.

Ал үчүн ар кандай ыкмалар колдонулат: олуттуу материалдарды кызыктыруучу оюн формасында берүү, же болбосо көпчүлүк учурда чындыкка коошпогон, үрөй учурган контентти камтыган материалдарды жарыялоо. Андан тышкары, маалыматты көп учурда өздөрүнүн кызыкчылыгы үчүн өзгөртүп колдонушат (манипуляциялашат).

МААЛЫМАТ МЕНЕН ТӨМӨНКҮЛӨРДҮ КЫЛСА БОЛОТ

- Жок нерсени ойлоп таап, аны чындык катары көрсөтүүгө
- Толук эмес бир тараптуу берүү менен бурмалоого
- Өзүңдүн оюңду жана комментарийлерди кошуп редакциялоого
- Манипулятор үчүн пайдалуу болгондой интерпретациялоого
- Кайсы бир маанилүү бөлүктөрүн жашырып, айтпай коюуга

Маалыматты бурмалоо окурманда туура эмес, жалган ой-пикирди калыптантат.

Социалдык тармактарда каалаган адам маалымат каналы болгондуктан, көптөгөн фейктер пайда болуп жатат.

АНЫКТАМА

Фэйк (*англ.* ***fake*** *– «жасалма, жалган»*) маалыматты берүүдө болсо – “жалган маалымат” деген маанини түшүндүрөт.

Бүгүнкү күндө эмнелерди “фейк” деп аташат:

- Фотошопто өзгөртүлүп жасалган сүрөттөр, адаштыруу максатында же башка окуяны иллюстрациялоо үчүн атайын монтаждалган видеороликтер. Мисалы, бир нече жыл мурун аскердик операцияларда тартылган видеону акыркы күндөрдө болгон окуя катары көрсөтүшөт.
- Көпчүлүк аны чындыктан айырмалай албаган жалган жаңылыктар (мурда “гезиттик өрдөк” деп аталгандарды азыр “төгүндүлөр”(вбросы) деп аташат).
- Социалдык тармактардагы башка адамдын (көбүнчө белгилүү адамдын) аты менен түзүлгөн баракчар.

Фейктерди жаратуунун максаты болуп көпчүлүк учурда алдамчылык саналат. Ал акча каражатын чогултуу (мисалы, «смартфондорду ойнотуу»), же жалган пикирлерди жарыялоо менен белгилүү товарларды сатууну көбөйтүү болушу да мүмкүн. Андан тышкары, фейктерди жарыялоо менен алдамчылар көп сандагы көрүүнү (трафикти генерациялоо) ишке ашыра

алышат. Мисалы, абдан таасирдүү баш сөздөрдүн жардамы менен: «Сенсация: англис тилин 1 айдын ичинде үйрөн!», «Ал 2 аптанын ичинде 20 кг га арыктады, сенин колуңан да келет!» ж.б.

Спам

«Маалыматтык таштандынын» тараган дагы бир түрү болуп спам эсептелет. Спам – бул көпчүлүк учурда жөнөтүүчүнү тактоого мүмкүн болбогон сиз каалабаган жарнамалык жөнөтмөлөр. Мындай билдирүүлөр электрондук почта, социалдык тармактар аркылуу, форумдар, комментарийлер жана СМС - билдирүүлөр аркылуу таралат. Статистикага таянсак, азыркы кездеги бардык электрондук каттардын 90%ы – бул спам. Башкача айтканда, бир пайдалуу электрондук катка спамы менен тогуз кат туура келет. Спамерлер (спамды жөнөткөндөр) өздөрүнүн спамдары үчүн кабыл алуучулардын даректерин ар түрдүү сайттардан чогултушат. Качандыр бир убакта силер ал сайттарга өзүңөрдүн маалыматыңарды (логин, эл.почта ж.б.) киргизгенсиңер.

Негизи спамдар силерди кайсы бир шилтемелер менен өтүүңөрдү же шек жараткан тиркемелерди ачууга багыттайт. Албетте, спамды өчүрүү эң жеңил болгону менен кээде капыстан спамдагы шилтемени басып алуу олуттуу көйгөйдү пайда кылуусу мүмкүн: вирус жуктуруп алуу же сиздин өздүк маалыматтарыңардын уурдалышына жол берүү.

Интернет-алдамчыларга алданып калбаш үчүн сын көз менен ой жүгүртүү зарыл, башкача айтканда, ар кандай шек жараткан маалыматтын алгачкы булагын табууга аракеттенүү керек.

Эң негизгиси – интернетте айтылгандардын бардыгына эле ишене бербөө.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *«Фейк» деген эмне? Окуу жылынын башталышын 15-сентябрга жылдырат деген маалымат «фейк» болуп эсептелеби? Эмне үчүн?*
2) *Социалдык тармактагы «фейктер» менен сыналгыдагы «фейктер» бири-биринен айырмаланабы? Алар эмнеси боюнча айырмаланышы мүмкүн экендигин түшүндүргүлө.*
3) *Өзүңөрдүн мини-изилдөөңөрдү жүргүзгүлө: «Фейк» окуяңарды 10 досуңарга айтып, алардын канчасы силерге ишенгенин эсептегиле.*
4) *Таблица түзгүлө: спамдын кандай түрлөрү болот жана алардын ар бири кандай зыян алып келиши мүмкүн?*

1.2-тема:

Шифрлөө жана электрондук-санариптик кол тамга

Санариптик кылымда электрондук документтер өзгөчө мааниге ээ болуп келүүдө.

Электрондук документ – бул электрондук алып жүрүүчүдө сакталган документ. Каалагандай юридикалык маанидеги келишимдер же маалымкаттар сыяктуу кагаз документтердей эле электрондук документтерге да кол койсо болот. Ал үчүн электрондук-санариптик кол тамга колдонулат.

Электрондук документтер кагазга караганда бир топ артыкчылыктарга ээ:

- Электрондук документтерди иштетүү жана жеткирүү убактысынын тездиги (каалагандай документке аралыктан эле кол коюп, аны электрондук почта менен жөнөтүү).
- Чыгашалардын азайышы: документти кагазга чыгаруунун, почта менен жөнөтүүгө төлөөнүн ж.б. кереги жок.

Башкача айтканда, мурда кандайдыр-бир укуктарды же акчаны берүүдө, мисалы, үйдү алып-сатуу же болбосо мамлекеттик мекемеден маалымкат алуу, ошол документ берген органда маалымкат алуучу адамдын сөзсүз болуусу менен гана ишке ашкан. Андан тышкары, бизге калем сап менен толгон-токой кагаздарга кол коюп чыгууга туура келген. Операторлор адамдын аты-жөнүн тактоо үчүн паспортунун же туулгандыгы тууралуу күбөлүгүнүн түп нускасын көрсөтүүнү өтүнүшкөн.

Санариптик кылымда көптөгөн ушул сыяктуу аракеттерди интернет аркылуу аралыктан жүргүзүүгө болот. Негизги шарт – оператор сизден келген маалыматты аутентификациялап (текшерүүдөн өткөрүп) туруусу керек. Бул үчүн ар кандай технологияларды колдонушат, мисалы, банкоматтан акча алуу үчүн 4 орундуу санды киргизүү керек, ал болсо сиздин картаңызга кирүүгө мүмкүнчүлүк түзөт, ал эми сиздин каттарыңызды окуу үчүн электрондук почтага паролуңузду киргизүү керек болот.

АНЫКТАМА

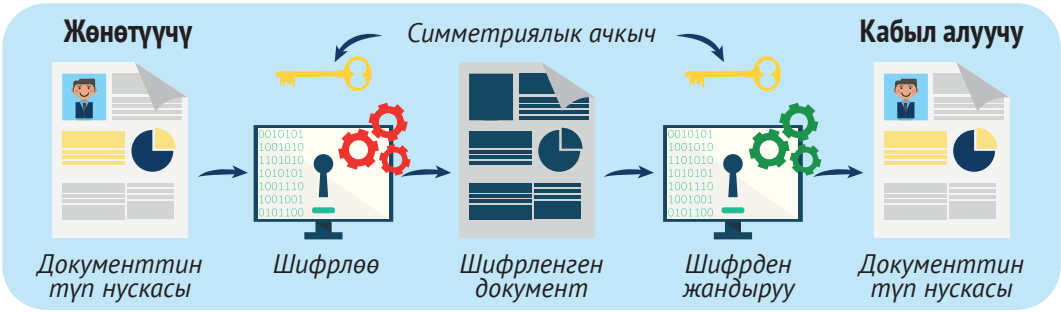
Аутентификация – бул бир нерсенин чындык экенин текшерүү процесси. **Аутентификациянын** мисалы катары маалыматтар базасынын серверинде сакталган пароль менен колдонуучу киргизген паролду салыштыруу болуп саналат.

Бирок, мамлекет катышкан электрондук документтерге, мисалы, келишимге, арызга кол коюу үчүн **электрондук-санариптик кол тамганын (ЭСКТ)** болуусу шарт. ЭСКТ – бул калем сап менен коюлуучу кол тамга сыяктуу эле уникалдуу кол тамга, болгону электрондук файлдар үчүн гана колдонулат. ЭСКТ берилген электрондук документке ким кол койгонун жана документ кол коюлгандан кийин өзгөргөнүн же өзгөрбөгөнүн аныктоого мүмкүндүк берет.

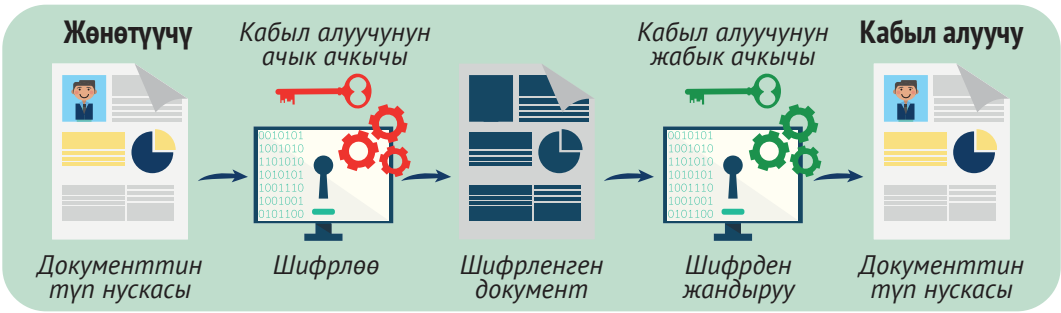
Башка сөз менен айтканда ЭСКТ – бул документти шифрлөө механизми болуп саналат.

Тарыхта шифрлөөнүн негизги эки түрү колдонулган:

- Симметриялык – бир гана ачкыч колдонулат, башкача айтканда, бир эле ачкыч менен документ шифрленет жана шифри чечмеленет.



- Асимметриялык – эки ачкыч колдонулат: бир ачкыч менен документ шифрленет, экинчиси менен шифри чечмеленет.



Асимметриялык шифрлөөдө, эгерде бирөө сизге шифрленген билдирүүнү жөнөткүсү келсе, ал сиздин ачык ачкычыңызды (сиз аны бардык каалоочуларга бересиз) алып, билдирүүнү шифрлеп, анан сизге жөнөтөт.

Ал эми сиз болсо, ал билдирүүнү алгандан кийин өзүңүздүн жабык ачкычыңызды колдонуп (ал сизге эле белгилүү) шифрленген билдирүүнү чечмелеп аласыз.

Ошентип, кол тамганы текшерүү аны кайрадан эсептөө жана текшерүүчү тараптагы ачкыч менен салыштыруу болуп эсептелет. Текшерүүдө кол тамга же туура, же туура эмес деген билдирүү чыгарылат. Эгерде кол тамга туура эмес деп чыкса, анда туура эмес ачкыч колдонулду же документ өзгөрүүгө дуушар болгон.

Акырында айтаарыбыз, өзүнүн корголушунун негизинде асимметриялык шифрлөө эң кеңири таралган. Аны **HTTPS** протоколун колдогон сайттар, мессенджерлер (эстегилечи, Whatsappта «сиздин билдирүүлөр өтмө шифрлөө менен корголгон» деп жазылып турат), wi-fi-роутерлер, банк системалары жана башкалар колдонуп келишет. **Электрондук кол тамга** да асимметриялык криптографияга негизделген. Андан тышкары, асимметриялык криптографиянын негизинде **блокчейн** алгоритми түзүлгөн, өз кезегинде бардык криптовалюталар, анын ичинде **биткоин** да анын негизинде түзүлгөн.

БУЛ КЫЗЫКТУУ!

Блокчейн (англ. ***blockchain***) – маалыматты камтыган жана эреже менен түзүлгөн блоктордун үзгүлтүксүз удаалаш чынжыры. Көпчүлүк учурда чынжыр блокторунун нускалары бири-биринен көз карандысыз көптөгөн ар башка компьютерлерде сакталат.

Башка сөз менен айтканда блокчейн – бул берилиштер сакталган жай жалпы процессор менен байланышпаган бөлүштүрүлгөн маалыматтар базасы. Чынжырдагы бардык колдонуучулар базадагы маалымат менен башкалар эмне кылып жаткандыгын көрө алышат, бирок алар өздөрүнө тиешелүү гана маалымат блокторун өзгөртө алышат. Бул бардык процесстерди ачык-айкын кылат, андыктан бул базада кандайдыр бир документти өзгөртүү мүмкүн эмес.

Блокчейн технологиясынын артынан эч бир дүйнөлүк банкка же кайсы бир өлкөнүн экономикасына көз каранды болбогон **биткоин** жаңы криптовалютасы пайда болду. Анын өзүнүн баасы (курсу) бар, аны интернетте сатып алса жана сатса болот. Биткоиндин эң негизги өзгөчөлүгү – бул анын анонимдүүлүгү.

СУРООЛОР ЖАН ТАПШЫРМАЛАР:

1) *Симметриялык жана асимметриялык шифрлөөлөрдүн негизги артыкчылыктарын атап бергиле.*
2) *Өзүңөрдүн жашооңордон мисал келтиргиле, симметриялык жана асимметриялык аутентификациялоо кай жерлерде колдонулат?*

1.3-тема:

Графикалык маалыматты коддоо

Сактоо формасы боюнча графикалык маалыматтарды ***аналогдук*** (*кагаздагы сүрөт*) жана **санариптик** (*компьютердик графика*) деп бөлүшөт.

Компьютердик графика – бул информатиканын бир бөлүмү, ал эсептөө техникасынын жардамы менен графикалык маалыматты алуу, түзүү жана иштетүү ыкмаларын окутат.

Бардык маалыматтын түрлөрү сыяктуу эле компьютердеги сүрөттөлүштөр да экилик коддордун удаалаштыгы түрүндө коддолот. Мындан тышкары, санариптик сүрөттөлүштөрдөгү ар бир чекит да өзүнүн түстүк кодуна ээ. Канчалык чекит жана колдонулган түстөр көп болсо, сүрөттөлүш ошончолук сапаттуу болот.

Түзүү ыкмасы боюнча графикалык сүрөттөлүштөрдү 3 топко бөлсөк болот:

Вектордук – бул параметрлери сан түрүндө, мисалы: өлчөмдөрү, чокуларынын координаталары, жантаюу бурчу, контурдун жана боёктун түсү сакталган жөнөкөй геометриялык фигуралардын топтому түрүндө коддолгон сүрөттөлүштөр.

Вектордук графиканын артыкчылыктары: сапатын жоготпостон масштабдоо мүмкүнчүлүгү, салыштырмалуу өтө чоң эмес маалыматтык көлөм.

Фракталдык – компьютердин эсинде сакталган формула боюнча түзүлгөн сүрөттөлүштөр. Фракталдык графиканын артыкчылыктары: түзүү жөнөкөйлүгү, чексиз масштабдоо мүмкүнчүлүгү – сүрөттүн татаалдыгын көбөйтүү практика жүзүндө файлдын көлөмүнө таасирин тийгизбейт.

Растрдык – ар бири өзүнүн координатасына жана түсүнүн кодуна ээ болгон пикселдердин көптүгүнөн түзүлгөн сүрөттөлүш. Бул графиканын артыкчылыктары: түстү берүү тактыгы жана сүрөттөлүштүн реалдуулукка жакындыгы.

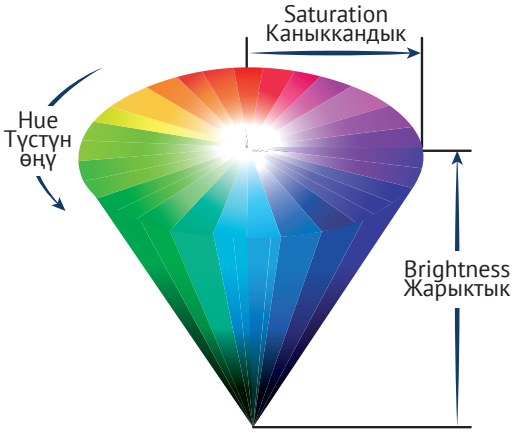
АНЫКТАМА

Дискреттөө – бул графикалык маалыматты аналогдук формадан дискреттик формага өзгөртүп түзүү, башкача айтканда үзгүлтүксүз графикалык сүрөттөлүштү өзүнчө элементтерге бөлүп чыгуу.

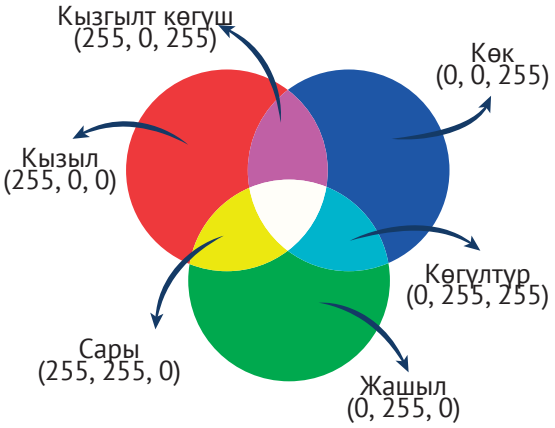
Сүрөттөлүштөр аналогдук формадан санариптик (дискреттик) формага дискреттөө жолу менен, мисалы, сканерлөө жолу менен өзгөртүлүп түзүлөт.

Растрдык сүрөттөлүштөр үчүн коддоо жана түстү берүүнүн негизги үч системасын карап чыгалы: HSB, RGB жана CMYK.

HSB модели үч компоненттен турат: түстүн өңү (Hue), түстүн каныгуусу (Saturation) жана түстүн жарыктыгы (Brightness). Айлананын борборунан чыккан вектор түстүн маанисин берет. Кошумча түстөр вектордун багыты менен аныкталат жана бурчтук градустар менен берилет. Түстүн каныгуусу вектордун узундугу менен аныкталат, ал эми түстүн жарыктыгы өзүнчө окто берилип, анын нөлдүк чекити кара түстү берет. Борборундагы чекит ак түскө (бейтарап), ал эми периметри боюнча жайгашкан чекиттер – таза түстөргө дал келет.



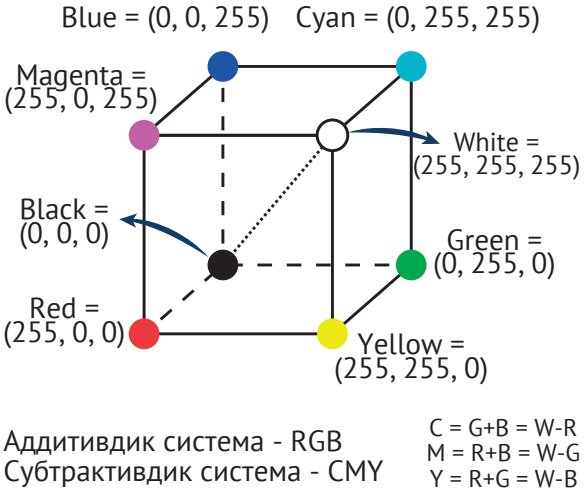
RGB модели: каалагандай түс үч түстүн айкалышынан түзүлөт: кызыл (Red, R), жашыл (Green, G), көк (Blue, B). Калган бардык түстөр жана кошумча түстөрү бул түзүүчүлөрдүн канчалык деңгээлде болгондугунан же жоктугунан алынат.



Мисалы, 256 градациялык түстө ар бир чекит 3 байт менен кодолот, анткени бир түстү коддоо үчүн 1 байт – 8 бит бөлүнөт (2^8=256). Модель өзү үч түстү колдонгондуктан, түстөрдүн мүмкүн болгон айкалыштарын коддоо үчүн 3 байт керектелет.

RGBнын (0, 0, 0) минималдык мааниси кара түскө туура келет (экрандын минималдык жарыктыгы), ак түскө – максималдык мааниси RGB (255, 255, 255), ал эми боз түскө – ар бир түстүн барабар мааниси туура келет (мисалы, RGB(35, 35, 35)).

CMYK модели басмаканада басып чыгарууларда колдонулат. Кошумча түстү калган эки түстү кошуу менен же ак түстөн кемитүү менен алат. Кызыл түс үчүн кошумча түстөр көгүш (Cyan, C) = жашыл + көк = ак – кызыл, жашыл түс үчүн – кочкул кызыл (Magenta, M) = кызыл + көк = ак – жашыл, көк түс үчүн – сары (Yellow, Y) = кызыл + жашыл = ак – көк. CMYK моделиндеги түстөрдүн аралашмасы накта кара түстү бербейт, ошондуктан K (Black) тамгасы менен белгиленген кошумча кара түстүн компоненти колдонулат.



Түстүү графиканы берүүнүн бир нече режимдерин айырмалашат:

а) толук түстүү (True color) – ар бир үч негизги түстүн жарыктыгын коддоо үчүн 256 кошумча түс колдонулат (2^8=256, сегиз экилик разряд), башкача айтканда, бир пикселдин түсүн коддоо үчүн (RGB системасында), 8*3=24 разрядды коротуш керек. Бул 16,5 млн түстү аныктоого мүмкүндүк берет. Түстүү графиканы берүү үчүн CMYK системасынын жардамы менен коддоодо 8*4=32 экилик разрядга ээ болуу керек (2^{32}=4,3 млн.).

б) High Color – ар бир чекитти коддоодо экилик разряддардын санын эки эсеге азайтуудан келип чыккан режим. Бул 16 разряддуу экилик сандардын жардамында коддолот. Бирок бул учурда түстөрдүн диапазону да ошончолук азаят.

Чагылдырылган түстөрдүн саны (K) менен аларды коддоочу биттердин санынын (a) дал келүүсү төмөнкү формула менен аныкталат:

$$K = 2^a$$

Түстөр экилик коддо коддолот. 16 түстүү сүрөттөлүштү коддоо үчүн ар бир пикселге 4 бит ($16=2^4$) керектелет. Ал эми 16 бит (2 байт) 2^{16}=65536 түрдүү түстөрдү кодго айландыруу үчүн колдонулат. Бир чекиттин түсүн коддоо үчүн 3 байтты (24 бит) колдонуу 16777216 (же 17 миллионго жакын) түстүн ар кандай өңдөрүн чагылдырууга мүмкүндүк берет.

Монитордун экранындагы графикалык маалымат

Сүрөттөлүштүн сапаты мониторду чечүүчү жөндөмдүүлүгү менен аныкталат, б.а. сүрөт түзүлгөн чекиттердин саны менен. Чечүүчү жөндөмдүүлүгү канчалык чоң болсо, сүрөттөлүш дагы ошончолук сапаттуу.

АНЫКТАМА

Чечүү – сүрөттөлүштүн бир дюйм өлчөмүнө туура келүүчү пикселдердин саны.

Түстүн тереңдиги – бул пикселдин түсүн коддоо үчүн колдонулган биттердин саны.

Монитордун экраны негизги эки режимде иштей алат: тексттик жана графикалык.

Графикалык режимдер төмөнкүдөй көрсөткүчтөрү менен мүнөздөлөт:

1. **Чечүүчү жөндөмдүүлүгү** – чекиттердин саны, алардын жардамы менен экранда сүрөттөлүштү алууга мүмкүн болот: мисалы, 1920 x 1080.
2. **Түстүн тереңдиги** – чекиттин түсүн коддоо үчүн колдонулган биттердин саны, мисалы, 8,16, 24, 32 бит. Монитордун экранында чагылдырылган түстөрдүн саны төмөнкү формула менен аныкталат:

$$K=2^I,$$

мында K – түстөрдүн саны, I – түстүн тереңдиги же биттик тереңдик

Таблица. Түстүн тереңдиги (I) жана чагылдырылган түстөрдүн саны (K)

Түстүн тереңдиги (I)	Чагылдырылган түстөрдүн саны (K)
1	2 (кара жана ак)
2	$2^2 = 4$
4	$2^4 = 16$
8	$2^8 = 256$
16 (HighColor)	$2^{16} = 65\,536$
24 (True Color)	$2^{24} = 16\,777\,216$
32 (True Color)	$2^{32} = 4\,294\,967\,296$

Сүрөттөлүштү коддоонун сапаты төмөнкүлөрдөн көз каранды:

1. **Дискреттөө жыштыгынан,** б.а. сүрөттөлүш бөлүнгөн фрагменттердин өлчөмүнөн.
2. **Коддоонун тереңдигинен,** б.а. бир чекит үчүн керек болгон түстүн санынан.

АНЫКТАМА

Видеоэс – бул графикалык сүрөттөлүш түзүлгөн оперативдүү эстин бөлүгү

Монитордун экранында сүрөттөлүштү чыгаруу үчүн сүрөттөлүштүн бардык чекиттеринин түстөрү сакталa турган компьютердин видеоэси керек. **Видеоэстин көлөмү** төмөнкү формула менен эсептелет:

$$V=I*X*Y,$$

мында I – өзүнчө чекиттин түсүнүн тереңдиги,

X, Y – экрандын горизонталдык жана вертикалык өлчөмдөрү (Хтин Уке көбөйтүндүсү экрандын чечүүчү жөндөмдүүлүгүн берет).

Мисалдар:

1-маселе. Ак-кара (боз градациясыз) растрдык графикалык сүрөттөлүш 10*10 чекиттик өлчөмгө ээ. Бул сүрөттөлүш эстин кандай көлөмүн ээлейт?

Чыгаруу:

Чекиттердин саны – 100

Болгону 2 түс болгондуктан: ак жана кара, түстүн тереңдиги 1ге ($2^1=2$) барабар.

Видеоэстин көлөмү 100*1 = 100 бит болот.

Жообу: 100 бит.

2-маселе. Өлчөмү 128 x 128 пиксель болгон растрдык сүрөттөлүштү сактоо үчүн 4 Кб эс бөлүнгөн. Сүрөттөлүштүн палитрасындагы түстөрдүн мүмкүн болгон максималдык саны канчага барабар?

Чыгаруу:

Сүрөттөлүштү түзгөн чекиттердин санын аныктайбыз. 128 * 128 =16384 чекит же пикселдер. Сүрөттөлүш үчүн берилген эстин 4 Кб көлөмүн бит менен туюнталы, анткени V = I *X * Y бит менен эсептелет. 4 Кб = 4*1024 = 4096 байт = 4096 * 8 бит = 32768 бит. Түстүн тереңдигин табабыз:

I = V/(X * Y) = 32768/16384 = 2

$N = 2^I$, мында N – палитрадагы түстөрдүн саны. $N = 2^2 = 4$.

Жообу: 4

3-маселе. Чечүүчү жөндөмдүүлүгү 1024 х 700 чекиттен жана түстүн палитрасы 65536 түстөн турган монитордун High Color режимин берүүгө керек болгон компьютердин видеоэсинин көлөмүн килобайт менен аныктагыла.

Чыгаруу:

1. K = 2^I формуласы боюнча, K – түстүн саны, I – түстүн тереңдиги. 2^I = 65536 болгондуктан, таблица боюнча түстүн тереңдиги I = 16 битке (216 = 65 536) барабар болот.

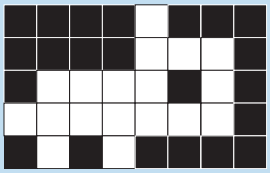
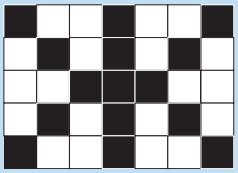
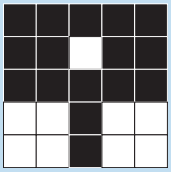
2. Сүрөттөлүштүн чекиттеринин саны 1024 * 700 = 716 800гө барабар

3. Видеоэстин талап кылынуучу көлөмү 16 бит * 716 800 = 11 468 800 битке барабар болот.

Эми муну килобайтка которобуз:
11 468 800 бит / 8 =1 433 600 байт
1 433 600 байт / 1024 = 1400 Кб
Жообу: 1400 Кб

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Берилген ак-кара сүрөттөр үчүн экилик коддорду түзгүлө жана аларды он алтылык эсептөө ситемасында жазгыла:*

а),  *б),*  *в).* 

2) *Түстүн тереңдиги менен файлдын көлөмү кандай байланышта?*

3) *Эгерде сүрөттө 65536 түс колдонулса, анын түсүнүн тереңдиги кандай? 256 түстүкүчү? 16 түстүкүчү?*

4) *64 түс колдонулган палитра файлда канча орунду (көлөм) ээлейт? 128 түс колдонулсачы?*

5) *Өлчөмү 800х1000 пиксель болгон жана 1 млн. түстү камтыган сүрөт 1Гб көлөмдөгү флешкага батабы?*

6) *Монитордун экраны – 1024х768 чекиттен турат, түстүн тереңдиги – 16 бит. Бул графикалык режим үчүн керек болгон видеоэстин көлөмү канчага барабар?*

2 - бөлүм

Компьютер жана ПК

2.1-тема:

Компьютердик графика

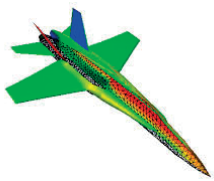
Компьютердик графика – андан ары сактоо жана иштетүү максатында реалдуу дүйнөдөн алынган визуалдык маалыматты санариптештирүүгө, түзүүгө, редакциялоого мүмкүндүк берүүчү компьютердик технологиялардын бөлүмү.

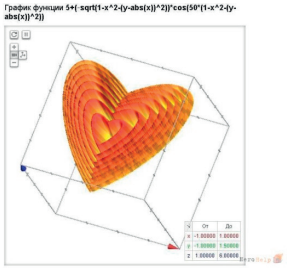
Компьютердик графиканы ар кандай кесиптин ээлери колдонушат: инженерлер жана конструкторлор, архитекторлор жана астрономдор, дизайнерлер жана модельерлер, сүрөтчүлөр жана окумуштуулар. Алардын ар бири үчүн графикалык программалар же графикалык пакеттер деп аталган атайын программалык камсыздоо түзүлөт.

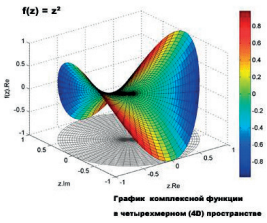
Негизги колдонуу аймактары

1 **Иш жана илимий графика** илимий изилдөө объекттерин графикалык иштетүү үчүн, андан тышкары сандык берилиштерди график, маалымдама, иллюстрация түрүндө көрсөтмөлүү берүү үчүн кызмат кылат.

Мисалдар:







Инженердик графика менен иштөөчү программалардын түрлөрү:

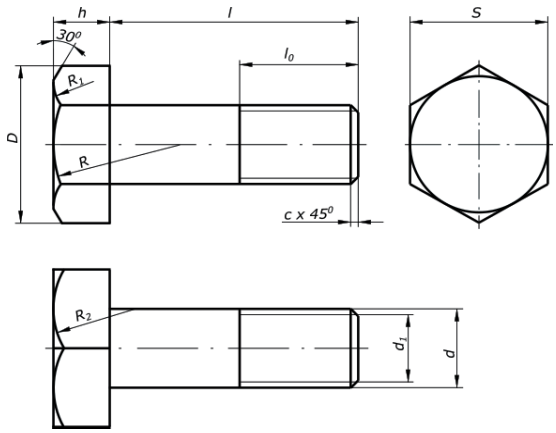
Gnuplot – эки жана үч өлчөмдүү графиканы түзүү үчүн эркин таралуучу программа.

Zhu3D - математикалык функцияларды жана жыйынтыктарды эсептөөчү OpenGLге негизделген интерактивдүү пакет. Анимация эффектиси, текстураны өзгөртүү, сүрөттөлүштү айландыруу ж.б. функциялары бар.

SciDAVis – илимий берилиштерге талдоо жүргүзүү жана визуалдаштыруу үчүн эркин ПК. Бир нече графикалык форматтарда, андан тышкары PDF, EPS же SVG форматтарында сактала алган 2D жана 3D графиктерди түзө алат. Python тили үчүн скрипттик интерфейсти да камтыйт.

MapViewer масштабын, координаталарды ж.б. өзгөртүү менен картаны түзүүгө жана корректирлөөгө мүмкүндүк берүүчү программа. Ал демографиялык берилиштерди иштетүүгө жана визуалдаштырууга мүмкүндүк берет.

2 **Инженердик графика** инженер-конструкторлордун, архитекторлордун, жаңы техниканы ойлоп табуучулардын ишинде кеңири колдонулат. Компьютердик графиканын бул түрү ДСАнын (долбоорлоо системасын автоматташтыруу) негизги элементи болуп саналат. Конструктордук графика тегиздиктеги сүрөттөлүштөр (проекция, кесилиш) жана мейкиндиктиктеги үч өлчөмдүү сүрөттөлүштөрдү да алууга мүмкүндүк берет.



3 **Иллюстративдик графика** – иллюстративдик графиканын пакеттери жалпы колдонуудагы ПКга кирет. Мисал – графикалык редакторлор (Gimp, Adobe Photoshop, Adobe Illustrator, CorelDraw ж.б.).

4 **Көркөм жана жарнамалык графика** – анын жардамы менен жарнамалык роликтер, мультфильмдер, компьютердик оюндар, видеосабактар, видеопрезентациялар жасалат. Бул максаттар үчүн графикалык пакеттер компьютердин көп ресурсун: ылдамдыкты жана эсти талап кылат. Башкалардан айырмаланган өзгөчөлүгү болуп реалдуулукка жакын жана «кыймылдуу» сүрөттөлүштөрдү түзүү мүмкүнчүлүгү эсептелет. Үч өлчөмдүү объекттердин сүрөттөрүн алуу, аларды айландыруу, жакындатуу, алыстатуу, деформациялоо, көп көлөмдөгү эсептөөлөр менен байланышкан. Жарыктын булагына жараша, көлөкөнүн жайгашуусуна жараша, беттин фактурасына жараша объекттин жарыктандыруусун берүү оптиканын мыйзамдарын эске алып эсептөөнү талап кылат.

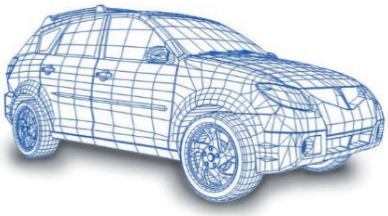
5 **Компьютердик анимация** – дисплейдин экранында кыймылдуу сүрөттөлүштөрдү алуу. Сүрөтчү экранда кыймылдагы объекттин баштапкы жана акыркы абалын гана түзөт, ал эми эки абалдын арасындагы калган бардык абалдарды компьютер математикалык эсептөөлөргө таянып аткарат. Белгилүү жыштыкта экранга ырааттуу чыгарылуучу алынган сүрөттөлүштөр кыймылдын иллюзиясын берет.

Сүрөттөлүштөр берүү ыкмасы боюнча экиге бөлүнөт:

• Эки өлчөмдүү графика **(2D)** – тегиздиктеги сүрөттөлүш же видео,

• Үч өлчөмдүү графика **(3D)** – үч өлчөмдүү мейкиндикте моделденген көлөмдүү объекттер, сүрөттөлүш же видео.

3D моделдөө – бул реалдуу объекттин (автомобиль, имарат, чагылган, астероид) же абстрактуу объекттердин (бир эле сүрөттөлүш кичирейтилген же чоңойтулган масштабда кайталана берген геометриялык фигуралардын фракталынын проекциясы) маалыматтык моделинин көлөмдүү образын иштеп чыгуу.



Үч өлчөмдүү сүрөттөлүштү түзүүнүн ирети:

1. Моделдөө – объекттердин үч өлчөмдүү математикалык моделин түзүү.
2. Текстуралоо – материалдын касиеттерин жана текстураларын иштеп чыгуу (мисалы, тунуктугу), растрдык, вектордук же процедуралык текстураларды көчүрүп көбөйтүү.
3. Жарыктандыруу – виртуалдык жарык булактарын моделдөө (жарыктын багыты, жарыктандыруу даражасы).
4. Анимация – объекттердин кыймылын түзүү.
5. Динамикалык симуляция – ар түрдүү элементтердин бири-бири менен жана моделденип жаткан процесстер менен өз-ара аракеттешүүсүн автоматтык түрдө эсептөө (мисалы, шамал, гравитация ж.б.).
6. Визуалдаштыруу (рендеринг) – 3 өлчөмдүү моделди 2 өлчөмдүү мейкиндикте (монитордун экранында) көрсөтүү.

Колдонулган программалар: Autodesk 3ds Max, Autodesk Maya, Autodesk Softimage, Blender, Cinema, 4D Houdini, LightWave 3D.

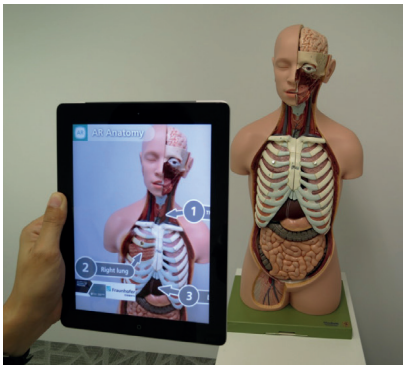
Мындагы программалардын ичинен үч өлчөмдүү графиканы түзүү үчүн Blender эркин ПК болуп саналат.

Blender татаал үч өлчөмдүү моделдерди жана интерактивдүү оюндарды түзүү үчүн колдонулушу мүмкүн жана өзүнө төмөнкүлөрдү камтыйт:

- моделдөө каражаттарын;
- анимацияларды;
- рендерингди;
- кийинки иштетүүнү;
- видеону үнү менен монтаждоону;
- «түйүндөрдүн» жардамы менен чогултууну.

Blender чөйрөсүндө программалоо Python тилинде ишке ашат.

3D графиканын өзгөчөлөнгөн артыкчылыгы болуп «кошумчаланган реалдуулук» (augmented reality, AR) эсептелет. Бардык айлана-чөйрө компьютердик графикадан түзүлгөн «виртуалдык реалдуулуктан» айырмаланып «кошумчаланган реалдуулукта» компьютердик графика кээ бир бөлүктөрүндө гана колдонулат. Мисалга алсак, Инстаграмм тиркемесиндеги «Маскалар» ушул технологиянын негизинде түзүлгөн: сиз камераны өзүңүздүн бетиңизге багыттайсыз, ал эми программа өзү эле беттин сүрөтүнө мультфильмдеги күчүктүкүндөй кулак жана мурунду, көз айнек же болбосо жасалгаларды «жабыштырып» коёт.



Башкача айтканда, реалдуу табигый чөйрөдө сүрөттөлүштөрдү таануу технологиясы (маркерлер) колдонулат, ал эми ага кошумчаланган реалдуулуктун программасы виртуалдуу 3D объекттерди кошуп көрсөтөт. Сиз маркерди өзгөртсөңүз болот: ар кандай багыттарга айлантып, ар түрдүүчө жарыктандырып, кээ бир бөлүктөрүн жаап ж.б. Бул учурда экрандагы 3D объекттердин абалы да ошого жараша өзгөрөт.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Адамдын күзгүдөгү сүрөттөлүшүн визуалдаштыруу кайсы компьютердик графикага кирет?*
2) *3 өлчөмдүү графика 3D моделдөөдөн эмнеси менен айырмаланат?*
3) *«Кошумчаланган реалдуулукту» дагы кайсы жерлерде колдонсо боло тургандыгына мисал келтиргиле.*

2.2-тема:

Робототехникага киришүү

Биз азыр көптөгөн үйдөгү жана жумуштагы майда-барат иштерди интеллектуалдуу жардамчылардын мойнуна илип койсо боло тургандай заманда жашап жатабыз. Мындай учурларда бизге жардамга роботтор келет.

Робот – бул программалануучу жана адамдын жардамысыз эле сырткы чөйрө менен аракеттешип, андан ары биз киргизген маселелерди аткаруучу механикалык түзүлүш.

Робототехника – программалануучу механикалык түзүлүштөрдү – роботторду долбоорлоо, иштеп чыгуу жана пайдалануу менен байланышкан иштердин чөйрөсү.

БУЛ КЫЗЫКТУУ

«Робот» деген сөздү 1921-жылы өзүнүн эркиндиги үчүн күрөшкөн, адам түспөлдүү жасалма кулдардын классы сүрөттөлгөн «Россумдун универсалдуу роботтору» деген чыгармасында чехиялык драматург Карл Чапек эң биринчи колдонгон. «Robota» деген чех сөзү «аргасыз кулчулук» дегенди түшүндүрөт. Ал эми «робототехника» сөзү 1941-жылы биринчи жолу илимий фантастиканын автору Айзек Азимов тарабынан колдонулган.

Бүгүнкү күндө роботтор көп милдеттерди аткарышат жана көптөгөн колдонулуштарга ээ. Роботтордун колдонулушу төмөнкүдөй аймактарга бөлүнөт:

• **өндүрүштүк роботтор** – аларды колдонуу көптөгөн өндүрүштүк процесстердин (жыйноо, чогултуу, боёо жана таңгактоо) тактыгын жогорулатууга жана ылдамдатууга мүмкүндүк берди. Роботтун мындай түрлөрү техникалык көрүү системалары менен камсыз болгон жана билдиргичтерден (датчиктерден) келген татаал жооптун негизинде чечим чыгара алышат. Алар татаал, коркунучтуу тапшырмаларды аткаруу үчүн колдонулушу мүмкүн, андан тышкары адам аткара албаган тапшырмаларда (бомбаны зыянсыздандырууда, ядролук реакторлорду тейлөөдө, океандын тереңдиктерин жана космосту изилдөөдө) колдонулат;

• **изилдөөчү роботтор** – опурталдуу жана татаал чөйрөлөрдө (космос мейкиндигин жана күн системасынын планеталарын изилдөөдө ж.б) тапшырмаларды аткаруу үчүн колдонулат;

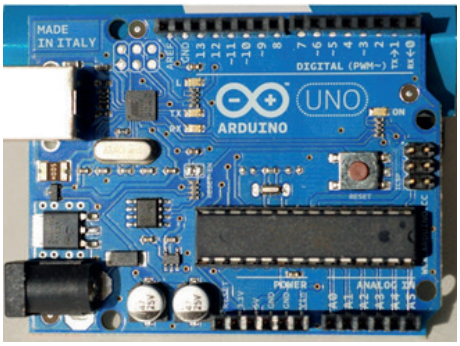
• **билим берүүчү роботтор** – окуу маселелерин аткаруу жана долбоорлоо үчүн колдонулат.

Заманбап роботтордун негизги компоненттери:

• **механикалык түзүүчүсү (тулкусу/алкагы)** – бул түздөн-түз роботтун конструкциясы, ал ар кандай формада жана өлчөмдө болушу мүмкүн. Робот-тор адам түспөлдүү болушу да мүмкүн, ошол эле учурда адамга таптакыр окшошпошу да мүмкүн, анткени роботтун долбоорунда сырткы келбетке эң аз, ал эми анын түрдүү милдеттерди аткарышына көп көңүл бурулат.

• **Программалык түзүүчүсү (башкаруу системасы)** – роботтун бардык элементтерин башкарат. Мисалы, роботтун билдиргичтери сырткы чөйрө менен аракеттенишкенде борбордук процессорго сигналды жөнөтүшөт, ал болсо программалык камсыздоонун жардамы менен берилиштерди иштетет да логиканын негизинде чечим кабыл алат. Ушундай эле көрүнүш колдонуучу тарабынан командаларды киргизүүдө да аткарылат.

Arduino



Arduino – бул өзүндө механикалык жана программалык түзүүчүлөрдү камтыган бир платалуу микрокомпьютер. Бул окуучулар арасында эң кеңири тараган аппараттык платформа болуп эсептелет, ал ар түрдүү роботторду курууга окутуу үчүн колдонулат. Arduino автономдуу интерактивдүү роботторду түзүүдө да жана ошону менен бирге компьютерде аткарылып жаткан программалык камсыздоого кошулууда да колдонула берет.

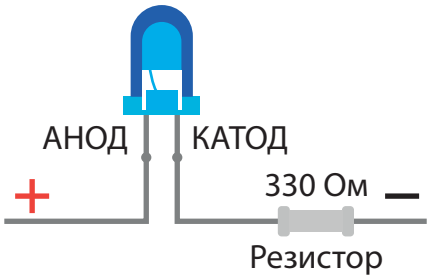
Робототехникада Arduino көп учурда **роботтун мээси** катары колдонулат.

Плата ар кандай жөнөкөй түзүлүштөрдү: баскыч (кнопка), билдиргич (датчик), мотор, экран сыяктуу кошууга арналган көптөгөн аналогдук жана санариптик портторго ээ.

Arduinonун ичинде жүктөгүч (загрузчик, BootLoader) - сиз ар дайым аппаратты иштеткенде кошо ишке кирген программа орнотулган болот. Arduinону программалоо үчүн платаны персоналдык компьютерге USB порт аркылуу туташтырып коюу жетиштүү болот.

Программаны жазуу үчүн эркин таралган Arduino IDE редактору колдонулат. Аны төмөнкү шилтеме аркылуу көчүрүп алууга болот:
http://www.arduino.cc/en/Main/Software.

1-маселе. Arduinoго жарык диодун кошобуз жана аны өчүп, күйүп тургандай кылабыз. Бул робототехникадагы бирден-бир базалык схема болуп саналат. Платага жарык диодун кошуудан мурда анын түзүлүшүн карайлычы. Жарык диоду бири экинчисинен узун келген эки чыгуучу зымдан (бутчалардан) турат. Узун чыгуучусу ток булагынын (мисалы батарейка) оң уюлуна туташкан – **анод** болуп саналат.

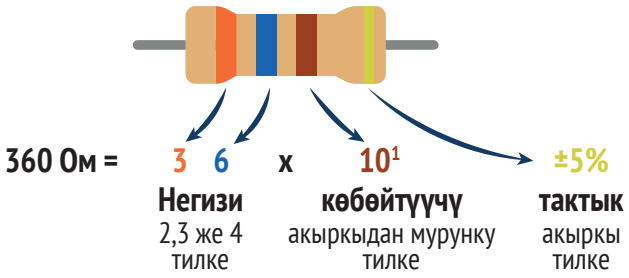


Катод – кыска чыгуучусу, минус менен же жалпы өткөргүч менен бириктирилет.

Резистор – бул ток үчүн жасалма «тоскоолдук». Ал электр энергиянын бөлүгүн жылуулукка айландыруу менен токтун күчүн чектейт. Резистордун чыгуучулары плюс да, минус да болуп эсептелбейт, ошондуктан ал уюлдуу эмес болуп саналат.

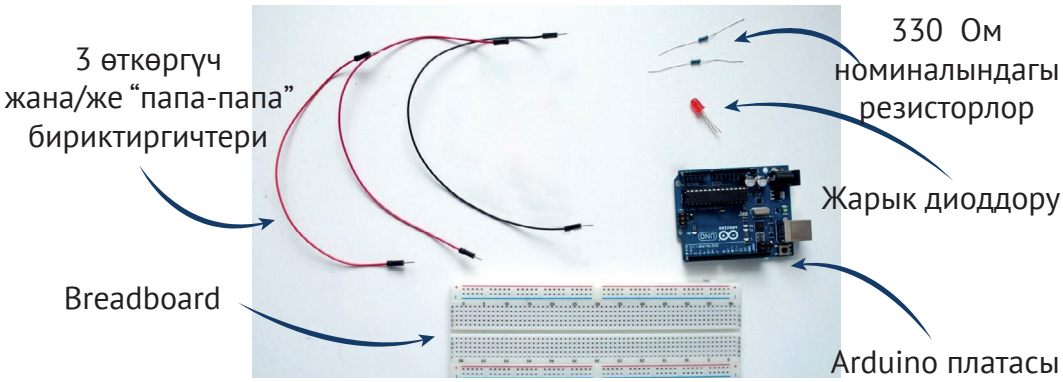
Резистордун эң негизги мүнөздөмөсү болуп каршылыгы саналат. Каршылыктын өлчөө бирдиги – **Ом**. Каршылык канчалык көп болсо, ошончолук көп ток жылуулукка айланат.

Каршылык деңгээлин же **резистордун номиналын** резисторду курчаган түстүү тилкелер менен аныкташат:

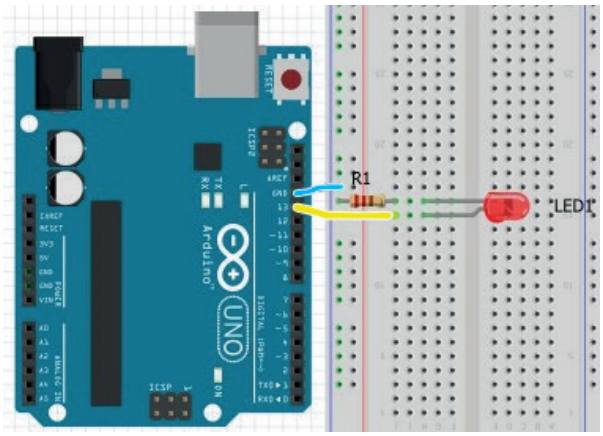


Жарык диодун резистор аркылуу гана жандыргыла!

Моделди чогултуу үчүн бизге төмөнкүлөр керек болот:



Жарык диодун туташтыруу схемасы:



Бул схеманын иштеши үчүн төмөнкү программа керек болот (программаны силер Arduino IDE редакторуна көчүрүп алгыла)

```
void setup() { //программаны баштоо үчүн setup милдеттүү функциясы
pinMode (13, OUTPUT);//13-пинди чыгууга (OUTPUTка) орнотуу
}
void loop() { //loop функциясы программаны циклде кайталайт
digitalWrite (13, HIGH); //жарык диодун жандыруу үчүн команда
delay (1000);     //1000 миллисекундага(1 сек.) кармалуу
digitalWrite (13, LOW);  //жарык диодун өчүрүүгө команда
delay (1000);     //1000 миллисекундага(1 сек.) кармалуу
}
```

Жыйынтыгында биз жарык диодунун 1 секунда сайын өчүп, күйгөнүн байкайбыз.

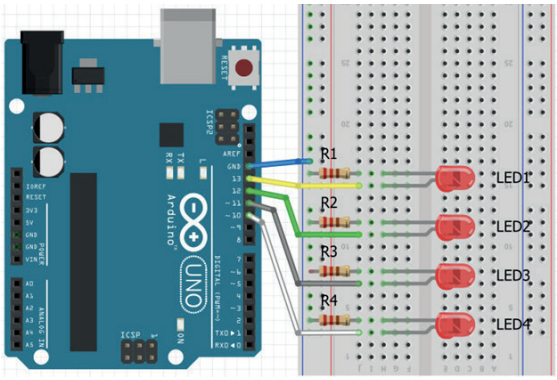
Arduino программалоо чөйрөсүндө функцияны чакыруу **void** сөзүнөн башталат.

Void setup() функциясы милдеттүү функция болуп, программанын башында ишке кирет. Бул функция менен аткарылган командалын блоктору фигуралуу кашаанын ичинде жазылат. Бул командаларды ал бир жолу гана — программанын башталыш учурунда аткарат.

Биздин мисалда **Void setuptын** ичинде **pinMode** нускамасы 13-пинди «чыгуу» (OUTPUT) режимине орнотот.

Void loop функциясы **Void setup** функциясынан кийин чакырылат жана командаларды Arduino платасы иштеп турган (электр тогу берилип турганда) бардык убакытта (циклдик түрдө) аткарыла берет. Биздин мисалда **delay** (1 секундага кармалуу) функциясы жарык диодунун өчүп, күйгөнүн ажыратып байкоого мүмкүндүк берет. Ансыз өчүрүп/күйгүзүү сигналын ажыратуу мүмкүн эмес.

2-маселе. Эми биздин схемага дагы 3 жарык диодун кошолу. Бул схема «жалпы катоду менен кошулуу схемасы» деп аталат.



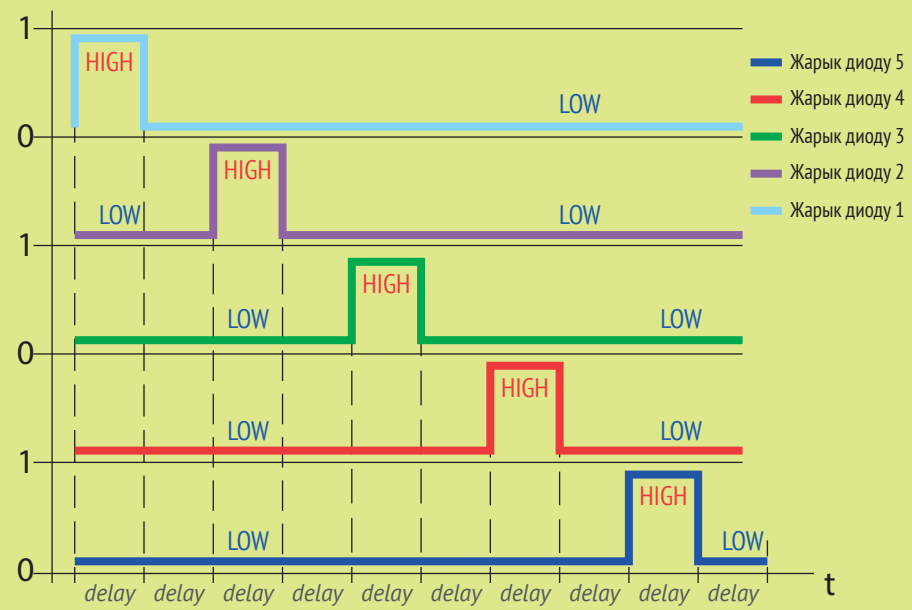

```
void setup() {
pinMode (13, OUTPUT);
//колдонулуп жаткан пиндерди чыгууга орнотолу
pinMode (12, OUTPUT);
pinMode (11, OUTPUT);
pinMode (10, OUTPUT);

}
void loop() {
digitalWrite (13, 1);       //жарык диодун күйгүзөбүз
delay (1000);
digitalWrite (13, 0);       //жарык диодун өчүрөбүз
delay (1000);    //1000 миллисекундага (1 секунда) кармалуу
digitalWrite (12, 1);
delay (1000);
digitalWrite (12, 0);
delay (1000);
digitalWrite (11, 1);
delay (1000);
```

```
digitalWrite (11, 0);
delay (1000);
digitalWrite (10, 1);
delay (1000);
digitalWrite (10, 0);
delay (1000);
}
```

Силер байкагандай, биз HIGH жана LOW командаларын 1 жана 0гө алмаштырдык, бул силердин программаңардын иштешине эч кандай таасир эткен жок.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Роботтун аяктары кандай аталат?*
2) *Интернеттен робототехниканын 3 мыйзамы боюнча маалымат тапкыла. «Терминатор» фильминдеги башкы каарман Арнольд Шварценеггер ушул мыйзамдардын чегинде иш алып бардыбы? Айтып бергиле.*
3) *Бардык жарык диоддору бир убакта күйүп, кайра бир убакта өчүшү үчүн кандай кылуу керек?*
4) *Беш жарык диоду кошулган схеманы чогултқула жана чуркоочу жарыктар үчүн төмөнкү алгоритм боюнча программа түзгүлө.*



3 - бөлүм

Программалоо

3.1-тема:

Рекурсия

Рекурсия – бул күнүмдүк турмушта кеңири тараган түшүнүк, бирок ал информатика жана математикада эң көп колдонулат. **Рекурсия** деп качан кайсы бир объект өзүн-өзү кайталаган процессти түшүнүшөт. Мисалы, тоок жумуртканы тууп, андан кайра тооктун пайда болушу. Же болбосо даракта бутактар өсүп, ал бутактарда андан да майда бутактардын өсүп чыгуусу.

Сүрөттүн автору: Пьер Метивье

Чексиз рекурсиянын классикалык мисалы катары бири-бирине карап турган күзгүнү алсак болот: аларда күзгүдөгү бара-бара өчүп бараткан чагылуулардан турган коридор пайда болот.

8-класстын курсунан силер билесиңер, көпчүлүк учурда бир функция башка функцияны чакырышы мүмкүн. Бирок ошол эле функция өзүн-өзү да чакырышы мүмкүн. Программа түздөн-түз өзүн чакырган (жөнөкөй рекурсия) же кыйыр (башка функциялар аркылуу), мисалы, **A** функциясы **B** функциясын чакырат, ал эми **B** функциясы **A**ны чакырган кырдаалдар **рекурсия** деп аталат.

Байкагандай, кыйыр кайрылууда чынжырдагы бардык функциялар – рекурсивдүү. Келгиле, программалоодо рекурсия кантип иштерин карап көрөлү.

1-маселе. 1 ден n ге чейинки бардык натуралдык сандардын суммасын эсептөө керек болду. Мисалы, эгер колдонуучу тарабынан 5 саны киргизилсе, анда программа 1ден 4кө чейинки сандардын суммасын чыгарыш керек. Программаны жазалы:

```
def summa(n):
    if n == 1:
        return 1
```

```
        else:
            return n + summa(n-1)
    print (summa(4)) #жыйынтык 10
```

Чыгарылышы мындай болот: **4 + (4-1) + ((4-1)-1) + (((4-1)-1)-1).** Б.а. `summa` функциясына n 1ге барабар болуп калганча 1ге кичине болгон аргумент берилип турат. «if n == 1» туюнтмасы **рекурсиянын базалык шарты** болуп эсептелет. Ушул туюнтманын негизинде гана рекурсия токтойт. Базалык шарты жок рекурсия чексиз болуп калышы мүмкүн.

Биз `return` оператору функцияны токтоторун билебиз, ошондуктан андан кийин `else` шартын жазуу – мааниге ээ болбой калат. Программанын туура жазылышы төмөнкүдөй болот:

```
    def summa(n):
        if n == 1:
            return 1
        return n + summa(n-1)
    print (summa(6))  #жыйынтык 21
```

2-маселе. n! саны үчүн факториалды эсептөө мисалында рекурсия кантип иштерин карайлы. Мисалы, 3 санынын факториалын эсептөө үчүн, 3*2*1ге көбөйтүү керек, б.а. n*(n-1)*((n-1)-1) амалын аткаруу керек.

```
def factorial(n):
  if n == 0:
    return 1
  return n * factorial(n - 1)
print(factorial(3))
>>>
6  #жыйынтык
```

Ушул эле сандын факториалын эсептөө маселесин `while` циклин колдонуп да жазса болот:

```
    n = int(input())
    factorial = 1
    while n > 1:
        factorial *= n
        n -= 1
    print (factorial)
```

ЭСИҢЕ ТУТ

n санынын факториалы деп, 1ден **n**ге чейинки бардык натуралдык сандардын көбөйтүндүсүн айтабыз:

n!=1*2*3*4*...*n

Мисалы, 5 санынын факториалы 120га барабар 120 (5!=1*2*3*4*5).

`for` циклинин жардамы менен факториалды эсептөө:

```
n = int(input())
factorial = 1
for i in range(2, n+1):
    factorial *= i
print (factorial)
```

ЭСИҢЕ ТУТ

Чексиз рекурсиянын жыйынтыгын чыгарууну токтотуу үчүн **Ctrl+C** баскычтарынын комбинациясын консолдо басуу керек.

Мындан маанилүү корутунду чыгат: **рекурсия циклди алмаштырат**. Рекурсияны колдонуу чыгаруунун **рекурсивдүү ыкмасы** деп, ал эми циклди колдонуу – **итеративдүү** («итерация» сөзүнөн) деп аталат. Итерация – бул бир эле коддун блогун кайталап аткаруу.

3-маселе. a санын b даражасына көтөрүү (a**b) үчүн рекурсивдүү функцияны жазалы. Санды даражага көтөрүүнүн алгоритми төмөнкүдөй аткарылат:

(a *...(a * (a * (a *(a * 1)))))

Математика курсунан биз a**0 = 1 экенин билебиз.

Маселени чыгаруу:

```
def func_step(a, b):
    if b == 0:
        return 1
    return a * func_step(a, b-1)
print(func_step(2, 4))  #мисалы, 2нин 4-даражасы
>>>
16  #жыйынтык
```

БУЛ КЫЗЫКТУУ!

• Рекурсиянын адамдын жашоосундагы мисалы – банк депозити. Сиз банкка процентке акча салдыңыз жана эсептелген проценттер сиздин эсебиңизде банкта калат жана аларга да проценттер эсептелет. Сиз качан банктан чогулган акчаларды алып кеткенче, процесс улана берет.

• Телефондун фронталдык камерасын күзгүгө каратып сүрөт тартсаңыз андагы сүрөттүн чагылышы да чексиз рекурсиянын мисалы болот.

Рекурсивдүү функциялар программалоодо көптөгөн маселелерди чечет, бирок тилекке каршы, аларды жазууда көп учурда каталар кетет. Эң көп таралган ката – бул чексиз рекурсия, бул учурда функцияны чакыруу чынжыры эч качан бүтпөйт жана компьютердин эс тутуму толуп калмайынча улана берет.

Көптөгөн чексиз рекурсиянын негизги эки себеби:

1. Рекурсиядан чыгуунун туура эмес жазуусу. Мисалы, эгер биз факториалды эсептөө программасында `if n==0` деген текшерүүнү унутуп койсок, анда factorial(0), factorial(-1)ди, ал болсо factorial(-2)ни чакырат ж.у.с.
2. Функциянын параметрлеринин туура эмес жазылышы. Мисалы, эгер factorial(n) функциясы кайрадан эле factorial(n) функциясын чакырса да чексиз чынжыр алынат.

Демек, рекурсивдүү функцияны иштеп чыгууда эң биринчи кезекте ***рекурсиянын базалык шарты*** деп аталган рекурсияны аяктоо шартын туура түзүү керек.

Андан тышкары, рекурсивдүү эсептөөлөр көбүрөөк компьютердик эс тутумду жана эсептөөгө кеткен убакытты талап кылат. Ошондуктан, рекурсивдүү программалардан көрө кадимки циклдик алгоритмдерди колдонуу сунушталат.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

***1) n** берилген санынын цифраларынын суммасын чыгаруучу рекурсивдүү функцияны жазгыла.*
***2)** Алынган сандын а) жуптугун; б) тактыгын аныктаган рекурсивдүү функцияларды жазгыла.*
***3)** Эң кичине жалпы бөлүүчүнү (ЭКЖБ) табуу үчүн рекурсивдүү функцияны жазгыла.*
***4)** Арифметикалык прогрессиянын айырмасы жана биринчи мүчөсү берилген. Прогрессиянын биринчи n мүчөлөрүнүн суммасын табуучу рекурсивдүү функцияны жазгыла.*
***5)** Геометриялык прогрессиянын бөлүмү жана биринчи мүчөсү берилген. Прогрессиянын биринчи n мүчөлөрүнүн суммасын табуучу рекурсивдүү функцияны жазгыла.*

3.2-тема:

Тизмелерди иштетүү алгоритмдери

8-класста силер тизмелер менен ар кандай амалдарды жасап көргөнсүңөр.

Бул темада биз тизмедеги берилиштерди издөө жана модификациялоонун (реверс, жылдыруу жана керектүү элементтерди тандоо) негизги стандарттык алгоритмдерин тереңирээк карайбыз.

ОШОНДОЙ ЭЛЕ КАРА:

8-класс

3.2-тема **Тизмелер, кортеждер, сөздүктөр**

3.6-тема **Массивдер**

Тизмедеги элементтерди издөө

Программалоо боюнча көптөгөн маселелерде тизмеде элемент бар же жок экенин текшерүү керек болот. Же болбосо максималдык же минималдык мааниси менен элементти табуу керек болот.

a – массивдин аты, **n** - тизмедеги элементтердин саны (бүтүн сан), ал эми **i** өзгөрмөсү тизменин элементинин индексин билдирген маселенин мисалында издөө алгоритмин карап көрөлү.

1-маселе. Тизмеден мааниси **x**ке барабар болгон **i** элементин табуу керек. Качан гана элемент табылганда, программа `True` деп, `break` командасын колдонуп циклден чыгуусу керек. Эгерде изделген элемент жок болсо – `False`ты чыгаруу керек.

```
a = [4, 15, -3, 11, -10, 9]
x = int(input('Изделүүчү элемент: '))
for i in a:              #a тизмесинин элементтерин иргөө
    if i == x:           #эгер элемент табылса
```

```
            print (True)
            break           #циклден чыгуу
    else:
        print (False)
```

Бул маселени дагы бир жол менен чыгарса болот. Ал үчүн тизмеде **x**ке барабар биринчи табылган элементтин индексин кайтарган **index** функциясын колдонуу керек:

```
a = [4, 15, -3, 11, -10, 9] #массивдин мисалы
print (a.index (-10))       #жыйынтык 4
```

Эгерде массивде изделген элемент жок болсо, программа катаны кайтарат жана ишти токтотот.

2-маселе. Тизмедеги максималдуу элементти табуу алгоритми мындай:

а) максималдуу элемент деп циклдин башында 0гө барабар болгон **mum** өзгөрмөсү алынат;

б) кезеги менен тизмедеги бардык элементтер каралып чыгат;

в) эгерде кезектеги элемент **mum** маанисинен чоң болсо, анда ушул эле өзгөрмөгө жаңы маани жазылат;

г) андай болбогон учурда максимумдун мааниси мурункудай эле калтырылат.

Программаны жазалы:

```
a = [4, 15, -3, 11, -10, 9]
mum = a[0]
for i in range(1, len(a)): #бардык тизмени карап чыгабыз
    if a[i] > mum:
        mum = a[i]
print(mum)    #жыйынтык 15
```

Минималдык элементти издөө программасы салыштыруу белгисинен гана айырмаланат:

```
a = [4, 15, -3, 11, -10, 9]
mum = a[0]
for i in range(1, len(a)):
    if a[i] < mum:    # < белгиси коюлат
        mum = a[i]
print(mum) #жыйынтык -10
```

Максималдык жана минималдык элементтерди издөө аракеттери көп колдонулгандыктан, Python тилинде тиешелүү **max()** жана **min()** камтылган функциялары бар:

```
a = [4, 15, -3, 11, -10, 9]
maximum = max (a)
minimum = min (a)
print (maximum)
print (minimum)
>>>
15
-10
```

Тизменин реверси

Тизменин элементтерин тескери иретте жайгаштыруу тизменин реверси деп аталат. Алгоритмдин принциби мындай: биз 0-элементти акыркы менен, 1-ни акыркыдан мурунку менен ж.б. алмаштырабыз. Ошентип, мындай алмаштыруунун саны тизменин узундугунун жарымына барабар болот.

Python тилинде индекстерди номерлөө 0дөн башталат. Ошондуктан, эгерде элементтердин жалпы саны N болсо, анда i үчүн каршысындагы элемент төмөнкү формула менен аныкталат: **N - i - 1**. Мисалы (төмөнкү таблицаны карагыла), 2 индексиндеги элементти алмаштыра турган элементтин индексин табуу үчүн, төмөнкү формула менен чыгарабыз:

N i
6 – 2 – 1 = 3

Демек, 2 индексиндеги элемент 3 индексиндеги элемент менен алмашат.

Массивдин элементтери	4	15	-3	11	-10	9
Индекстер	0	1	2	3	4 же (n -2)	5 же (n -1)

Элементтин экинчи түгөйүн табуу үчүн, биз тизменин биринчи жарымындагы эле индекстерди карайбыз. Бул болсо циклди тизменин ортосунан токтотуу керек дегенди түшүндүрөт:

```
a = [4, 15, -3, 11, -10, 9]
n = len(a)  #тизмедеги элементтердин санын эсептейбиз
for i in range(n//2):#тизменин 1-жарымынын индекстери
    a[i], a[n-i-1] = a[n-i-1], a[i] #элементтердин ордун алмаштырабыз
```

```
print(a)
>>>
[9, -10, 11, -3, 15, 4]
```

Тизмени реверстөө амалын стандарттык **reverse()** методунун жардамы менен да аткарса болот:

```
a.reverse ()
```

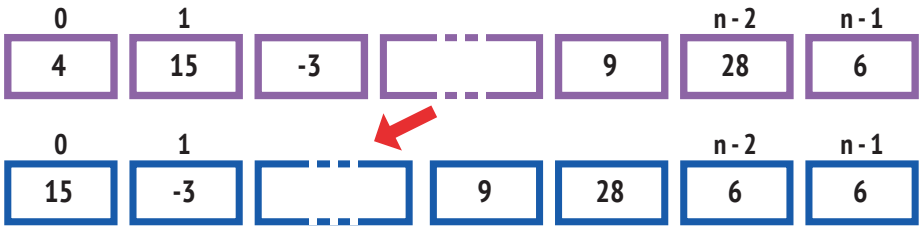
ЭСИҢЕ ТУТ

Python тилинде индекстердин маанилери терс болушу да мүмкүн. Бул учурда номерлөө аягынан баштап жүрөт. Мисалы, тизменин эң акыркы элементинин индекси -1, андан беркисиники -2 ж.б.

Элементтерди жылдыруу

Тизменин бир элементин өчүрүүдө анын оң жагында жайгашкан бардык элементтер бир уячага сол жакка жылат, б.а. **a[1]** элементи **a[0]** элементтин ордуна, **a[2] - a[1]** элементтин ордуна ж.б.у.с. Жаңы элементти коюуда андан кийинки турган бардык элементтер оңго жылат.

Төмөнкү схемада көрүнүп тургандай, 1-элементти өчүрүүдө, тизменин элементтеринин жалпы саны өзгөрбөйт, анткени акыркы элемент кайталанып акыркы эки орунду ээлейт.



Алгоритмин жазалы:

```
a = [4, 15, -3, 11, -10, 9]
n = len(a)
for i in range (n-1):
    a[i] = a[i+1]
print(a)
>>>
[15, -3, 11, -10, 9, 9]
```

Көңүл бурсаңар мында тизменин чегинен чыгып кетпеши, б.а. жок элемент a=[n] ге кайрылбашы үчүн цикл a=[n-1] де аяктап жатат.

Бирок биринчи элементтин мурунку маанисин акыркынын ордуна жазып койсо деле болот. Мындай жылдыруу циклдик деп аталат. Ал үчүн алдын ала (цикл башталганга чейин) биринчи элементти **b** кошумча өзгөрмөсүнө сактап коюу керек, ал эми цикл аяктагандан кийин аны тизменин акыркы уячасына жазуу керек:

```
a = [4, 15, -3, 11, -10, 9]
n = len(a)
b = a[0]
for i in range(n-1):
    a[i] = a[i+1]
a[n-1] = b
print(a)
>>>
[15, -3, 11, -10, 9, 4]
```

Pythondун **append()** жана **pop()** камтылган функцияларын колдонуп жылышууну оңой аткарса болот:

```
a = [4, 15, -3, 11, -10, 9]
a.append(a.pop(0))
print(a) #жыйынтык [15, -3, 11, -10, 9, 4]
```

Мында **pop(0)** методу өчүргөн 0чү индекстеги элемент **append()** методунун жардамы менен тизменин аягына кошулуп калды.

Критерий боюнча элементтерди тандоо

Тизмеден кандайдыр бир белгилүү шартка жооп берген элементтерди тандап алуу үчүн **b** жаңы тизмесин түзүп, аларды ошол жакка чогултуу керек. Шарты менен генератордун жардамында **a** тизмесинен жуп сандарды тандоо үчүн кодду жазалы:

```
a = [4, 15, -3, 11, -10, 9]
b = [i for i in a if i%2 == 0]
print(b)
```

a тизмесиндеги элементтерди **b** тизмесине тандап алуу үчүн бизге белгилүү **append()** методун колдонууга болот. Бул учурда берилген шартка жооп берген элементтер жаңы **b** тизмесине кошулат. Программаны мындай жазса болот:

```
a = [4, 15, -3, 11, -10, 9]
b = []
for x in a:
    if x % 2 == 0:
        b.append(x)
print(b)
>>>
[4, -10]
```

Тизмени модификациялоо

Тизмелер менен иштөө үчүн жана андагы берилиштерди өзгөртүү үчүн, Python тилинде көптөгөн ар кандай функциялар бар, мисалы:

ФУНКЦИЯ	МААНИСИ	МИСАЛ
print (a)	а тизмесин экранга чыгарат	a = [16, 'b', 34, 'c']#бардык мисалдар үчүн базалык тизме print (a) >>> [16, 'b', 34, 'c']
append ()	Тизменин артына бир элемент кошот	a.append (18) print (a) >>> [16, 'b', 34, 'c', 18]
clear ()	Тизменин бардык элементин өчүрөт	a.clear () print (a) >>> []
count ()	Берилген мааниси менен элементтердин санын кайтарат	print (a.count (16))#тизмеде мааниси 16га барабар канча элемент бар экендигин эсептейт >>> 1
extend ()	Базалык тизменин аягына башка тизмени кошот	b = [18, 'h'] #экинчи тизме a.extend (b) print (a) >>> [16, 'b', 34, 'c', 18, 'h']
index ()	Биринчи табылган окшош элементтин индексинин номерин кайтарат	print (a.index (34)) >>> 2
insert ()	Элементтерди индекси боюнча коюп чыгат	a.insert (1, 22) #индексти эсептөө 0дөн башталгандыктан, базалык тизмеде 1 индексинде 'b' турат, демек анын ордуна 22 цифрасы коюлуп, калгандары оңго жылат. print (a) >>> [16, 22, 'b', 34, 'c']

pop ()	Берилген индекстеги элементти өчүрөт	a.pop (0) #0 индексиндеги маанини өчүрүү
		print (a)
		>>> ['b', 34, 'c']
		
		a.pop ()
		print (a)
		>>> [16,'b', 34] #эгерде мааниле-ри берилбесе, анда акыркы элемент өчүрүлөт

remove ()	Берилген мааниси менен 1-элементти өчүрөт, эгерде элемент табылбаса ValueError билдирүүсү чыгат	`a.remove (34)`
		`print (a)`
		`>>> [16, 'b', 'c']`

reverse ()	Тизменин элементтерин тескери иретте жайгаштырат	`a.reverse ()`
		`print (a)`
		`>>> ['c', 34, 'b', 16]`

sort ()	Тизмени сорттойт (бир типтеги элементтүү тизме үчүн гана)	`a = [16, 8, 34, 3] #тизмеде жалаң бүтүн (int) сандар`
		`a.sort ()`
		`print (a)`
		`>>> [3, 8, 16, 34] #өсүү тартибинде сорттойт`
		
		`a = ['m', 'b', 'o', 'c'] # тизмеде жалаң символдор (str)`
		`a.sort ()`
		`print (a)`
		`#алфавит боюнча сорттойт`
		`>>> ['b', 'c', 'm', 'o']`

Саптар методдорунан айырмаланып (109-б.) тизмелер методдору тизменин өзүн өзгөртөт. Андыктан, жыйынтыкты жазуу үчүн жаңы тизмени түзүүнүн кажети жок.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Тизмени (0, 20) интервалындагы кокустук сандар менен толтургула.* ***x*** *санын киргизгиле жана* ***x****ке барабар бардык маанилерди тапкыла.*

2) n *элементтен турган сан маанисиндеги бир өлчөмдүү массив берилген. Массивдин элементтерин оң жакка айланма жылдырууну аткар, б.а. a(0) --> a(1); a(1) --> a(2); ... a(n-1) --> a(0).*

3) *Кокустук бүтүн сандардын тизмеси берилген. Тизменин бардык так сандарын нөлдөр менен алмаштыргыла жана алардын санын чыгаргыла.*

4) *Тизмени кокустук сандар менен толтургула жана тизменин* ***x*** *жана* ***y*** *индекстери менен белгиленген үзүмдөгү элементтери үчүн (бул элементтерди кошуу менен) реверс жүргүзгүлө.*

3.3-тема:

Тизмелерди сорттоо

Көпчүлүк учурда керектүү маалыматты издөөнү жеңилдетүү үчүн биз сорттоону колдонобуз. Мисалы, сөздүктө алфавит боюнча сөздөрдү сорттоо издөөнү жеңилдетет.

Pythondогу тизмелерди **сорттоо** – бул анын элементтерин берилген иретте жайгаштыруу болуп саналат.

Сорттоо ирети ар кандай болушу мүмкүн: сандар үчүн адатта маанилеринин өсүү (кемүү) тартибинде сорттоо каралат. Мисалы, [3 1 0 5 2 7] тизмесин өсүү тартибинде сорттоодо [0 1 2 3 5 7] тизмеси алынат. Символдук берилиштер адатта алфавиттик тартипте сорттолот.

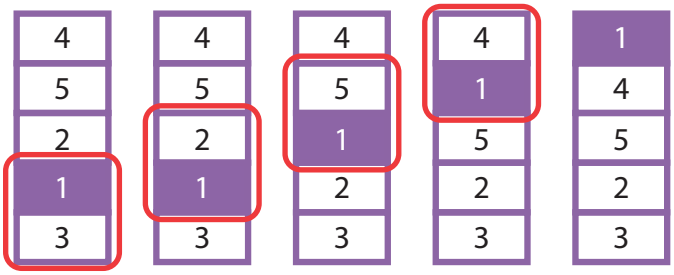
Берилиштерди уюштуруу үчүн сорттоо алгоритмдеринин көптөгөн түрлөрү бар. Сорттоо кантип иштей тургандыгын түшүнүү үчүн алардын кээ бирлерин карап көрөлү.

Көбүкчөлүү сорттоо (алмаштыруу методу)

Тизмени бул ыкма менен сорттоо үчүн, тизмеден солдон оңду көздөй коңшу элементтерди экиден салыштырып өтүү керек. Качан сол жактагы элемент оң жактагысынан «оор» болуп калса, анда алардын ордун алмаштыруу керек. Ошентип эң чон элементтер тизменин аягына түшүп, ал эми эң «женил» элементтер тизменин үстүнө (тизменин башына) көбүкчөлөргө окшоп көтөрүлүп калат. Ушундан улам алгоритмдин аты келип чыккан.

Эгерде бул алгоритмдин аракеттеринин иретин жазса ал төмөнкүчө болот:

1. Тизменин алгачкы эки элементин салыштырабыз. Эгерде 1-элемент 2-элементтен чоң болсо, алардын ордун алмаштырабыз. Эгерде алар керектүү иретте турса, анда ордунда калтырабыз.



2. Экинчи элементтердин жубуна өтөлү: алардын маанисин салыштырабыз жана керек болгон учурда ордун алмаштырабыз.

3. Тизменин аягына жеткенде процессти кайра башынан баштайбыз. Бирок, эң чоң элемент тизменин аягында тургандыктан, салыштыруулардын саны 1ге азаят. Бул процесс тизмедеги акыркы элементтердин жубу каралганча кайталана берет.

Көбүкчө методу боюнча толук программаны жазып көрөлү. Мында:

- **i** өзгөрмөсү минималдык элемент жазылган уячанын индексин сактайт. Алгач бул биринчи уяча болот.

- **j** өзгөрмөсү учурда каралып жаткан уячаны билдирет.

```
from random import randint
n = 10 #тизмени 10 кокустук сан менен толтурабыз
a = [randint (1,99) for n in range (n)]
print (a)
for i in range (n-1): #тизмеден өтүүлөрдүн саны
    for j in range (n-i-1): #салыштыруулардын саны iге азаят
        if a [j] > a [j+1]: #j жана j+1 элементтерин салыштырат
            a[j], a [j+1] = a[j+1], a[j] #керек болсо эле-
менттердин ордун алмаштырат
print(a)
>>>
[5, 84, 90, 37, 30, 32, 29, 62, 17, 99]
[5, 17, 29, 30, 32, 37, 62, 84, 90, 99]
```

Биз көрүп тургандай, мында эки цикл иштейт: сырткы жана ички. Сырткы циклдин өтүү саны алмашуулардын санына барабар деп берилет. Ал эми алмашуулардын саны элементтердин санынан 1ге кем. Анткени, мисалга алсак 2 элементи бар тизмеде бир эле алмашуу болот, 3 элементи бар тизмеде 2 алмашуу ж.б.

Ички циклдин өтүү саны ар бир өтүүдө 1ге азайып турат, анткени бир жолу өтүүдө бир элемент өзүнүн ордуна (тизменин башына) туруп калат жана биз ага тийишпейбиз.

Бул сорттоо алгоритми өтө жай иштейт, анткени ал тизмеде канча элемент болсо ошончо жолу элементтерди баштан аяк иргеп чыгуусу керек. Эгерде тизмеде 100 элемент болсо, анда программа 100 жолу башынан аягына чейин өтүп чыгышы керек. «Көбүкчөлүү» сорттоо алгоритми окутууда гана колдонулуп практикада дээрлик колдонулбайт: массивдин аягында турган кичинекей элементтер (аларды «ташбакалар» деп да аташат) тизменин башына өтө узак убакытта жетишет.

Тандоо менен сорттоо ыкмалары

Бул алгоритм тизмени эки бөлүккө бөлөт: сорттолгон жана сорттолбогон элементтери менен. Алгач бардык тизме сорттолбогон деп эсептелет. Сорттолбогон бөлүктөн эң кичине элементи тандалып алынат да биринчи менен орун алмашат. Эми биринчи элемент – бул тизменин сорттолгон бөлүгү болуп калат. Тизменин биринчи элементи сорттолгон болгондуктан, биз эң кичине элементти калган элементтерден издейбиз да аны экинчи элемент менен алмаштырабыз. Ушундай жол менен сорттолгон бөлүгү өсүп, ал эми сорттолбогон бөлүгү азайып олтурат. Бул процесс тизмеде акыркы элемент сорттолгонго чейин улантылат.

Тизмени тандоо методу боюнча сорттоонун алгоритмин жазалы. Мында **i** өзгөрмөсү минималдык элемент жазылган уячанын индексин, ал эми **j** өзгөрмөсү каралып жаткан элемент жазылган уячанын индексин сактайт.

```
for i in range(n-1):
    n_min = i
    for j in range(i+1,n):
        if a[j] < a[n_min]:
            n_min = j
    if i != n_min:
        a[i], a[n_min] = a[n_min], a[i]
```

Бул жерде табылган минималдык элемент өзүнүн ордунда турбаса б.а. **i != n_min** болгондо гана орун алмашуу жүрөт. Минималдык элементтин номерин издөө циклде аткарылгандыктан, бул сорттоо алгоритми да камтылган циклди түзөт.

Тез сорттоо (Quick Sort)

Көбүкчөлүү жана тандоо менен сорттоо методдору чоң берилиштеги массивдер үчүн жай иштейт. Ошондуктан чоң массивдерди сорттоо үчүн атайын «тез сорттоонун» рекурсивдүү алгоритми колдонулат (англ. quicksort).

Бул сорттоонун идеясы мындай: алгач массив болжол менен ортосунан тең экиге бөлүнөт. Бөлүнгөн жерден бир элемент тандалып, ал «таяныч» элемент катары каралат. Андан соң сол жак бөлүгүндөгү элементтерден «таяныч» элементтен чоң же ага барабар элементтер тандалып алынат жана оң бөлүгүнө орун алмаштырылат. Андан ары оң жагы жана сол жагы өзүнчө массивдер катары каралат да кайрадан экиге бөлүнөт.

Ошентип баштапкы тизмени сорттоо анын ичиндеги эки тизмечени сорттоого алып келди. Эми алгоритмдин баштапкы кадамдары эки жаңы тизмечелерге өзүнчө кайрадан (рекурсивдүү) колдонулат.

Биз айтып кеткендей сорттоонун ылдамдыгы чоң массивдер менен иштөөдө маанилүү. Төмөндөгү таблицада кокустук маанилер менен толтурулган ар кандай өлчөмдөгү массивдердин биз караган алгоритмдерди колдонуп сорттоо убактысы (секунда менен) көрсөтүлгөн.

№	Көбүкчөлүү сорттоо	Тандоо менен сорттоо	Тез сорттоо
1000	0,09 с	0,05 с	0,002 с
5000	2,4 с	1,2 с	0,014 с
15000	22 с	11 с	0,046 с

Таблицадан көрүнүп тургандай, мисалы 15 миң элементи менен массивди тез сорттоо көбүкчө методуна караганда 500 эсе тез иштейт экен.

Стандарттуу сорттоо

Python тилинде көптөгөн берилиш тибин сорттоодо оңой иш алып барган timsort аттуу камтылган алгоритм бар. Мындай сорттоону чакыруунун эки ыкмасы бар:

1. **sort** методу:

```
a = [5, 2, 3, 1, 4]
a.sort()
print (a)
```

Мындай ыкмада тизменин элементтери берилген **a** тизмесинин ичинде сорттолот, б.а. тизменин өзү өзгөрөт, sort методу маанилерди кайтарбайт, ошондуктан, sort методунун чакыруу жыйынтыгын арифметикалык туюнтмаларда же жыйынтыкты чыгарууда колдонууга болбойт.

2. **sorted** функциясы:

Методдон айырмаланып sorted функциясы ага берилген тизмени өзгөртпөйт, ал бардык өзгөртүүлөрдү жаңы тизмеге киргизет. sorted функциясын мындай колдонсо болот:

```
a = [5, 2, 3, 1, 4]
b = sorted(a)
print (b)
```

sorted функциясын берилиштерди киргизүүдө же чыгарууда жазса болот. Төмөндө колдонуучу тараптан бош орун аркылуу киргизилген берилиштерди сорттоп жана тизме катары чыгарып бере турган бир саптуу программа жазылган:

```
print(''.join(map(str, sorted(map(int, input().split())))))
```

Адатта сорттоо өсүү тартибинде же тагыраак айтканда, ар бир кийинки элемент мурункусунан чоң же барабар болушу керек. Массивди кемүү тартибинде сорттоо үчүн (ар бир кийинки элемент мурункусунан кичине же ага барабар) сорттоо функциясына reverse = True деп көрсөтүш керек:

```
b = sorted (a, reverse = True) #же
a.sort (reverse = True)
```

БУЛ КЫЗЫКТУУ!

Timsort – бул сорттонун гибриддик алгоритми, б.а. ал өзүнө эки башка түрдүү сорттоону камтыйт. Реалдуу дүйнөдө берилиштердин массиви көп учурда иреттүү берилгендиги бул алгоритмдин негизги идеясын түзөт. Мындай берилиштерде **timsort** сортоо алгоритмдеринин көбүнө караганда кыйла тезирээк иштейт.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

***1) n** элементтен турган бир өлчөмдүү сандык маанидеги массив берилген. Баштапкы массивдин элементтеринен эки жаңы массив түзгүлө. Биринчисине 3кө бөлүнгөн гана сандар, ал эми экинчисине 5ке бөлүнгөн гана сандар кирсин.*

***2)** Биринчи суроодогу программаны уланткыла жана 3кө бөлүнгөн сандарды өсүү тартибинде, ал эми 5ке бөлүнгөн сандарды кемүү тартибинде жайгаштыра тургандай кылып программаны толуктагыла.*

***3)** Бүтүн сандар менен тизме берилген. Тизмедеги эң кичине жана эң чоң сандын көбөйтүндүсүнүн жыйынтыгын чыгаруучу программаны түзгүлө.*

***4)** Колдонуучудан сапка бир нече сөз киргизүүсүн сураган программаны жазгыла. Сөздөрдү алардын узундугунун өсүү тартиби боюнча сорттогула.*

3.4-тема:

Матрицалар

Матрица – бул таблицалык структурага ээ болгон эки өлчөмдүү массив. Эки өлчөмдүү массивдер деле бир өлчөмдүү массивдердей баяндалат. Айырмасы эки өлчөмдүү массивдин элементинде эки координата (эки индекс) – элемент жайгашкан саптын жана мамычанын номерлери бар.

Python тилинде таблицалар менен иштөө үчүн тизмелерди колдонушат. Эки өлчөмдүү таблица – бул ар бир элементи тизме болуп сакталган тизме («тизмечелердин тизмеси»). Мисал катары 3 саптан жана 3 мамычадан турган таблицаны карайлы:

Таблицадагы берилиштер Pythonдо төмөнкүдөй жазылат:

```
a = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

Бул тизмени бир сапка да жазса болот:

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]].
```

Бул тизмени экранга чыгарууда эки камтылган циклди колдонушат. 1-цикл саптын санын аныктаса, 2-цикл анын ичиндеги элементтерди терип чыгат. Элементтери үтүр менен эмес, бош орун менен ажыратылган ар бир тизмечени жаңы саптан чыгарган программаны жазалы:

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for i in range ( len(a) ):
    for j in range ( len(a[i]) ):
        print (a[i][j], end = '' )
    print ()
```

```
>>>
1 2 3
4 5 6
7 8 9
```

Мында **i** – камтылган тизменин индекси (саптын номерин аныктайт), ал эми **j** – камтылган тизменин ичиндеги элементтин индекси (мамычанын номерин аныктайт); **len(a**) – бул чоң тизмедеги камтылган тизмелердин саны (бул жерде алар 3), **len(a[i]**) – мамычалардын саны менен дал келген камтылган тизмедеги элементтердин саны.

a [i] [j] – бул j индекси менен i камтылган тизмесиндеги элемент:

`a[0][0]==-1, a[0][1]==0, a[0][2]==1, a[1][0]==-1,` ж.у.с.

Бул мисалды мындай жазса да болот:

```
for row in a:      #a сабында
    for elem in row:  #саптагы элемент үчүн
        print(elem, end=' ') #элементтерди чыгар
    print()
```

1-маселе. Матрицаны кокустук сандар менен толтурабыз. Саптын жана мамычанын санын клавиатурадан киргизебиз.

Индекстер экөө болгондуктан, матрицаны толтуруу үчүн камтылган циклди колдонобуз. Мындан ары **n** сабынан жана **m** мамычасынан турган **a** матрицасы бар деп, ал эми **i** жана **j** – саптын жана мамычанын индексин билдирген бүтүн сандуу өзгөрмөлөр деп эсептейли. Ар бир тизмече **append()** методунун жардамында кокустук сандар менен толтурулат:

```
import random
n = int(input ('Саптын санын киргизгиле: '))
m = int(input ('Мамычанын санын киргизгиле: '))
a = []
for i in range (n):
    a.append([])
    for j in range (m):
        a[i].append (random.randint (10,40)) #ар бир эле-
ментке 10дон 40ка чейинки кокустук сан ыйгарылат
for i in a: #ар бир камтылган тизме жаңы саптан чыгарылат
    print (i) #бирок чарчы кашаада
>>>
[33,16,31,33] #n=2,m=4 болгондогу кокустук сандар
[39, 35, 11, 15]
```

Ушул эле маселенин жыйынтыгы кашаалары жок чыксын десек:

```
import random
n = int(input ('Саптын санын киргизгиле: '))
m = int(input ('Мамычанын санын киргизгиле: '))
a = [[random.randint(10, 40) for i in range(m)] for j in
range(n)]
for row in a:
    print(' '.join([str(elem) for elem in row])) #бардык
элементтер биригишет,ортосунда бош орун аркылуу тизмектелет
```

Эки өлчөмдүү массивди иштетүү

Матрицанын бардык элементтерин иргеп чыгуу үчүн дал ушундай эле камтылган эки циклди колдонуу керек. Биринчи цикл саптын номерлерин иргейт, экинчи цикл болсо саптын ичиндеги элементтерди иргейт. Мисалы, бардык элементтердин суммасы (s) мындай эсептелет:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s) #жыйынтыгы 45
```

Бул жазуулар үчүн даяр функция **sum**ду колдонсо да болот:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    s += sum(row)
print (s)
```

Матрицанын кээ бир элементтерин иштетүүнү мисалда карап көрөлү.

2-маселе. n саптан жана **m** мамычадан турган квадраттык массив берилсин дейли. Негизги диагоналды 1лер менен, анын сол жагындагы аймакты 2лер менен, ал эми оң жагындагы аймакты 0дөр менен толтуруу керек. Негизги диагональ – бул a[0,0], a[1,1], ..., a[n-1,n-1] элементтери, б.а. саптын номери мамычанын номерине барабар. Негизги диагоналды бирлер менен толтуруш үчүн бир цикл керек:

```
for i in range(n):      #A[i][i] менен иштейбиз
    a[i][i] = 1         #аны бирлер менен толтурабыз
```

Диагоналдан оң жактагы элементтерди 0дөр менен толтурушубуз керек. Ал үчүн **i** индексиндеги ар бир саптагы a[i][j] элементтерине j=i+1, ..., n-1 үчүн 0 маанисин ыйгарабыз. Мында бизге камтылган цикл керек болот:

```
for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0
```

Ушундай жол менен эле j=0,..., i-1 үчүн a[i][j] элементине 2 маанисин ыйгарабыз:

```
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
```

Эми циклдерди чогултсак, мындай чыгарылышты алабыз:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) *Эки өлчөмдүү массив берилген. Экранга төмөнкүлөрдү чыгаргыла: а) экинчи саптагы бардык элементтерди; б) экинчи мамычадагы бардык элементтерди.*

2) *Эки өлчөмдүү массив берилген. Массивдин каалагандай эки элементинин ордуларын алмаштырган программаны жазгыла.*

3) ***n*** *саптан жана* ***m*** *мамычадан турган тик бурчтуу* ***a*** *матрицасын кокустук сандар менен толтургула. Массивдин элементтеринин орточо арифметикалык маанисин тапкыла.*

4) *Матрицанын ар бир сабы үчүн табылган орточо маанилеринин эң чоңун аныктагыла.*

4 - бөлүм

Компьютердик тармактар жана интернет

4.1-тема:

Келечектин технологиялары

Интернет технологиялардын өнүгүшү дүйнөнүн артка кайткыс өзгөрүүсүнө алып келди. Компьютерлер жана девайстар учурдун ажырагыс бөлүгү болуп, бул түзүлүштөр бизди келечекте да жандап жүрө тургандыгы шексиз.

БУЛ КЫЗЫКТУУ

Футуролог (келечекти айтуучу), ойлоп табуучу, технологиялардын өнүгүшүн изилдөөчү жана Googledун машиналык окутуу тармагынын техникалык директору ***Рэй Курцвейл*** бир нече жылдан бери технологиялык божомолдорун жарыялап турат. Анын көп айтуулары чындыкка айланды. Мисалы, ал шахматтык партияда дүйнөдөгү белгилүү шахматчыны компьютер жеңе алат деп, ошондой эле кошумчаланган жана виртуалдык реалдуулук жөнүндө да алдын-ала айткан.

Бул темада биз «секирик» деп аталган технологиялары (англ. Disruptive technologies) жөнүндө айтқыбыз келип турат. Алар кийинки жылдарда биздин дүйнө жөнүндөгү көз карашыбызды түп-тамырынан бери өзгөртөт.

1 Жасалма интеллект

Жасалма интеллект *(ЖИ, англ. Artificial intelligence, AI)* – бул адамдын мээсинин когнитивдүү функцияларын көчүргөн, компьютерлерге жана машиналарга гана тиешелүү болгон интеллекттин өзгөчө тиби. ЖИ адамдын чечимдерди кантип кабыл ала тургандыгын түшүнүүгө багытталган.

Интеллектуалдык компьютердик программалардын башка программалардан болгон эң негизги артыкчылыгы – бул алардын өз алдынча үйрөнүүгө жөндөмдүүлүгү жана өзүнүн ичиндеги каталарды оңдоо мүмкүнчүлүгү. ЖИ адам менен жана башка программалар менен канчалык көбүрөөк аракеттенишсе, ал ошончолук көп маалыматты сактап алат жана ал анын «жашоо тажрыйбасынын» бир бөлүгү болуп калат.

ЖИ технологиялары учурда нейрондук тармак жана булуттук эсептөөлөрдүн кеңири таралышынын негизинде интенсивдүү өнүгүп жатат. Курцвейлдин божомолдору боюнча 2030-жылы ЖИ толугу менен адам сыяктуу ойлонууга жөндөмдүү болот.

2 Интернет буюмдар (*англ.* ***Internet of thins, IoT***)

Компьютерлер ушунчалык кичирейтилди, эми аларды кийимдерге кошуп тигүүгө, тиш щёткага, саатка, лампага орнотуп койсо болот. Биз буюмдарыбызды аралыктан – интернет аркылуу башкарууга мүмкүндүк алдык. Учурда ар бир девайс жана кийим бири-бири менен аракеттенишип тургандай өзүнүн тармактагы IP-дарегине ээ.

Ден соолугубузга кам көргөн «акылдуу приборлор» боло турган сасык тумоонун алдын алышы мүмкүн. Атайын билдиргич менен камсыз болгон ар бир троллейбусту жана автобусту эми шаардын интернет-картасынан көрүүгө болот. «Акылдуу» көчө чырактары күндүн жарыгынын деңгээли төмөндөгөндө автоматтык түрдө жанат, ал эми жол чырактын жашыл жарыгы жолдун кайсы тарабында автомобилдердин агымы көп болсо ошол тарапка көбүрөөк күйөт.

Бир эле машиналар эле эмес бүткүл шаар, а түгүл өлкө да акылдуу боло баштады. Силер мүмкүн фильмдерден полиция кызматкери бир баскычты басып – өлкөдөгү каалаган видеобайкоо камераларына кошулгандыгын көргөнсүңөр. Ушул бардык артыкчылыктарга карабастан, өзүңөрдүн ар бир девайсыңарды интернетке кошордон мурда ойлонгула, силер чындап эле бардык жерде өзүңөрдүн санариптик изиңерди калтыргыңар келеби?

3 Робот техникасы

Роботтор көптөн бери эле биздин арабызда. Албетте, жасалма интеллект менен камсыз болуп алар толук бойдон адамдын түспөлүндөй болбосо да, кайсы бир учурда адамдарга гана тиешелүү ролдорду аткарууда. Футурологдордун божомолдору боюнча жакынкы 10 жылда роботтор биздин үйүбүздө муздаткыч, кир жуугуч машина сыяктуу эле кадимки буюмдардын катарына кирет.

Роботтор адамдардын бөлүгү боло башташты. Мисалы, роботтоштурулган же «бионикалык» деп аталган протездерди адамдын дене мүчөсү кылып өөрчүтүү. Учурда мындай протездерди башкаруу үчүн жакын жайгашкан булчуңдардан алынган сигналдарды окуучу атайын системалар колдонулат.

Ал эми мындай протездин ээси түзүлүшү менен тийген объекттин касиеттери (ысык, бодуракай ж.б.) жөнүндө кайтарым маалыматты түздөн-түз мээге ала турган мезгил алыс эмес. Эң жөнөкөй протездер азыр жеткиликтүү, аларды оңой эле 3D принтерде басып чыгарып алса болот.

4 3D басып чыгаруу

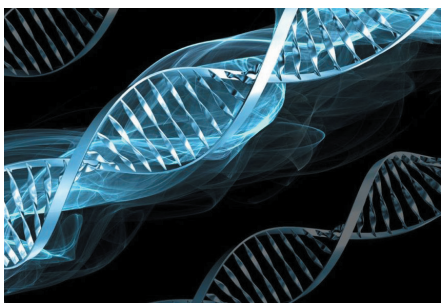
Эгерде биз каалагандай нерсени, анын ичинде автомобилди да принтерде басып чыгарып алсак эмне болот эле? Муну элестетүү кыйын, бирок бул чындап эле болуп жатат: адамдар өздөрүнө үйдү, эмеректерди, идиш-аяктарды, транспортту, кийимди, дененин бөлүктөрүн, ал эмес тамакты да басып чыгарып калды, анткени кондитердик 3D принтерлер пайда болду! Элестеткиле, силер алманы принтерден басып чыгардыңар, бирок картриджде шекер түгөнгөндүктөн аныңар ачуу болуп калгандыгын.

3D басып чыгаруунун өнүгүшү менен товарлары бар ири контейнерлерди ар жактан ташып келүүнүн кажети жок болот. Болгону керектүү товардын чиймеси менен файлды жөнөтүү гана жетиштүү болот. Мүмкүн ушунун негизинде биз зыяндуу өндүрүштү, глобалдуу транспорттук системанын натыйжасында болуп жаткан абанын булганышын токтото аларбыз.

5 Биотехнология жана гендик инженерия

Окуу китебин жазып жаткан учурда ар кандай жигердүү биотехнологияларды иштеп чыгуу боюнча дүйнөдө миңдеген медициналык сыноолор өтүп жатат. Жаңы вакциналар иштелип чыгып жатат, CRISPR технологиясын колдонуу менен гендерди редакциялоо (биологиялык организмдердин генотиптерин жасалма жана максаттуу өзгөртүү) боюнча клиникалык сыноолор жүргүзүлүп жатат. Гендик инженериянын негизги ыкмасы – тиешелүү гендерди бөлүп алуу жана аларды модификациялоо: мутацияланган гендерди оңдоо, жоголгон гендерди калыбына келтирүү ж.б. Бул иммундук системасы бузулгандагы, кандын уюшу, онкология менен байланышкан ооруларды дарылоодо өзгөчө актуалдуу.

Экинчи өтө тездик менен өнүгүп жаткан багыт – бул наномедицина. Ал наноматериалдарды иштеп чыгуудан башталып, наноэлектрондук биосенсорлор менен бүтөт. Мисалы, организмдин клеткасына чейин кирип, аларды «тамактандырып» жана ашыкчаларын жок кылуучу нанороботтор иштелип чыгууда.



6 Виртуалдуу реалдуулук (ВР)

2030-жылдын аягына чейин ВР ушунчалык жогорку сапатка жеткендиктен, аны чыныгы реалдуулуктан айырмалоо мүмкүн болбой калат. Интернет тармагынын өнүгүшү бир убакта бир нече колдонуучулар колдоно ала тургандай виртуалдуу дүйнөнү курууга мүмкүндүк берет.

Азыркы күндө ВР бир эле оюн-зоок индустриясында эле эмес, автомобиль куруу өндүрүшүндө, аэрокосмостук өндүрүштө жана кеме курууларда кеңири колдонулууда. ВР технологиясы иштеп чыгуу убактысын кыскартып жана азырынча чыга элек продуктуну да сыноого мүмкүнчүлүк берет.

7 Калыбына келүүчү энергия жана жашыл технологиялар

Калыбына келүүчү же «жашыл» энергия – бул адамдык масштабда түгөнбөй турган булактардан алынган энергия. Эксперттердин ою боюнча, калыбына келүүчү энергиянын булактары климаттын өзгөрүүсүн жана планетабыздын булганышын кескин азайтат.

Калыбына келүүчү энергия шамал турбиналары, фотоэлектрдик элементтер, күн панелдери, геотермалдык энергия, океан толкундарынын энергиясы ж.б. ойлоп табуулардын негизинде колдонууга мүмкүн болууда.

Андан тышкары өзүнүн габариттерин сактоо менен энергияны чогултуучулардын кубаттуулугунун өсүшү да калыбына келүүчү энергиянын булактарына болгон суроо-талаптарды олуттуу өстүрүүдө.

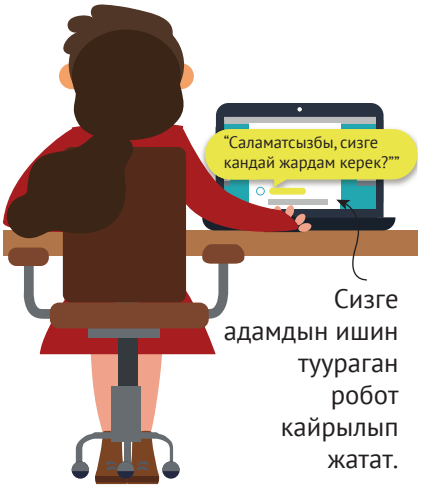
СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) *Кандай калыбына келүүчү энергиянын булактарын билесиңер? Силердин оюңарча «жашыл» технологияларды колдонуу эмне себептен биздин планета үчүн маанилүү?*
2) *Гендик модификациялоону этикага жатат деп эсептейсиңерби?*
3) *Силер 3D принтерлерден, ЖИ, виртуалдуу реалдуулуктан коркунучтарды көрө алдыңарбы? Бул технологиялар адамзат үчүн эмнеси менен коркунучтуу болушу мүмкүн?*

4.2-тема:

Санариптик дүйнөдөгү коопсуздук

Санариптик дүйнө же анын дагы бир аты болгон – кибермейкиндик – бул компьютерлер жана компьютердик тармактар, өзүнүн мыйзамдары жана эрежелери бар параллель аалам; чексиз мүмкүнчүлүктөрдү ачкан жана коркунучтарды камтыган мейкиндик.



Виртуалдык мейкиндикте жагымсыз окуялардын болбошу үчүн кээ бир эрежелерди сактоого туура келет.

Биринчиден, интернетке кирип жатканда эле – бул ачык жана көп кырдуу дүйнө жана биз ал жерде жалгыз эмес экендигибизди унутпашыбыз керек. Ар бир профилдин, аккаунттун артында тирүү адам, адамдардын тобу же программалык бот жайгашат.

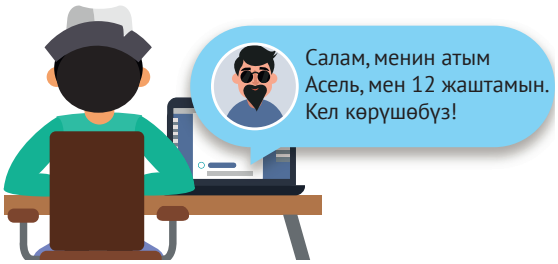
Интернетте иштеп жатып, бир эле боттор өздөрүн адамдар катарында көрсөтпөшүн унутпашыбыз керек. Адамдар да өздөрүн башка адамдардай көрсөтүшү мүмкүн.

Мындай адамдар ар түрдүү максаттарды көздөшү мүмкүн – сиздин аккаунтуңузду бузуудан (сиздин паролуңузду уурдоо, профилиңизди ар кандай максатта колдонуу) баштап, физикалык жактан аңдуу, шантаж жана опурталдуу аракеттерге тартууга чейин.

АНЫКТАМА

«Бот» («робот» сөзүнүн кыскартылышы) – адамдын ишин туураган программа. Чат-бот чаттагы маектешүүчүнү туурайт.

Эсиңерде болсун, силер интернетте бөлүшкөн бардык маалыматтар шылуундар тарабынын колдонулушу мүмкүн. Тармакта эмне жөнүндө сүйлөштү, кандай маалыматтарды жайгаштырууну жана кимди достукка кабыл алууну биринчи ойлонгула.

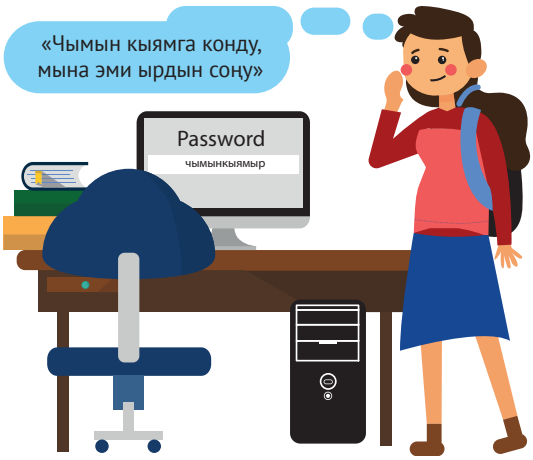


Биз башында айтып кеткендей интернетте өзүн башка адам кылып көрсөтүү үчүн өтө көп мүмкүнчүлүктөр бар. Бир эле адамдын бир учурда бир нече аккаунту болушу мүмкүн, бир жерде ал өспүрүм болсо, башкасында аскер кызматкери, мамлекеттик чиновник болот.

Эгерде досуңардан анын телефонуна акча салып коюуну суранган же белгиленген жерге жолугууга кел деген же башка бир күтүүсүз билдирүү келсе, аны аткаруудан мурда эң биринчи досуңузга байланышып, чындап эле ал жөнөткөнүн тактап алыңыз. Мүмкүн анын аккаунтун бузуп, анын атынан башкалар билдирүү жөнөтүшү мүмкүн.

Андан тышкары, сизге шилтеме боюнча өтүүнү же тиркелген документтерди ачууну (фото, видео) сунуштаган электрондук каттар келиши мүмкүн. Келген маалыматтар сизди кызыктырышы (сиз миллион утуп алдыңыз!) же коркутушу (биз сиздин сырыңызды бардыгына айтабыз!) мүмкүн. Мындай билдирүүлөр көпчүлүк учурда бир эле максатта келиши мүмкүн – вирус жуктуруу же аккаунтуңузду уурдоо. Андыктан, шылуундардын кайырмагына илинбеңиздер!

Биздин аккаунттар – бул биз, биздин жакындарыбыз жана биздин мамилебиз жөнүндө маалыматтар. Башка адамдардын колуна өтсө алар бир эле бизге эмес, биз менен байланышта болгондордун бардыгына зыян алып келиши мүмкүн. Мындай окуя болбош үчүн аккаунтту ишенимдүү пароль менен коргоо керек.



Ишенимдүү пароль ондон кем эмес, баш жана кичине тамгаларды, цифраларды жана атайын символдорду (тор белгиси, жылдызча, төмөнкү сызык ж.б.) өзүнө камтыйт.

Ишенимдүү паролдо өздүк маалымат (ат, фамилия, туулган датасы, үй жаныбарынын аты, кызыгуулар жана сиз жөнүндө социалдык тармактардан билип ала турган маалыматтар) камтылбашы керек.

Ишенимдүү паролду эстеп калуу албетте кыйын. Ал үчүн эстеп калуунун ар түрдүү технологиялары колдонулат. Мисалы, силер өзүңөргө гана түшүнүктүү болгон, логикалык жактан байланышкан үч-төрт сөздү курап алсаңар болот.

Андан да жакшысы бириктирилген сөздөрдү цифралар же символдор менен ажыратып койсоңор болот. Мисалы: ***чымын1кыям2ыр3***

Андан да сонуну: ***muh@1v@renie2stihotvorenie3***

Окуу китебинде берилген паролду эч качан колдонбогула! Өзүңөр ойлоп тапкыла!

Эч качан түрдүү сайттарда бир эле паролду колдонбогула!

Ар түрдүү паролдорду эстеп калууда кыйналбаш үчүн атайын программаларды – пароль менеджерлерин колдонууга болот.

Пароль менеджерлери паролду түзүүгө, сактоого жана колдонууга жардам берет. Мындай программалардын ичинен эң кеңири тараганы болуп KeePass жана LastPass эсептелет.

LastPass ✱✱✱✱ KeePass

Сиздин паролуңуз абдан ишенимдүү болгон күндө да, баары бир ал башкалардын колуна түшүп калуу ыктымалдуулугу бар. Мисалы, кимдир-бирөө сиздин паролду терип жатканыңызды көрүп калышы мүмкүн же аны фишинг аркылуу алып алышы мүмкүн.

Өзүңүздүн коопсуздугуңузду камсыз кылуу үчүн сиз интернетте макулдашкан жазылуу же жаңы кызмат тейлөөсүнүн шарттарын абдан көңүл бөлүп окуп чыгуу керек. Мисалы, мобилдик операторлордун сайтында сунушталган «жаңы рингтон» же «анекдоттор» деген сыяктуу типтеги ар түрдүү тейлөөлөргө жазылуу. Көпчүлүк учурда сиз «Окуу» деген чоң, кызыктыруучу баскычты көрөсүз жана ага бир басуу менен сизден абоненттик акы алып тура турган тейлөөгө жазылып каласыз.

Коргонуш үчүн ар дайым шартын (алар адатта майда шрифт менен жазылат) окуп чыккыла жана операторлордон келген шектүү тейлөө кызматтарына жазылуу жөнүндөгү СМС билдирүүлөрдү байкабай койбоңуз. Эгерде сиз өз алдыңызча жазылуудан баш тарта албай жатсаңыз, өзүңүздүн операторуңуздун байланыш салонуна кайрылып, тейлөө кызматын өчүртүп жана жаңы жазылууларды блоктоп (тосмолоп) салуу мүмкүнчүлүгүн сураныңыз.

Фишинг ***(fishing «балык кармоо, колго түшүрүү»)*** – бул башка бирөөлөрдүн аккаунтуна жана маалыматына уруксат алууга багытталаган тармактык алдамчылык. Көпчүлүк учурда бул сырткы келбети боюнча түп нускадан айырмасы жок болгон сайтты колдонуучуга сунуштоо менен ишке ашырылат. Ал жерде колдонуучуга пароль жана логинди киргизүү сунушталып, кийин пароль алдамчылардын колуна түшөт да колдонуучу өзүнүн аккаунтуна кире албай калат.

Колдонуучуларды аккаунтун жоготуп алуудан коргоо максатында азыркы сервистер эки факторлуу аутентификацияны колдонууну сунуш кылышат.

Эки факторлуу аутентификация – бул паролдон тышкары, телефондун жардамында аккаунтка кирүүгө мүмкүн кылган кошумча коргоо деңгээли. Анда колдонуучу телефонуна СМС билдирүү катары келген же телефондо атайын тиркеменин жардамында генерацияланган кодду да киргизүүсү керек болот.

Ошентип, кылмышкер сиздин аккаунтуңуздун паролун билген күндө да сиздин телефонуңузга кире албай туруп, аккаунтуңузду колдоно албайт.

Телефонуңузга да пароль коюуну унутпаңыз!

Ушул жөнөкөй эрежелерди сактоо менен сиз өзүңүздүн маалыматыңызды күтүүсүз чабуулдардан коргоп, өзүңүздү ишенимдүү сезе аласыз.

Бардык жогоруда айтылгандар - булар санариптик мейкиндикте коопсуздукту камсыз кылуу сунуштарынын бир гана бөлүгү. Буга антивирустук коргоо, маалыматтын резервдик көчүрмөлөрүн түзүү жана башка көптөгөн сунуштар да кирет.

Коопсуздуктун эң негизги эрежесин дагы унутпаңыз – ойлонгон адамды алдоо алда канча кыйынга турат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) «Вирустар кантип силердин компьютериңерге кирет» деген темада ойлонгула. Алардан кантип коргонуш керек?
2) Силердин санариптик маалыматыңарга болгон кандай коркунучтарды (вирустардан тышкары) айтып бере аласыңар?
3) Өзүңөр жана жакындарыңар жөнүндөгү кандай маалыматтарды интернетке жарыялоого болбойт жана эмне үчүн?

Тиркемелер

№1 тиркеме

Текстгин стандарттык коддоолору (кодировкалары) CP-1251:

Ѓ 128	ѓ 129	‚ 130	ѐ 131	„ 132	… 133	† 134	‡ 135	€ 136	‰ 137	Љ 138	‹ 139	и 140	И 141	Ќ 142	Ў 143
ѓ 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	ё 152	™ 153	ѓ 154	› 155	ѝ 156	и 157	ќ 158	ў 159
nbsp 160	Ў 161	ы 162	Э 163	¤ 164	ы 165	¦ 166	§ 167	Ё 168	© 169	Ю 170	« 171	¬ 172	shy 173	® 174	Я 175
° 176	± 177	ы 178	э 179	ґ 180	µ 181	¶ 182	• 183	ё 184	№ 185	ю 186	» 187	э 188	ю 189	я 190	я 191
А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255

Текстгин стандарттык коддоолору (кодировкалары) КОИ-8:

─ 128	│ 129	┌ 130	┐ 131	└ 132	┘ 133	├ 134	┤ 135	┬ 136	┴ 137	┼ 138	▀ 139	▄ 140	▉ 141	▌ 142	▐ 143
▒ 144	▒ 145	▒ 146	⌠ 147	▬ 148	● 149	√ 150	≈ 151	≤ 152	≥ 153	nbsp 154	⌡ 155	° 156	2 157	• 158	÷ 159
═ 160	‖ 161	╒ 162	ё 163	╓ 164	╔ 165	╕ 166	╖ 167	╗ 168	╘ 169	╙ 170	╚ 171	╛ 172	╜ 173	╝ 174	╞ 175
╟ 176	╠ 177	╡ 178	Ё 179	╢ 180	╣ 181	╤ 182	╥ 183	╦ 184	╧ 185	╨ 186	╩ 187	╪ 188	╫ 189	╬ 190	© 191
ю 192	а 193	б 194	ц 195	д 196	е 197	ф 198	г 199	х 200	и 201	й 202	к 203	л 204	м 205	н 206	о 207
п 208	я 209	р 210	с 211	т 212	у 213	ж 214	в 215	ь 216	ы 217	з 218	ш 219	э 220	щ 221	ч 222	ъ 223
Ю 224	А 225	Б 226	Ц 227	Д 228	Е 229	Ф 230	Г 231	Х 232	И 233	Й 234	К 235	Л 236	М 237	Н 238	О 239
П 240	Я 241	Р 242	С 243	Т 244	У 245	Ж 246	В 247	Ь 248	Ы 249	З 250	Ш 251	Э 252	Щ 253	Ч 254	Ъ 255

№2 тиркеме

Электрондук таблицанын элементтери	Бөлүп алуу ыкмалары
Уяча	Курсорду уячага келтирүү жана ЧСБны басуу, бул учурда мамычанын атынын жана бир сызыкта жаткан саптын номеринин түсү өзгөрөт (уяча активдүү болуп калат).
Жанаша жайгашкан уячалардын диапазону	1-ыкма: курсорду диапазондун биринчи бурчтагы уячасына келирүү, shift баскычын басып, коё бербей туруп, курсорду диапазондун карама-каршы бурчуна жылдыруу, ЧСБны басуу. 2-ыкма: курсорду биринчи бурчтагы уячага келтирүү, ЧСБны басып, коё бербестен диапазондун карама-каршы бурчуна жылдыруу.
Жанаша жатпаган уячалардын диапазондору	Ctrl баскычын басып, коё бербестен, жанаша жайгашкан уячаларды бөлүп алуу ыкмасын колдонуп, өзүнчө диапазондорду бөлүп алуу.
Сап	Сол жактагы сызгычтагы саптын номерин басуу керек.
Мамыча	Жогорку сызгычтагы мамычанын аталышына басып коюу керек.
Барак	Мамычанын баш сөзү А менен саптын баш сөзү 1 дин ортосундагы тик бурчтукту басып коюу керек.

№3 тиркеме

Арифметикалык операторлор

Оператор	Мааниси	Мисалдар
+	**Кошуу** – операторлон сол жана оң жактагы маанилерди суммалайт	*10 + 5 жыйынтыгында 15 болот* *27 + -3 жыйынтыгында 24 болот* *9,4 + 7 жыйынтыгында 16,4 болот*
-	**Кемитүү** – сол операндан оң операндды кемитет	*15 - 5 жыйынтыгында 10 болот* *20 - -3 жыйынтыгында 23 болот* *13,4 - 7 жыйынтыгында 6,4 болот*
*	**Көбөйтүү** – операндарды көбөйтөт	*5 * 5 жыйынтыгында 25 болот* *7 * 3,2 жыйынтыгында 22,4 болот* *-3 * 12 жыйынтыгында -36 болот*
/	**Бөлүү** – сол жактагы операндды оң жактагысына бөлөт	*15 / 5 жыйынтыгында 3 болот* *5 / 2 жыйынтыгында 2,5 болот*
%	**Модулу боюнча бөлүү** – сол жактагы операндды оң жактагысына бөлөт жана калдыгын чыгарып берет	*6 % 2 жыйынтыгында 0 болот* *7 % 2 жыйынтыгында 1 болот* *13,2 % 5 жыйынтыгында 3,19999999 болот*
**	**Даражага көтөрүү** - сол операндды оң жактагы даражага көтөрөт	*5 ** 2 жыйынтыгында 25 болот* *2 ** 3 жыйынтыгында 8 болот* *-3 ** 2 жыйынтыгында -9 болот*
//	**Бүтүн бөлүктүү бөлүү** – бөлүүнүн жыйынтыгында бүтүн бөлүгү гана чыгарылат, үтүрдөн кийинки бөлүгү алынып салынат	*12 // 5 жыйынтыгында 2 болот* *4 // 3 жыйынтыгында 1 болот* *25 // 6 жыйынтыгында 4 болот*

Салыштыруу операторлору

==	Эки операнд тең барабар экенин салыштырат, эгерде барабар болсо, анда шарт чындык болот.	*5 == 5 жыйынтыгында True болот* *True == False жыйынтыгында False болот* *“hello”== “hello” жыйынтыгында True болот*
!=	Эки операнд тең барабар экенин салыштырат, эгерде барабар эмес болсо, анда шарт чындык болот.	*12 != 5 жыйынтыгында True болот* *False != False жыйынтыгында False болот* *“hi”!= “Hi” жыйынтыгында True болот*
<>	Эки операнд тең барабар экенин салыштырат, эгерде барабар эмес болсо анда шарт чындык болот.	*12 <> 5 жыйынтыгында True болот* *!= операторуна окшош*
>	Сол операнд оң жактагыга караганда чоң экенин салыштырат, эгер чоң болсо, анда шарт чындык болот.	*5 > 2 жыйынтыгында True болот* *True > False жыйынтыгында True болот* *“A”>“B” жыйынтыгында False болот*
<	Сол операнд оң жактагыга караганда кичине экенин салыштырат, эгер кичине болсо, анда шарт чындык болот.	*3 < 5 жыйынтыгында True болот* *True < False жыйынтыгында False болот* *“A”<“B” жыйынтыгында True болот*
>=	Сол операнд оң жактагыга караганда чоң же барабар экенин салыштырат, эгер чоң же барабар болсо, анда шарт чындык болот.	*1 >= 1 жыйынтыгында True болот* *23 >=3,2 жыйынтыгында True болот* *“C”>=“D” жыйынтыгында False болот*
<=	Сол операнд оң жактагыга караганда кичине же барабар экенин салыштырат, эгер кичине же барабар болсо, анда шарт чындык болот.	*4 <= 5 жыйынтыгында True болот* *0 <= 0,0 жыйынтыгында True болот* *-0.001 <= -36 жыйынтыгында False болот*

Ыйгаруу операторлору

=	Сол жактагы операндга оң жактагы операнддын маанисин ыйгарат.	*b = 23 b өзгөрмөсүнө 23 маанисин ыйгарат*
+=	Оң жактагы операнддын маанисине сол жактагынын маанисин кошуп, аны сол жактагы операндга ыйгарат.	*b = 5* *a = 2* *b += a барабар: b = b + a, b = 7*
-=	Сол жактагы операнддын маанисинен оң жактагынын маанисин кемитип, аны сол жактагы операндга ыйгарат.	*b = 5* *a = 2* *b -= a барабар: b = b - a, b = 3*
*=	Оң жактагы операнддын маанисине сол жактагынын маанисин көбөйтүп, аны сол жактагы операндга ыйгарат.	*b = 5* *a = 2* *b *= a барабар: b = b * a, b = 10*
/=	Сол жактагы операнддын маанисин оң жактагынын маанисине бөлүп, аны сол жактагы операндга ыйгарат.	*b = 10* *a = 2* *b /= a барабар: b = b / a, b = 5*
%=	Операнддарды модулу боюнча бөлөт жана жыйынтыгын сол жактагыга ыйгарат.	*b = 5* *a = 2* *b %= a барабар: b = b % a, b = 1*
**=	Сол жактагы операндды оң жактагынын маанисиндей даражага көтөрөт жана жыйынтыгын сол жактагыга ыйгарат.	*b = 3* *a = 2* *b **= a барабар: b = b ** a, b = 9*
//=	Сол жактагы операндды оң жактагыга бүтүн бөлүктүү бөлөт жана жыйынтыгын сол жактагыга ыйгарат.	*b = 11* *a = 2* *b //= a барабар: b = b // a, b = 5*

Арифметикалык операторлор

Оператор	Мааниси	Мисалдар
and	Логикалык оператор “ЖАНА”, эгерде эки операнд тең чындык болсо шарт чындык болот.	*True and True барабар True* *True and False барабар False* *False and True барабар False* *False and False барабар False*
or	Логикалык оператор “ЖЕ”, эгерде жок дегенде бир операнд чын болсо, анда бардык сүйлөм чындык болот.	*True or True барабар True* *True or False барабар True* *False or True барабар True* *False or False барабар False*
not	Логикалык оператор “ЭМЕС”, операнддын логикалык маанисин карама-каршысына өзгөртөт.	*not True барабар False* *not False барабар True*

Мүчөлүк операторлор

Мүчөлүк операторлор сап, тизме, кортеж же сөздүктөргө окшогон курама маалыматтардын тибинде элементтердин бар же жок экенин текшерет.

Оператор	Мааниси	Мисалдар
in	Эгерде элемент удаалаштыкта бар болсо анда чындыкты, ал эми жок болсо анда жалганды кайтарып берет.	*“cad” in “cadilac” кайтарат True 1 in[2,3,1,6] кайтарат True “hi” in {“hi”:2, “bye”:1} кайтарат True 2 in {“hi”:2, “bye”:1} кайтарат False (сөздүктөрдө маанисиндеги эмес, ачкычтардагы элементтин бардыгы текшерилет)*
not in	Эгерде элемент удаалаштыкта жок болсо анда чындыкты кайтарып берет.	*in операторунун жыйынтыгына карама-каршы жыйынтыктар.*

Теңдештик операторлору

Теңдештик операторлору компьютердин эсиндеги эки объекттин жайгашышын салыштырат.

Оператор	Мааниси	Мисалдар
is	Эгерде эки операнд тең бир объектти көрсөтсө, анда чындыкты кайтарат.	*x is y чындыкты кайтарып берет, эгерде id(x) барабар id(y) болсо*
is not	Эгерде эки операнд тең бир объектти көрсөтсө, анда жалганды кайтарат.	*x is y чындыкты кайтарып берет, эгерде id(x) барабар эмес id(y) болсо .*

№4 тиркеме

Растрдык сүрөттөрдүн форматтарынын мисалдары

ФОРМАТ	АРТЫКЧЫЛЫГЫ	КЕМЧИЛИГИ
BMP (.bmp) – атайын Windows үчүн иштелип чыккан стандарттык кысылбаган графикалык формат.	- палитра менен коддоону жана чыныгы түстүү режимди да колдойт.	- көп орунду ээлейт; - файл көп орунду ээлегендиктен, интернеттеги колдонуусу чектелүү.
JPEG (.jpg .jpeg) – фотографияларды коддоо үчүн атайын иштелип чыккан формат.	- жарыктыгы жана түсү тегиз өткөн реалдуу көрүнүштөрдү камтыган сүрөттөрдү жана фотографияларды башка форматтарга караганда эң сонун кысат.	- кысуу сүрөттүн сапатын жоготуу менен жүрөт.
GIF (.gif) – мурдагы форматтардан айырмаланып палитраны (2 ден 256 түскө чейинки) коддоону гана колдогон формат.	- сүрөттүн кээ бир бөлүгү тунук болушу мүмкүн, б.а. веб-бараkта алардын артынан фон көрүнүп турат; - анимацияны колдойт.	- түстүн аз санда болушу; - бир нече статикалык кадрлардын удаалаштыгынан турган анимациялык сүрөттөрдү колдойт, андан тышкары ар бир кадр экранда канча убакыт көрсөтүлүшү жөнүндөгү маалыматты да колдойт.
PNG (.png) – чыныгы түстүү режимди да, палитра менен коддоону да колдогон формат; сүрөттүн кээ бир бөлүгү тунук же жарым тунук болушу мүмкүн.	- кысууда сапат жоголбойт; - калыбына келтирүү жана сүрөттү кайра сактоо сапатты жоготуусуз ишке ашат.	- бул формат башында GIF эскирген форматты алмаштыруу үчүн долбоорлонгонуна карабастан, PNG бир файлда бир эле сүрөттү сактай алат.

Глоссарий

CMS – мазмунун (контентин) башкаруу системалары (*англ.* content management system, CMS) – бул сайттын мазмунун башкаруу үчүн пайдаланылуучу программалык камсыздоо.

Аутентификация – бул кайсы бир нерсенин аныктыгын текшерүү процесси. Аутентификациянын мисалы катары колдонуучу тарабынан киргизилген пароль менен сервердин маалыматтар базасында сакталган паролду салыштыруу болушу мүмкүн.

«Бот» («робот» сөзүнүн кыскартылышы) – адамдын иштерин туураган программа. Чат-бот чаттагы маектешти туурайт.

Видеоэс – бул графикалык сүрөттөлүш түзүлгөн оперативдүү эстин бөлүгү.

Гиподинамия – организмдин төмөнкү кыймыл активдүүлүгү, жалпы физикалык активдүүлүктүн төмөндөшү.

Дискреттөө – бул графикалык маалыматты аналогдук формадан дискреттик формага өзгөртүп түзүү, башкача айтканда үзгүлтүксүз графикалык сүрөттү өзүнчө элементтерге бөлүп чыгуу.

Жөнөкөй логикалык айтым – бул маанисин жоготпостон туруп кичирейтүүгө же бөлүүгө мүмкүн болбогон айтым.

Интерпретатор – бул сиздин программаңызды окуп, андагы камтылган нускаманы аткаруучу программа.

Контент (*англ.* мазмуну) – бул колдонуучуга арналган маалымат жана тажрыйба. Сайттын контенти – бул анда камтылган текст, сүрөт, видео, аудио.

Код – бул маалыматты чагылдыруу үчүн шарттуу белгилердин жана эрежелердин системасы.

Коддоо – бул берилген коддун жардамы менен маалыматты чагылдыруу.

Логикалык туюнтма – бул логикалык амалдар менен бириктирилген логикалык айтымдар.

Лэндинг (*англ.* landing page) – эң негизги максаты сайтка кирүүчүнү кандайдыр-бир конкреттүү (максаттуу) аракетти жасатуу болгон веб-баракча: мисалы, бир нерсе үчүн добуш берүү, кайсы бир товар менен таанышуу жана аны сатып алуу, китепке заказ берүү ж.б.

Лонгрид (*англ.* Longread) – ар кандай мультимедиялык элементтердин (фотография, сүрөттөр, видео, диаграмма ж.б.) жардамы менен блокторго бөлүнгөн узун текст (макала, баян).

Маалыматтар базасы – бул эсептөө системасында эффективдүү издөө жана иштетүү максатында логикалык системалаштырылган маалыматтар топтому.

Маалыматтык сабаттуулук - бул адамдын маалыматты издөө, тандоо, баалоо жана колдонуу көндүмдөрүнүн керектигин аңдап билүү жөндөмдүүлүгү.

Өзгөрмө – бул аты, тиби жана мааниси бар чоңдук. Өзгөрмөнүн мааниси программанын аткарылышында өзгөрүп турушу мүмкүн.

Протокол – бул тармак аркылуу маалыматты берүүнү жүргүзгөн тиешелүү эрежелер.

Табигый тил – адамдардын баарлашуусу үчүн колдонулат (орус, кыргыз, англис ж.б. тилдер).

Татаал айтым логикалык амалдарга дал келген логикалык байламталар менен бириккен жөнөкөй айтымдардан турат.

Түстүн тереңдиги – бул пикселдин түсүн коддоо үчүн колдонулган биттердин саны.

Формалдаштыруу – бул символдордун жардамы менен конкреттүү мазмундан (айтымдан) формалдуу жазууга өтүү.

Формалдуу тил – бул сүйлөм түзүүнүн так эрежелери менен мүнөздөлгөн жасалма тил: ноталар, Морзе алиппеси, химиялык элементтер символдору, программалоо тилдери.

Фэйк (*англ.* fake – «жасалма, жалган») маалыматты берүүдө «жалган маалымат» деген маанини түшүндүрөт.

Чечүү – сүрөттөлүштүн бир дюйм өлчөмүнө туура келүүчү пикселдердин саны.

И. Н. Цыбуля, Л. А. Самыкбаева,
А. А. Беляев, Н. Н. Осипова, У. Э. Мамбетакунов

Информатика: 7–9-класс

Китептин э-версиясы www.lib.kg сайтында жайгаштырылган.

Котормочу: Касымбек Жунусалиев

Текст редактору: Исмаил Кадыров

Корректор: Өктөм Калыева

Техникалык эксперттер: А. Палитаев, И. Кошмурзаев,
И. Ташиев, С. Маматов, А. Сабырова

Адабий редактор: Диана Светличная

Арт-директор: Мария Казакова

Компьютердик калыпка салган: Абдымалик Токталиев

Басууга 16.09.2020-ж кол коюлду. Офсеттик кагаз. Офсеттик басма.
Форматы 70x100 $^{1}/_{16}$. PT Sans ариби. Көлөмү 13 накта
басма табак. Нускасы 69 850 даана. Буйрутма №321.

Басма-полиграфиялык комплекс «Принт Экспресс» ЖЧК басмаканасында басылды
Кыргыз Республикасы, Бишкек ш., Профсоюз көч. 49

КЫМБАТТУУ ОКУУЧУ, КИТЕБИҢДИ ТАЗА УРУНГУН

Сен бул китеп келерки окуу жылында да колдонуларын унутпа.

Бул баракчада класс жетекчиң китепти кандай урунганыңды жана кандай тазалыкта өткөрүп бергениңди белгилейт.

Окуу китептин пайдаланылгандыгы жөнүндө маалыматтар.

№	Окуучунун аты-жөнү	Окуу жылы	Окуу китептин абалы	
			жыл башында	жыл аягында
1				
2				
3				
4				
5				