

PYTHON ПРОГРАММАЛОО ТИЛИНИН НЕГИЗДЕРИ



Н.Кубатов

Киришүү

Бул колдонмодо - программалоо тилдери, алардын өнүгүү этаптары, программанын кодун которуу усулдары, негизги типтер жана маалыматтар структуралары(бүтүн жана анык сандар, саптык, тизме, сөздүк, кортеж), өзгөрмөлөр, шарттуу оператор, өзгөчө абал жана аларды кайра иштетүү, циклдык операторлор, функциялар жана модулдар, обп жана turtle, tkinter графикалык китепканасы, маалыматтарды киргизүү жана чыгаруу жана файлдар менен иштөө каралган.

Бул колдонмо:

мектеп окуучулары, студенттер

программаны жаңы баштагандар.

информатика мугалимдери жана программалоого кызыккандар үчүн сунушталат.

Колдонмо интернеттен алынган булактардан, электрондук китептерден жана видео сабактардын негизинде жазылды.

E-mail: kubanura57@gmail.com

©ANARSOFT, Н. Кубатов

Ноокен – 2022-ж.

Программалоого киришүү. Python программасына киришүү жана интерпретаторун орнотуу.

Программалоо тилдеринин кыскача өнүгүү тарыхы.

Машиналык тил деген эмне?, эмне үчүн ассемблер, жогорку деңгээлдеги жана объектиге багытталган программалар пайда болду? Эмне үчүн транслятор керек жана эмне үчүн ал компилятор же интерпретатор боло алат?

Программа – бул максатка жетүү үчүн колдонучуу тарабынан жазылган буйруктардын ирети.

Адатта программа адамдар үчүн эмес, машиналар үчүн кабыл алынган, аны колдонуу үчүн атайын программалоо тилдери бар. Атайын тилдерди пайдалануунун себеби, табигый тилде жазылган алгоритмдерди түшүнүү үчүн керек. Машинаны жөндөө үчүн түрдүү аныкталган синтаксистик(мисалы программанын өздүк сөзүнүн жайын алмаштырууга болбойт) жана чектелген(мисалы сөздөрдүн жана символдордун аныкталган топтому) касиетке ээ программалоо тилдери бар.

Программалоо тилдеринин негизги өнүгүү тарыхынын этаптары.

Биринчи программаны машиналык тилде жазышкан б.а. ал учурда ЭЭМдин программалык жабдылышы болгон эмес, машиналык тил – бул компьютердин аппараттык жабдылышы менен байланыштыруучу бирден – бир жолу эле. Маалыматтарды жана буйруктарды программистер цифралык түрдө жазышкан(мисалы 16 же 2 эсептөө системасы). Адамдардын машинадан артыкчылыгы цифраларга караганда сөздөрдү түшүнгөндүгүндө. Адамдардын сөз менен иштөөгө умтулусу ассемблердин пайда болушуна алып келди. Мындай учурда көгөй пайда болот: машина сөздөрдү түшүнүүгө кадыр эмес. Машиналык тилге которуу үчүн котормочу керек. Ошондуктан, ассемблер түзүлгөндөн бери программалык кодду машиналык кодго айландыруучу трансляторлор ар бир программалоо тили үчүн түзүлгөн. Ассемблерлер бүгүнкү күнгө чейин колдонулат(системалык программаларда алардын жардамы менен төмөнкү деңгээлдеги операциялык системанын интерфейсин, драйверлердин компонентин түзүүгө болот). Ассемблерден кийин жогорку деңгээлдеги программалар чыга баштады. Бул программалоо тилдери үчүн татаал котормочулар талап кылынат. Бирок жогорку деңгээлдеги тилдер адамдар үчүн ыңгайлуу. Ассемблерден айырмасы жогорку деңгээлдеги программалоо тилдери көчмө болуп эсептелет. Бул дегени бир программаны жазгандан кийин аны башка компьютерге которуп аткаrsa болот, эгер анда дал келуучу транслятор орнотулган болсо. Кийинки негизги кадам объектиге багытталган программалоо тили, программаларды татаалдаштырууга өркүндөтүлгөн түрү болуп эсептелет. Мындай программалоо тилдеринин жардамы менен программистер жасалма объектилерди башкарууда аныкталган мааниде программаны чындыкка жакындатат. Бүгүнкү күндө көпчүлүк учурларда татаал проектерди ишке ашырууда объектиге багытталган программалоо жардам берет.

Программалоо тилдери.

Азыркы учурда көптөгөн түрдүү жана бири-бирине окшош программалоо тилдери бар. Бүгүнкү күндөгү компьютер технологияларынын жардамында аткарыла турган тапшырмалардын саны жана ар кылдыгын эске алсак, бул көрүнүштүн себеби ачык-айкын болот. Ар кандай тапшырмаларды чечүү үчүн ар кандай аспаптар талап кылынат б.а. ар кандай тилдер жана программалоо тилинин парадигмалары.

Азыркы программалоо тилдерин түрдүү критерийлер менен класстарга бөлсө болот. Мисалы, тапшырмаларды аткаруу түрү боюнча(системалык жана колдонмо багытындагы программалар, web-барактарын иштеп чыгуу үчүн, маалыматтар базасы, мобилдик тиркемелерди иштеп чыгуу ж.б.). Алардын арасынан бүгүнкү күндө атактуулука жеткен тилдер Java, C++, PHP булардын катарына Python да кирет.

Трансляция.

Жогоруда айтылгандай бир тилде жазылган(мисалы, жогорку деңгээлдеги программалоо тили) кодду башкага(мисалы, машиналык тил) которууда атайын транслятордук программа талап кылынат.

Котормочуга мындай алгоритмди которуу татаал болот. Программаны трансляциялоонун эки(компиляция(түзүү) жана интерпретация(түшүндүрүү)) жолу бар. Компилятор баштапкы программалык кодду(программист жазган программа) дароо машиналыкка которот. Баштапкы кодго байланышы жок өзүнчө аткаруучу файлды түзөт. Аткаруучу файлды аткарууну аракеттер системасы камсыз кылат. Аткаруучу файл түзүлгөндөн кийин аны окуу үчүн компилятор керек эмес. Интерпретацияда, кодду аткаруу иреттик тартипте жүрөт(саптан сапка). Мындайча айтканда аракеттер системасы менен интерпретатор өз ара аракеттенишет. Эч кандай программалык кодду түзгөн файл менен эмес. Интерпретатор программалык коддун бөлүгүн окуп, аны машиналык(ОС түшүнө турган) кодот жана ОСга берет. ОС берилген кодду аткарат жана интерпретатордон кийинки маалыматты күтөт. Python так ушундай тил. Python интерпретатордук программалоо тили. Компилятор программаны аткарууда ылдамырак, башкача айтканда ал өзүнө даяр машиналык кодду алат. Бирок, заманбап компьютерлерде интерпретатордук программалардын ылдамдыгынын секиндиги байкалбайт.

Транслятор- программалоо тилдеринде түзүлгөн программаны машиналыкка которуучу атайын программа.

Компилятор – бардык программалык кодду дароо машиналык тилге которот. Аткаруучу файлды түзөт.

Интерпретатор – Программалык кодду саптан сапка которот.Операциялык система менен түзмө - түз аракеттенет.

Python программалоо тили.

Python - ар кандай тиркемелерди түзүү үчүн иштелип чыккан популярдуу жогорку деңгээлдеги программалоо тили. Бул веб-тиркемелер, оюндар жана терезелүү программалар жана маалымат базалары менен иштөө. Python машина үйрөнүү жана жасалма интеллект изилдөө чөйрөсүндө кыйла кеңири таралган.

Python биринчи жолу 1991-жылы голландиялык иштеп чыгуучу Гвидо Ван Россум тарабынан жарыяланган. Ошондон бери тил өнүгүүнүн узак жолун басып өттү. 2000-жылы 2.0 версиясы, 2008-жылы 3.0 версиясы жарык көргөн. Версиялардын ортосундагы чоң боштуктарга карабастан, камтылган версиялар дайыма чыгарылып турат.

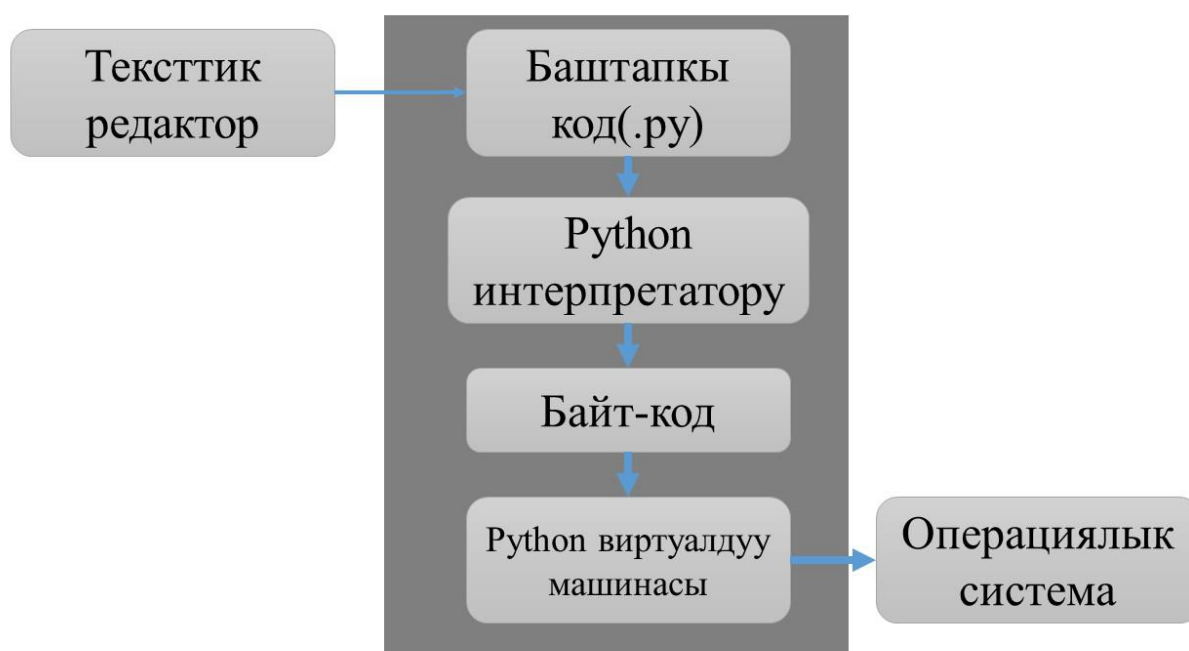
Python программалоо тилинин негизги өзгөчөлүктөрү:

Скрипт тили. Программанын коду скрипт түрүндө аныкталат.

Программалоонун ар кандай парадигмаларын, анын ичинде объектиге багытталган жана функционалдык парадигмаларды колдойт.

Программанын интерпретациясы. Скрипттер менен иштөө үчүн скрипти ишке киргизген жана аткара турган котормочу керек.

Python программасынын аткарылышы ушундай көрүнөт: Биринчиден, биз тексттик редактордо берилген программалоо тилиндеги туюнтмалардын жыйындысы менен сценарий жазабыз. Биз бул сценарийди аткаруу үчүн интерпретаторго өткөрүп беребиз. Интерпретатор кодду аралык байт-кодго которот, андан кийин виртуалдык машина пайда болгон байт-кодду операциялык система тарабынан аткарылуучу нускамалардын жыйындысына которот. Бул жерде формалдуу түрдө интерпретатор тарабынан баштапкы кодду байт-кодго которуу жана виртуалдык машинанын байт-кодду машиналык инструкциялардын жыйындысына которуу эки башка процесс болгону менен, алар чындыгында интерпретатордун өзүндө айкалышкандыгын белгилей кетүү керек.



Портативдик жана көз карандысыздык. Бизде кандай операциялык система бар экендиги маанилүү эмес - Windows, Mac OS, Linux, биз жөн гана интерпретатор бар болсо, ушул операциялык системалардын бардыгында иштей турган скрипт жазышыбыз керек. Автоматтык эстутум башкаруу. Динамикалык типтештирүү.

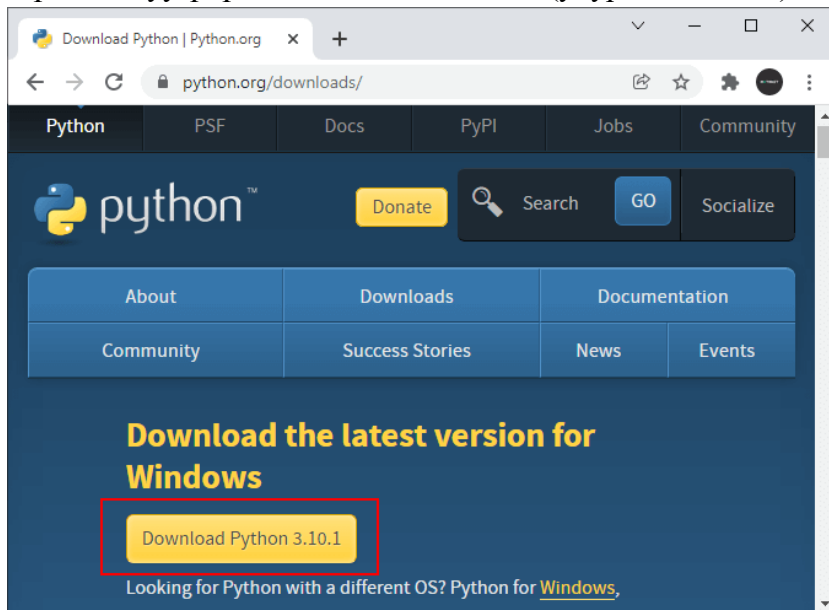
Python абдан жөнөкөй программалоо тили болуп саналат, ал кыска жана ошол эле учурда абдан жөнөкөй жана түшүнүктүү синтаксиси бар. Демек, аны үйрөнүү оңой жана бул чындыгында ал үйрөнүү үчүн эң популярдуу программалоо тилдеринин бири экендигинин себептеринин бири. Тактап айтканда, 2014-жылы ал АКШда үйрөнүү үчүн эң популярдуу программалоо тили катары таанылган.

Python билим берүү тармагында гана эмес, конкреттүү программаларды, анын ичинде коммерциялык программаларды жазууда да популярдуу. Ошондуктан, биз колдоно ала турган бул тил үчүн көптөгөн китепканалар жазылган.

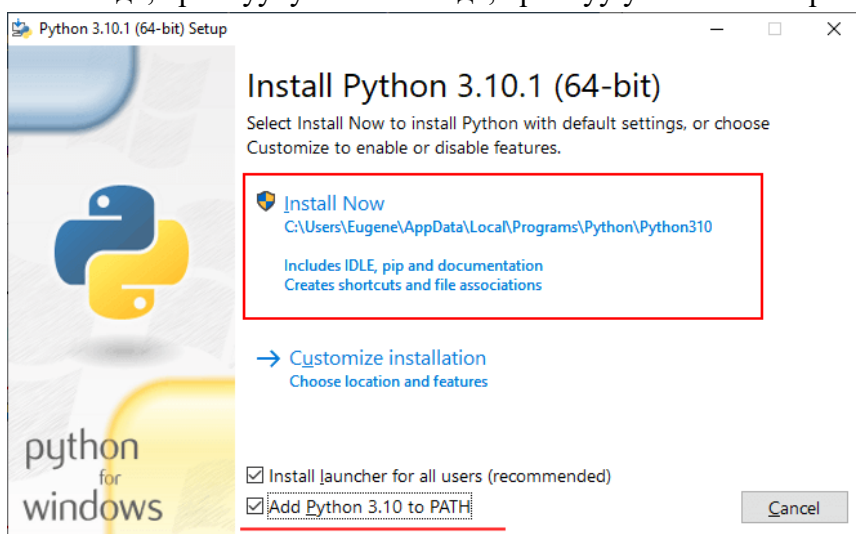
Мындан тышкары, бул программалоо тили абдан чоң коомчулукка ээ, интернетте бул тил боюнча көптөгөн пайдалуу материалдарды, мисалдарды таба аласыз, квалификациялуу адистерден жардам ала аласыз.

Python интерпретаторун орнотуу.

Pythonдо программаларды түзүү үчүн бизге интерпретатор керек. [Аны орнотуу үчүн https://www.python.org/downloads/](https://www.python.org/downloads/) баракчасына өтүңүз жана тилдин акыркы версиясын жүктөп алуу үчүн шилтемени табыңыз (учурда ал 3.10.1):



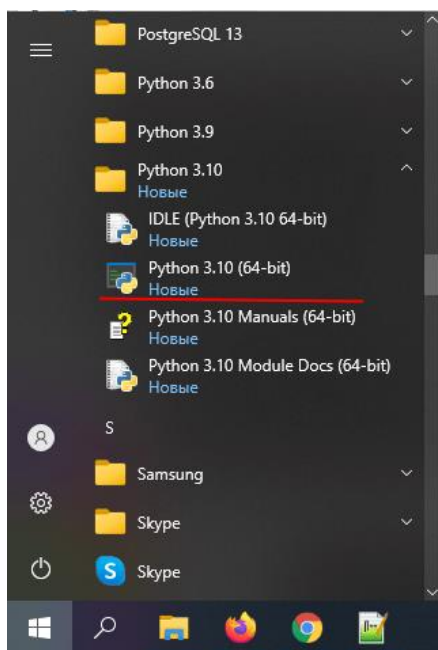
Баскычты басуу менен, учурдагы ОСга туура келген Python орнотуучусу жүктөлөт. Windows 7 жана мурунку версиялары колдоого алынбагандыгын эске алыңыз. Windows ОСде, орнотуучу башталганда, орнотуу устасынын терезесин ишке киргизет:



Бул жерден интерпретатор орнотула турган жолду орното алабыз. Келгиле, аны демейки катары калтыралы.

Ошондой эле, эң ылдый жагында интерпретаторго жолду чөйрө өзгөрмөлөрүнө кошуу үчүн "Add Python 3.10 to PATH" кутучасын белгилеңиз.

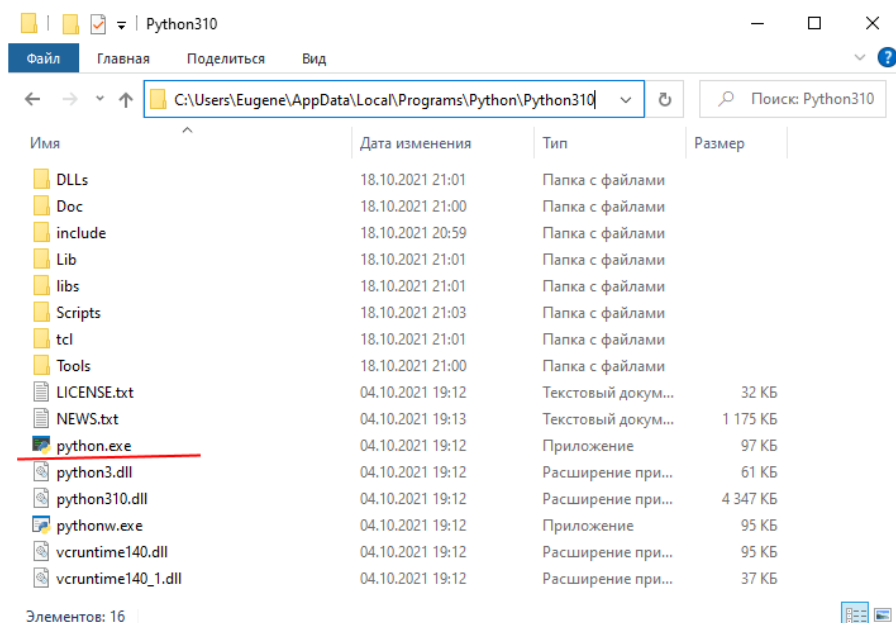
Windows ОСтин ПУСК менюсуна орнотулгандан кийин, биз ар кандай python утилиталарына жетүү үчүн иконаларды таба алабыз:



Бул жерде Python 3.10 (64-бит) утилитасы скриптти иштете турган интерпретатор менен камсыз кылат. Файлдык тутумда интерпретатор файлдын өзүн орнотуу аткарылган жолдон табууга болот. Windows'та демейки жол C:\Users\[колдонуучунун аты]\AppData\Local\Programs\Python\Python310 жана интерпретатордун өзү python.exe файлын билдирет. Linux ОСде орнотуу /usr/local/bin/python3.10 жолу боюнча жүргүзүлөт.

Биринчи программа.

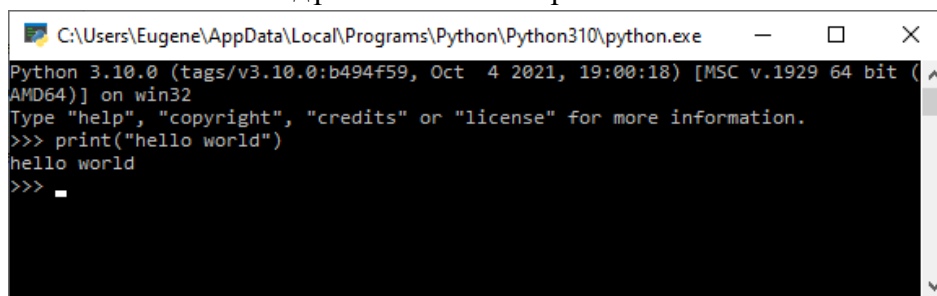
Интерпретатор орнотулгандан кийин, мурунку темада айтылгандай, биз Pythonдо тиркемелерди түзө баштайбыз. Ошентип, биринчи жөнөкөй программаны түзөлү. Акыркы темада айтылгандай, интерпретатор программасы, эгерде орнотуу учурунда дарек өзгөртүлбөсө, Linux'та usr/local/bin/python310 жолу боюнча, ал эми Windows'до C:\Users\[колдонуучунун аты боюнча орнотулган.]\AppData\Local\Programs\Python\Python310\ жана python.exe деп аталган файлды билдирет.



Интерпретаторду иштетиңиз жана ага төмөнкү сапты киргизиңиз:

```
1 print("салам дүйнө")
```

Жана консол "салам дүйнө" сапты чыгарат:

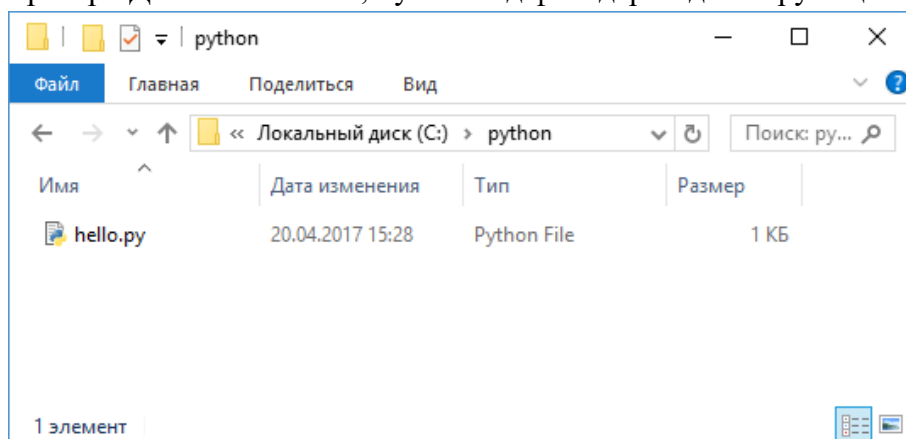


```
C:\Users\Eugene\AppData\Local\Programs\Python\Python310\python.exe
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> _
```

Бул программа үчүн, консолго кандайдыр бир сапты басып чыгарган `print()` функциясы колдонулган .

Программа файлын түзүү.

Чындыгында, эреже катары, программалар тышкы скрипт файлдарында аныкталат жана андан кийин аткаруу үчүн котормочуга берилет. Ошентип, программа файлын түзөлү. Бул үчүн, C дискинде же файл тутумунун башка жеринде скрипттердин `python` папкасын аныктаңыз. Жана бул папкада биз `hello.py` деп атай турган жаңы текст файлын түзөбүз. Демейки боюнча, Python код файлдары адатта `.py` кеңейтилишине ээ.



Бул файлды каалаган текст редакторунда ачып, ага төмөнкү кодду кошуңуз:

```
1 name = input("Атыңызды киргизиңиз:")
2 print("Салам", name)
```

Сценарий эки саптан турат. Биринчи сап `input()` функциясын колдонуп, колдонуучунун атын киргизүүнү күтөт. Киргизилген ат андан кийин `name` өзгөрмөсүнө өтөт.

Экинчи сап киргизилген ат менен саламдашууну басып чыгаруу үчүн `print()` функциясын колдонот.

Эми буйрук сабын/терминалды ишке киргизели жана `hello.py` булак файлы жайгашкан папкага өтүү үчүн `cd` буйругун колдонолу (мисалы, менин учурда бул `C:\python` папкасы).
`cd c:\python`

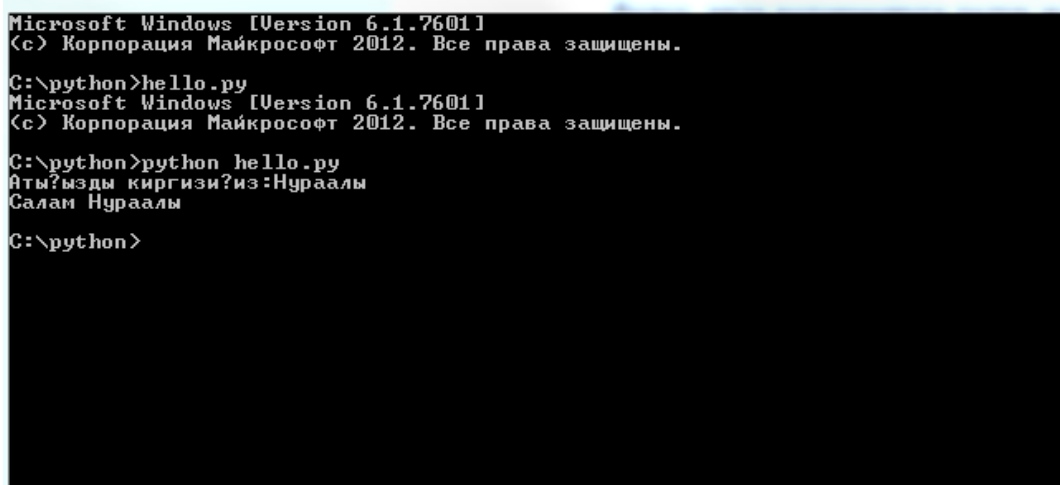
Андан кийин, адегенде котормочуга толук жолду, андан кийин скрипт файлына толук жолду киргизиңиз. Мисалы, менин учурда, консолдо сиз төмөнкүлөрдү киргизишиңиз керек:

```
C:\Users\Eugene\AppData\Local\Programs\Python\Python310\python.exe hello.py
```


Бирок орнотуу учурунда "Add Python 3.10 to PATH" параметри көрсөтүлгөн болсо, башкача айтканда, Python интерпретаторуна баруучу жол чөйрө өзгөрмөлөрүнө кошулган болсо, анда котормочуга толук жолдун ордуна, сиз жөн гана python жазсаңыз болот:

python hello.py

ишке киргизүүнүн ыкмасын колдонуу:



```
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт 2012. Все права защищены.

C:\python>hello.py
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт 2012. Все права защищены.

C:\python>python hello.py
Атыңызды киргизи?из:Нураалы
Салам Нураалы

C:\python>
```

Натыйжада, программа атын, андан кийин саламдашууну киргизүү үчүн сунушту көрсөтөт.

PyCharmды орнотуу

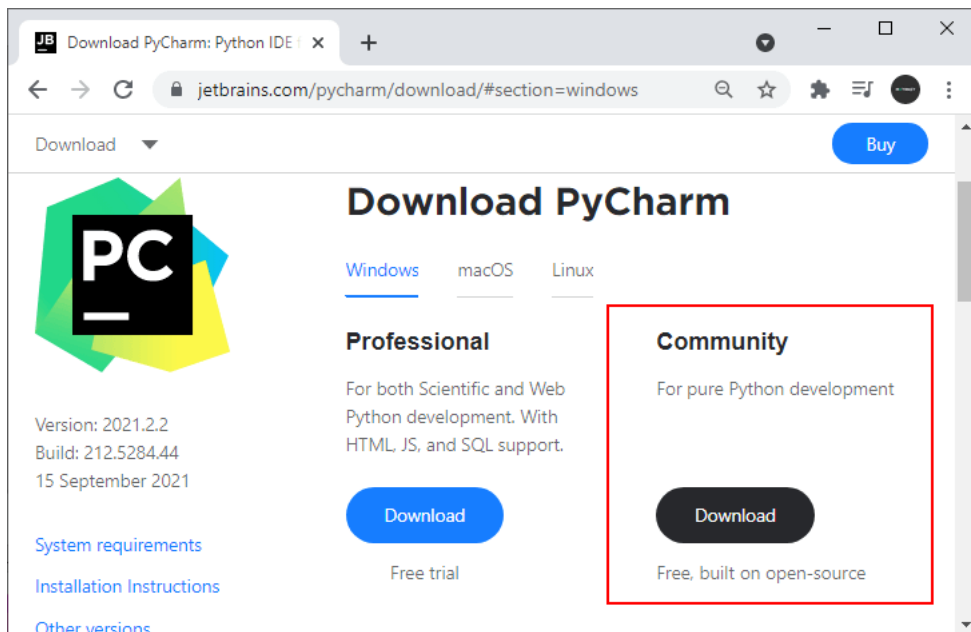
Акыркы темада Python тилинде эң жөнөкөй скрипт түзүү баяндалган. Сценарийди түзүү үчүн тексттик редактор колдонулган. Менин учурда бул Notepad ++ болчу. Бирок программаларды түзүүнүн дагы бир жолу бар, бул ар кандай интеграцияланган иштеп чыгуу чөйрөлөрүн же IDEлерди колдонуу.

IDEлер бизге кодду терүү үчүн текст редактору менен камсыз кылат, бирок стандарттуу текст редакторлордон айырмаланып, IDEлер ошондой эле толук синтаксисти бөлүп көрсөтүүнү, автотолтурууну же акылдуу кодду кыйытууну, түзүлгөн скрипти дароо аткаруу мүмкүнчүлүгүн жана башка көптөгөн нерселерди камсыздайт.

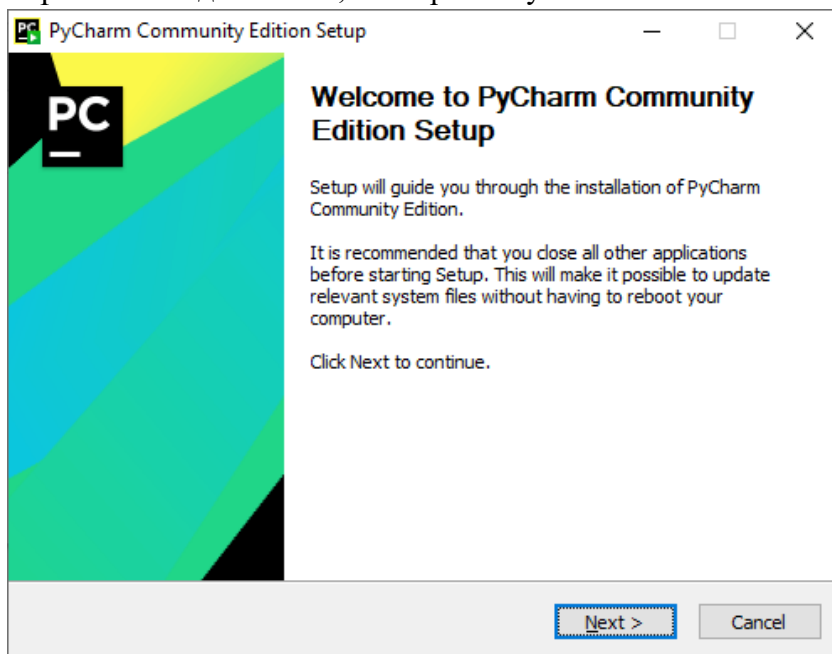
Python үчүн колдонула турган ар кандай иштеп чыгуу чөйрөлөрү бар, бирок эң популярдуу PyCharm JetBrains тарабынан түзүлгөн. Бул чөйрө динамикалуу өнүгүп, дайыма жаңыланып турат жана эң кеңири таралган операциялык системалар үчүн жеткиликтүү - Windows, MacOS, Linux.

Ырас, анын бир маанилүү чектөөсү бар. Тактап айтканда, ал эки негизги версияда жеткиликтүү: акы төлөнүүчү Professional(Кесиптик) чыгарылыш жана бекер Community(Коомчулук) чыгарылышы. Көптөгөн негизги функциялар Community акысыз чыгарылышында да бар. Ошол эле учурда, веб-иштеп чыгуу сыяктуу бир катар функциялар акы төлөнүүчү Professional’да гана жеткиликтүү.

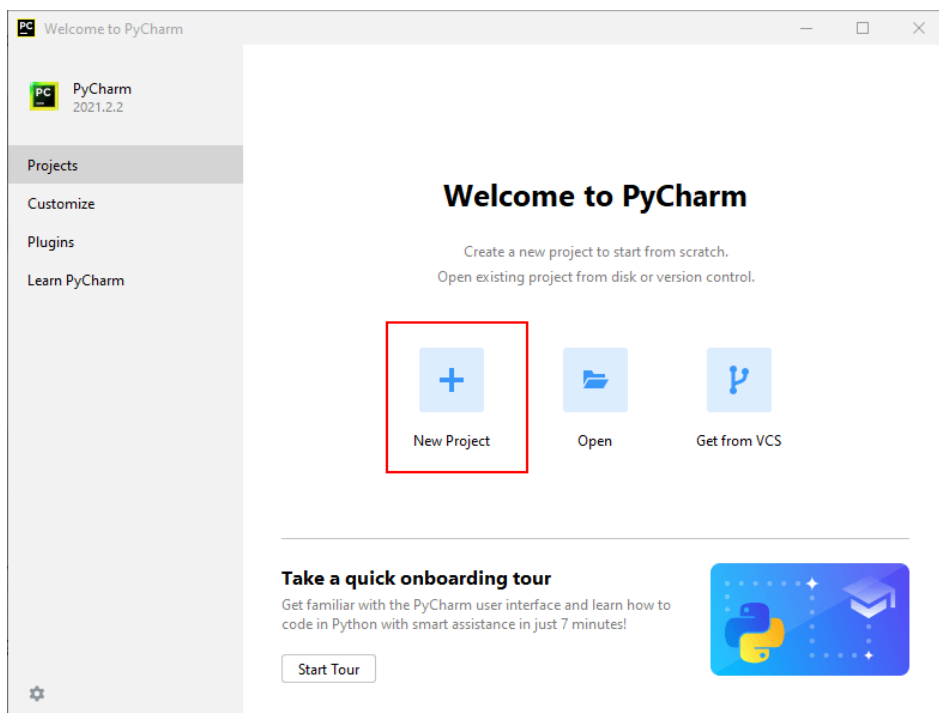
Биздин учурда, биз Community акысыз чыгарылышын колдонобуз. Бул үчүн [. жүктөө барагына](#) өтүп, PyCharm Community орнотуу файлын жүктөп алыңыз.



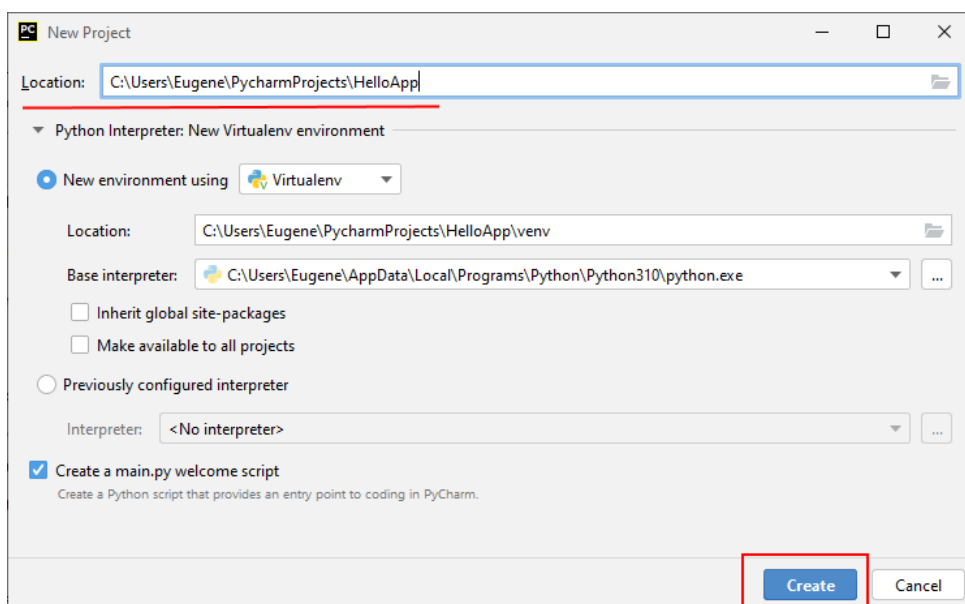
Жүктөп алгандан кийин, аны орнотобуз.



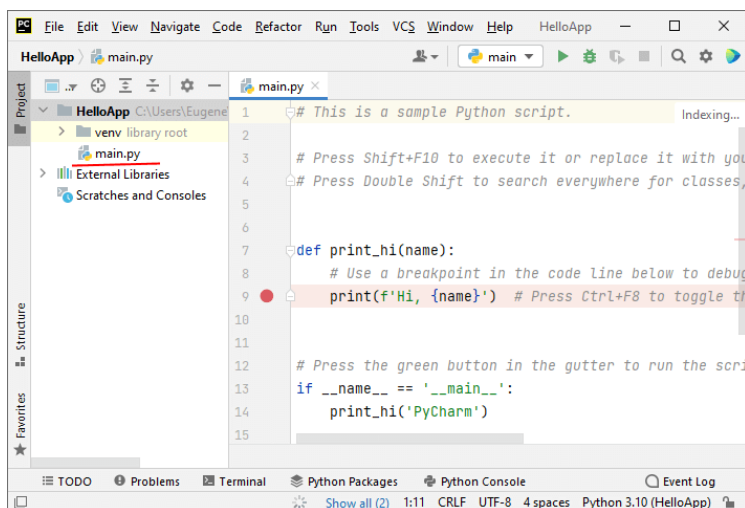
Орнотуу аяктагандан кийин, программаны иштетиңиз. Биринчи баштаганда, баштапкы терезе ачылат:



Келгиле, долбоор түзөлү жана бул үчүн биз New Project тандайбыз. Андан кийин, биз долбоорду орнотуу үчүн терезени ачабыз. Location талаасында сиз долбоорго жолду көрсөтүшүңүз керек. Менин учурда, долбоор HelloApp папкасына жайгаштырылат. Папканын аталышы долбоордун аталышы болот.



Долбоордун жолун кошпогондо, биз бардык башка орнотууларды демейки катары калтырып, долбоорду түзүү үчүн "Create" баскычын басыңыз. Бул бош долбоорду түзөт:

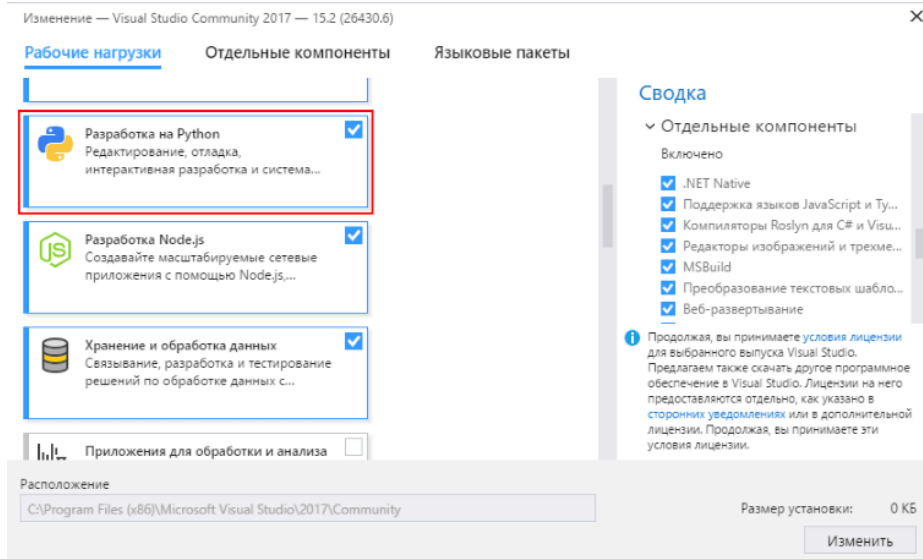


main.py файлы демейки мазмуну менен чөйрөнүн ортосунда ачылат .

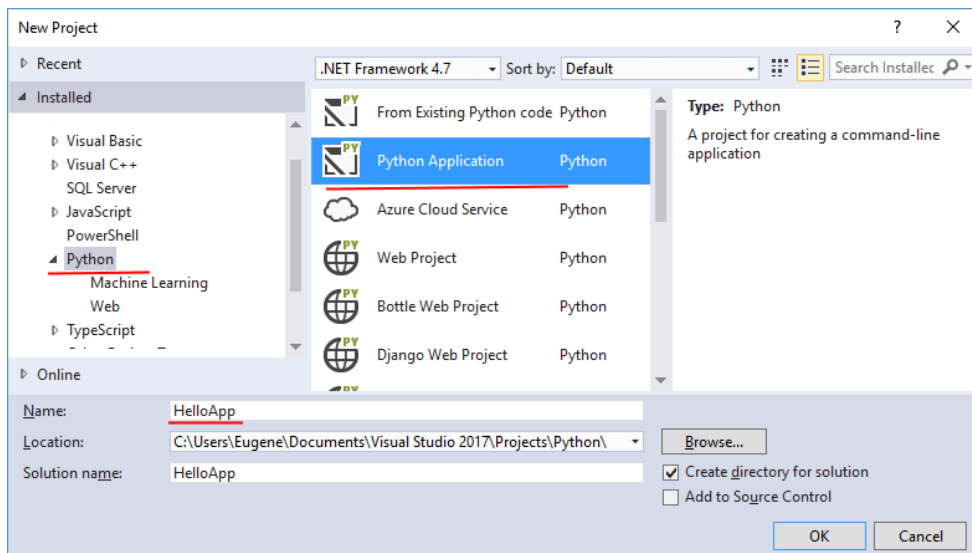
Visual Studioну орнотуу.

Python менен иштөөгө мүмкүндүк берген бир иштеп чыгуу чөйрөсү - Visual Studio. Мисалы, бул веб-иштеп чыгуу, анын ичинде ар кандай алкактарды колдонуу. Ошол эле учурда, Visual Studioдогу Python өнүктүрүү куралдары учурда Windows версиясында гана жеткиликтүү.

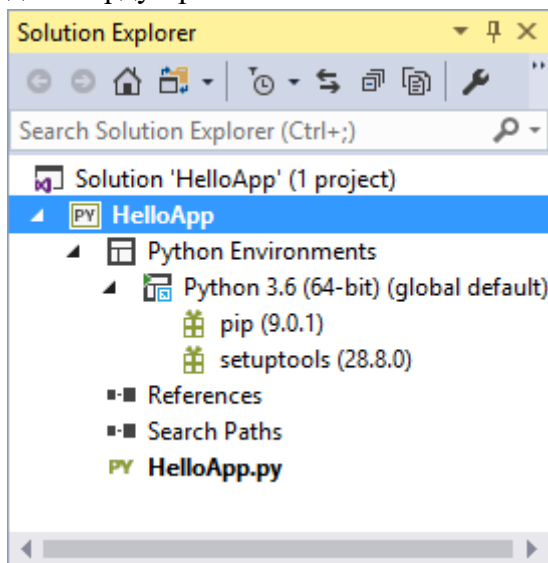
Ошентип, келгиле, <https://visualstudio.microsoft.com/downloads/> шилтемесинен Visual Studio 2019 Community орнотуу файлын жүктөп алалы . Орнотуу файлын иштеткенден кийин, орнотула турган Python параметрлеринин арасынан тандаңыз:



Visual Studio орнотулгандан кийин, аны иштетиңиз. Менюдан File (Файл) -New (Түзүү) -project (долбоор) пунктун тандаңыз, ошондо биз жаңы долбоорду түзүү үчүн терезени көрөбүз. Бул терезеде, сол дарак менюсунда, биз Python тилине өтө алабыз:



Сол жактагы Pythonду тандоо менен, терезенин борбордук бөлүгүндө биз бул программалоо тилинде иштеп чыгуу үчүн түзө турган долбоор түрлөрүнүн бай палитрасын көрө алабыз. Буга веб-иштеп чыгуу, машина үйрөнүү, булут долбоорлору, терезелүү тиркеме долбоорлору ж.б. Бул учурда долбоордун түрү катары Python тиркемесин, башкача айтканда, жөнөкөй консолдук тиркемелердин түрүн тандап, жаңы долбоорду HelloApp деп атайлы. OK баскычын чыкылдатыңыз жана Visual Studio жаңы долбоорду түзөт:



Solution Explorer терезесинин оң жагында долбоордун түзүмүн көрө аласыз. Демейки боюнча, бул жерде биз төмөнкү элементтерди көрө алабыз:

Python Environments: бул жерде сиз колдонулган бардык чөйрөлөрдү көрө аласыз, атап айтканда, бул жерде сиз колдонулган компилятор жөнүндө маалыматты таба аласыз.

References: Учурдагы долбоор тарабынан колдонулган бардык тышкы көз караңгылыктар бул түйүндө жайгаштырылат

Search path: Бул түйүн Python модулдары үчүн издөө жолдорун көрсөтүүгө мүмкүндүк берет

HelloApp.py: чыныгы Python булак файлы

Pythonдо программалоого киришүү

Python программасы инструкциялардын жыйындысынан турат. Ар бир нускама жаңы сапка жайгаштырылат. Мисалы үчүн:

```
1 print(2 + 3)
2 print("Салам")
```

Pythonдо чегинүү чоң роль ойнойт. Туура эмес чегинүү чындыгында ката. Мисалы, кийинки учурда биз ката алабыз, бирок код жогорудагыдай дээрлик бирдей болот:

```
1 print(2 + 3)
2 print("Салам")
```

Ошондуктан, жаңы нускамаларды саптын башында коюу керек. Бул Python жана C# же Java сыяктуу башка программалоо тилдеринин ортосундагы маанилүү айырмачылыктардын бири.

Бирок, кээ бир түзүлүштөр бир нече саптан турушу мүмкүн экенин эстен чыгарбоо керек. Мисалы, if шарты :

```
1 if 1 < 2:
2     print("Салам")
```

Бул учурда, эгерде 1, 2ден аз болсо, анда "Салам" сабы көрсөтүлөт. Ал эми бул жерде мурунтан чегинүү болушу керек, анткени print("Hello") оператору өз алдынча эмес, if шарттуу түзүлүшүнүн бир бөлүгү катары колдонулат. Мындан тышкары, чегинүү, коддуу [стилдик көрсөтмөсүнө](#) ылайык, 4кө эселенген мейкиндиктерден (б.а. 4, 8, 16 ж.б.) 4 эмес, 5 чегинүү болсо да жасоо керек. , анда программа да иштейт.

Мындай инструкция анчалык деле көп эмес, ошондуктан кайсы жерде зарыл жана мейкиндикти кайда коюу керектиги жөнүндө көп баш аламандык болбошу керек.

Регистер сезгич

Python регистрге сезимтал тил, ошондуктанPrint жана PRINT башка башка билдирүү. Ал эми консолго чыгаруу үчүн биз Print ыкмасын колдонууга аракет кылабыз:

```
1 Print("Салам дүйнө")
```

Жыйынтыгында биз эч нерсе албайбыз

Комментарийлер

Комментарийлер коддун бир бөлүгү эмне кылаарын белгилөө үчүн колдонулат. Программа которулганда жана аткарылганда интерпретатор комментарийлерге көңүл бурбайт, ошондуктан алар программанын иштешине эч кандай таасир этпейт. Pythonдо комментарийлер блокто жана саптык комментарийлерде келет. Саптык комментарийлердин алдында торчо белгиси коюлат - # . Алар өзүнчө сапта болушу мүмкүн:

```
1 # Консолго чыгаруу
2 # Салам дүйнө билдирүүлөрү
3 print("Салам дүйнө")
```

белгисинен кийинки символдордун каалаган топтому комментарийди билдирет. Башкача айтканда, жогорудагы мисалда, коддун биринчи эки саптары комментарийлер.

Алар ошондой эле инструкциялар аткарылгандан кийин тил көрсөтмөлөрү менен бир сапта жайгаша алат:

```
1 print("Hello World") # Кабарды консолго басып чыгаруу
```

Блок комментарийлеринде үч жалгыз тырмакча комментарий текстинин алдында жана кийин коюлат: "комментарий тексти" . Мисалы үчүн:

```
1  """
2      Консолдук чыгаруу
3      салам дүйнө билдирүүлөрү
4  """
5  print("Салам дүйнө")
```

Негизги функциялар.

Python бир катар камтылган функцияларды камсыз кылат. Алардын айрымдары, өзгөчө тил үйрөнүүнүн баштапкы этаптарында абдан көп колдонулат, ошондуктан аларды карап көрөлү.

Консолго маалыматты чыгаруунун негизги функциясы print() функциясы болуп саналат . Биз чыгаргыбыз келген сап бул функцияга аргумент катары берилет:

```
1  print("Hello Python")
```

Консолго бир нече маанилерди басып чыгаруу керек болсо, анда биз аларды үтүр менен бөлүп басып чыгаруу функциясына өткөрүп алабыз:

```
1  print("Толук аты-жөнү:", "Кубатов", "Нураалы")
```

Натыйжада, бардык өткөн маанилер боштуктар менен бир сапка жайгаштырылат:

Толук аты-жөнү: Кубатов Нураалы

Эгерде print функциясы чыгаруу үчүн жооптуу болсо, анда input функциясы маалыматты киргизүү үчүн жооп берет. Бул функция кошумча параметр катары киргизүү сунушун алат жана биз өзгөрмөдө сактай турган киргизүү сабын кайтарат:

```
1  name = input("Атыңызды киргизиңиз:")
2  print("Салам", name)
```

Консолдук чыгаруу:

Атыңызды киргизиңиз: Нураалы

Салам Нураалы

Өзгөрмөлөр. Маалыматтык типтер, типтештирүү. Маалыматты киргизүү жана чыгаруу.

Өзгөрмөлөр

Өзгөрмөлөр маалыматтарды сактоо үчүн. Python тилиндеги өзгөрмө аты алфавиттик белги же астын сызык менен башталышы керек жана алфавиттик-сандык белгилерди жана астын сызууну камтышы мүмкүн. Мындан тышкары, өзгөрмөнүн аты Python тилинин ачкыч сөздөрүнүн аталышы менен бирдей болбошу керек. Ачкыч сөздөр көп эмес, аларды эстеп калуу оңой:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	Raise	

Өзгөрмөнүн аты каалагандай болушу мүмкүн. Бирок аларды жазуу үчүн бир канча эрежелер бар:

Өзгөрмөгө болушунча маалыматтын маанисине жараша ат берүү керек.

Өзгөрмөнүн аты тилдин командалык сөздөрү менен дал келбеши керек.

Өзгөрмөнүн аты тамга же «_» символу менен башталышы керек, цифра менен эмес.

Өзгөрмөнүн аты боштуктан турбашы керек.

Мисалы, өзгөрмө түзөлү:

```
1 name = "Нураалы"
```

Бул жерде өзгөрмө name аныкталган, ал "Нураалы" сабын кармайт.

Python өзгөрүлмө аталыштын эки түрүн колдонот: camel case жана underscore notation.

Camel case өзгөрмө аталышындагы ар бир жаңы камтылган сөз баш тамга менен башталышын билдирет. Мисалы үчүн:

```
1 userName = "Нураалы"
```

Underscore notation өзгөрмө аталышындагы камтылган сөздөрдүн астын сызык менен бөлүнгөнүн билдирет. Мисалы үчүн:

```
1 user_name = "Нураалы"
```

Жана ошондой эле регистрдин сезгичтигин эске алышыңыз керек, андыктан name жана Name өзгөрмөлөрү ар кандай объекттерди көрсөтөт.

```
1 # эки башка өзгөрмөлөр
```

```
2 name = "Нураалы"
```

```
3 Name = "Акмат"
```

Өзгөрмөлөрдү аныктагандан кийин, биз аны программада колдоно алабыз. Мисалы, консолго анын мазмунун басып чыгарууга аракет кылалы.

```
1 name = "Нураалы"
```

```
2 print(name) # консолго name өзгөрмөнүн маанисин басып чыгарабыз
```

Маалымат типтери

Python программасында дайым колдонулуучу маалыматтын типтери:

Категория	Тиби	Түшүндүрмө
Сандар	int	Бүтүн
	float	Чыныгы сандар
	bool	Логикалык(True\False)
	complex	Татаал(комплекстик)
Ырааттуулуктар	str	“ ” – саптык(строки)
	list	[] – тизме(списки)
	tuple	() – кортеж
None	Type(None)	Бош объект

Чыныгы жашоодо биз кыймыл аракеттерди айланабыздагы предметтер же объекттер боюнча жасайбыз. Биз алардын касиеттерин өзгөртөбүз жана жаңы милдеттерди

аткартабыз. Компьютердик программалар дагы виртуалдык, цифралык жашоодо объекттерди ушундай окшоштукта башкарат. Объектиге-багытталган программалоо деңгээлине жеткенге чейин мындай объекттерди маалымат деп эсептейбиз. Албетте, маалыматтар түрдүү болот. Көбүнчө компьютердик программалар саптар жана сандар менен иштейт. bool, int, float, str типтерин карайбыз.

Логикалык маанилер

Bool тиби эки логикалык маанини билдирет: True (чын, туура) же False (жалган, ката). True мааниси бир нерсенин чын экенин көрсөтүү үчүн колдонулат. Ал эми False мааниси, тескерисинче, бир нерсенин жалган экенин көрсөтүп турат. Бул типтеги өзгөрмөлөрдүн мисалы:

```
1 isMarried = False
2 print(isMarried) # False
3
4 isAlive = True
5 print(isAlive) # True
```

Бүтүн сандар

int тиби бүтүн санды билдирет, мисалы, 1, 4, 8, 50.

```
1 age = 21
2 print("Жаш:", age) # Жашы: 21
3
4 count = 15
5 print("Саны:", count) # Саны: 15
```

Демейки боюнча, стандарттуу сандар ондук сандар катары каралат. Бирок Python экилик, сегиздик жана он алтылык сандарды да колдойт.

Сан экилик системаны билдирерин көрсөтүү үчүн сандын алдына 0b префикси коюлат :

```
1 a = 0b11
2 b = 0b1011
3 c = 0b100001
4 print(a)
5 print(b)
6 print(c)
```

Сан сегиздик системаны билдирерин көрсөтүү үчүн сандын алдына 0o префикси коюлат :

```
1 a = 0o7
2 b = 0o11
3 c = 0o17
4 print(a)
5 print(b)
6 print(c)
```

Сан он алтылык системаны билдирерин көрсөтүү үчүн, санга 0x префикси коюлат :

```
1 a = 0x0A
2 b = 0xFF
3 c = 0xA1
4 print(a)
5 print(b)
6 print(c)
```

Белгилей кетчү нерсе, кайсы системада консолго чыгаруу үчүн print функциясына санды өткөрсөк, ал демейки боюнча ондук санда чыгарылат.

Бөлчөк сандар

float тиби 1.2 же 34.76 сыяктуу калкыма чекит санын билдирет . Чекит бүтүн жана бөлчөк бөлүктөрүнүн ортосундагы бөлүүчү катары колдонулат.

```
1 h = 1.68
2 pi = 3.14
3 w = 68.
4 print(h) # 1.68
5 print(pi) # 3.14
6 print (w) # 68.0
```

Калкыма чекиттин саны экспоненциалдык белгилер менен аныкталышы мүмкүн:

```
1 x= 3.9e3
2 print(x) # 3900.0
3
4 x= 3.9e-3
5 print(x) # 0.0039
```

float санда 18 гана маанилүү белги болушу мүмкүн. Ошентип, бул учурда эки гана белги колдонулат - 3.9. Жана эгер сан өтө чоң же өтө кичине болсо, анда биз санды көрсөткүчтүн жардамы менен окшош белгилер менен жаза алабыз. Көрсөткүчтөн кийинки сан негизги санды көбөйтүү керек болгон 10 санынын күчүн көрсөтөт - 3,9.

Саптар

str тиби саптарды билдирет . Сап "салам" жана 'салам' сыяктуу бир же кош тырмакчага алынган символдордун ырааттуулугун билдирет. Python 3.x тилинде саптар Юникод символдорунун жыйындысын билдирет

```
1 message = "Hello World!"
2 print(message) # Hello World!
3
4 name = Нураалы
5 print(name) # Нураалы
```

Андан тышкары, сапта көп символдор болсо, биз аны бөлүктөргө бөлүп, коддун ар кандай саптарына жайгаштырсак болот. Бул учурда, бүт сап кашаага алынат, ал эми анын айрым бөлүктөрү тырмакчага алынат:

```
1 text = ("Laudate omnes gentes laudate"
2         "Магнификат в секула")
3 print(text)
```

Эгерде биз көп саптуу текстти аныктагыбыз келсе, анда мындай текст үч кош же бир тырмакчага алынат:

```
1 ""
2 Бул комментарий
3 ""
4 text = "" Бул
5 көп
6 сапты камтыган
7 текст
```

```
8    ""
```

```
9    print(text)
```

Үчтүк жалгыз тырмакчаларды колдонууда, аларды комментарийлер менен чаташтырбоо керек: эгерде үчтүк жалгыз тырмакчадагы текст өзгөрмөгө дайындалса, анда бул комментарий эмес, сап.

Саптагы башкаруу ырааттуулугу

Сап бир катар атайын символдорду камтышы мүмкүн - башкаруу ырааттуулугу. Алардын ичинен айрымдары:

\ : саптын ичине боштук кошууга мүмкүндүк берет

\' : саптын ичине бир тырмакча кошууга мүмкүндүк берет

\\" : саптын ичине кош тырмакча кошууга мүмкүндүк берет

\\n : Жаңы сапка жылат

\\t : өтмөк кошот (4 чегинүү)

Келгиле, бир нече ырааттуулукту колдонолу:

```
1    text = "Кабар:\\n\\\"Hello World\\\""
```

```
2    print(text)
```

Программанын консолдук натыйжасы:

Кабар:

"Hello World"

Мындай ырааттуулук кээ бир учурларда бизге жардам бере алат да, мисалы, сапка цитата коюу, таблица түзүү, башка сапка өткөрүү. Бирок алар да жолтоо болушу

мүмкүн. Мисалы үчүн:

```
1    path= "C:\\python\\name.txt"
```

```
2    print(path)
```

Бул жерде path өзгөрмө файлга кандайдыр бир жолду камтыйт. Бирок, саптын ичинде "\\n" символдору бар, алар башкаруу ырааттуулугу катары чечмеленет. Ошентип, биз төмөнкү консолдун натыйжасын алабыз:

c:\\python

name.txt

Мындай абалды болтурбоо үчүн r символу саптын алдына коюлат.

```
1    path = r"C:\\python\\name.txt"
```

```
2    print(path)
```

Сапка маанилерди коюу

Python башка өзгөрмөлөрдүн маанилерин сапка коюуга мүмкүндүк берет. Бул үчүн, саптын ичинде өзгөрмөлөр тармал кашааларга {} коюлат жана f символу бүт саптын алдына коюлат :

```
1    userName = "Нураалы"
```

```
2    userAge = 29
```

```
3    user = f"аты: {userName} жашы: {userAge}"
```

```
4    print(user) # аты: Нураалы жашы: 37
```

Бул учурда, userName өзгөрмөнүн мааниси {userName} ордуна киргизилет. Ошо сыяктуу эле, {userAge} ордуна , userAge өзгөрмөнүн мааниси киргизилет.

Мазмуну

Программалоого киришүү. Python программасына

киришүү жана интерпретаторун орнотуу.	3
Python программалоо тили.	4
Python интерпретаторун орнотуу.	6
Биринчи программа.	7
PyCharmды орнотуу	9
Visual Studioну орнотуу.	12
Pythonдо программалоого киришүү	14

Өзгөрмөлөр. Маалыматтык типтер, типтештирүү.

Маалыматты киргизүү жана чыгаруу.	15
Өзгөрмөлөр	15
Маалымат типтери	16
Консолго киргизүү жана чыгаруу	20
Тибин өзгөртүү	21
Сандар менен арифметикалык амалдар	23

Шарттуу туюнтмалар 28

if шарттуу оператору 30

match конструкциясы 33

Циклдер 35

while циклы 35

for циклы 37

Циклден чыгуу. break жана continue 39

Саптар, Тизмелер, кортеждер,сөздүктөр жана көптүктөр 40

Саптар 40

Тизме 47

Кортеждер 53

Сөздүктөр 55

Көптүктөр 60

Өзгөчө учурлар менен иштөө 62

Функциялар 65

Nonlocal 76

Рекурсивдүү функция 77

lambda, zip, map, filter, Enumerate, reduce функциялары жана list comprehension 78

Модулдар 84

Негизги орнотулган модулдар 87

random модулу 87

math модулу 89

datetime модулу 90

Файлдар менен иштөө

Файлдарды ачуу жана жабуу	94
Текст файлдары	96
CSV файлдары	98
shelve модулу	102
OS модулу	104
Объектке багытталган программалоо	
Класстар жана объекттер	105
Инкапсуляция	108
Декараторлор	110
Мурас	111
Полиморфизм	112
object классы. Объекттин сап чагылдырылышы	115
GUI графикалык интерфейс түзүү.	
turtle графикалык модуль	116
tkinter графикалык модуль	119