2 Python-ға кіріспе

Python-ның дамуы 1980 жылдардың аяғында голландиялық Гвидо ван Россумнан басталды және жүйе 1991 жылы шықты. Python- бұл Windows, Mae OS, Linux аппараттық платформаларында жұмыс істей алатын кроссплатформалық бағдарламалау тілі. Тіл жасаушылар Python-ды 21 операциялық жүйемен жұмыс жасайтын компьютерлерде қолдануға болатындығын атап өтті.

Python - бұл объектіге бағытталған тіл, бірақ егер C # -де объектілі-бағдарлы бағдарламалаудың негіздерін түсіну керек болса, Java бастаушы бағдарламашы үшін қиын болса, онда Python-да ОББі енгізу талғампаз және қарапайым. Python тілінен бағдарламалау жолын бастауға шешім қабылдаған адам үшін маңызды артықшылығы оның аудармашысының ақысыз таратылуы және https://www.python.org/ сайтында жүктеу мүмкіндігі бар.

2.1 Жобаны Python-да құру процессі

Python тілі - бұл белсенді дамып келе жатқан тіл, оның жаңа нұсқалары жылына бірнеше рет пайда болады, ал тілдің негізгі ерекшеліктері бағдарламаның жаңа шығарылымына байланысты емес. Бағдарламалау ортасын құру және сіздің алғашқы жобаңызды құру процесін қарастырайық.

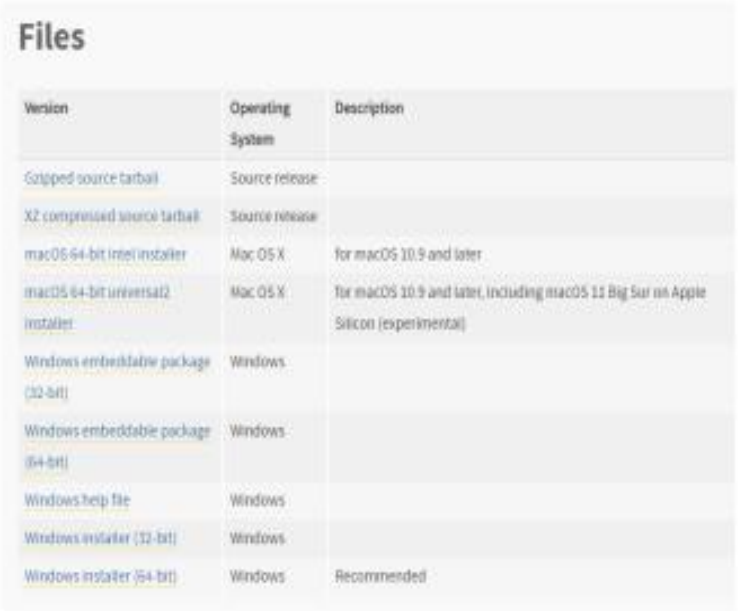
Енді web-ресурс www.python.org сайтына кірейік, сонда 4 суретте көрсетілген терезе ашылады.

Сурет 4 –www.python.org сайтының негізгі беті

Жүктеулер сілтемесін басу арқылы сіз компьютеріңіз жұмыс істеп тұрған операциялық жүйені ғана емес, сонымен қатар Python тілінің нұсқасын да таңдай аласыз (5-ші сурет).

Сурет 5 – Python тілінің нұсқасын таңдау беті

Python тілінің аудармашысы көптеген операциялық жүйелерде жұмыс істейді және кроссплатформалық болып табылады. Әр түрлі платформаларға арналған қондырғыларды(инсталяторларды) таңдау 6-шы суретте көрсетілген

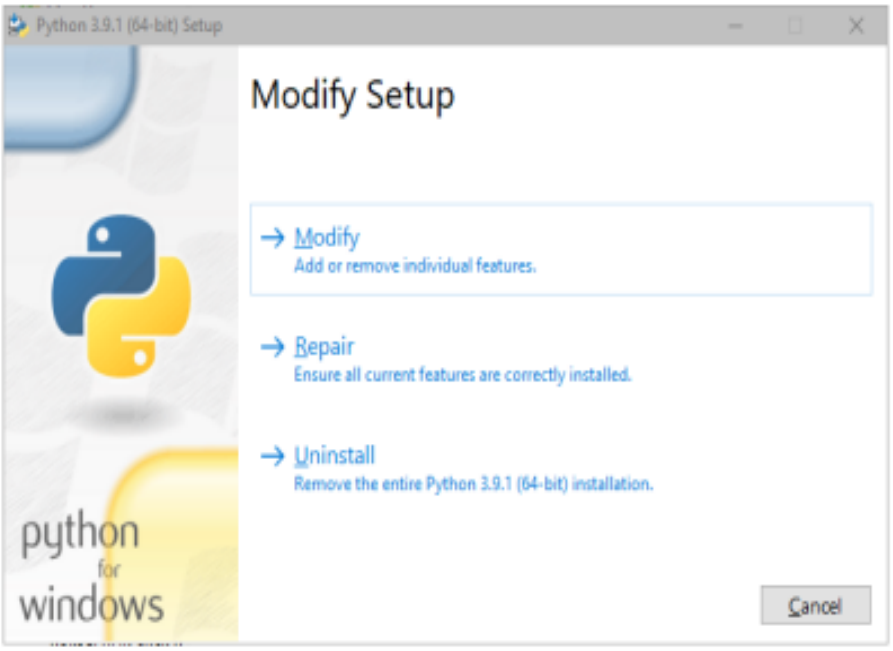


Сурет 6 – Әр түрлі платформаларға арналған Python қондырғыларының тізімі

Linux операциялық жүйесінде Python интерпретаторы үнсіз келісім бойынша орнатылған . Оны операциялық жүйенің кірістірілген депозитарийлерінен жаңартуға болады.

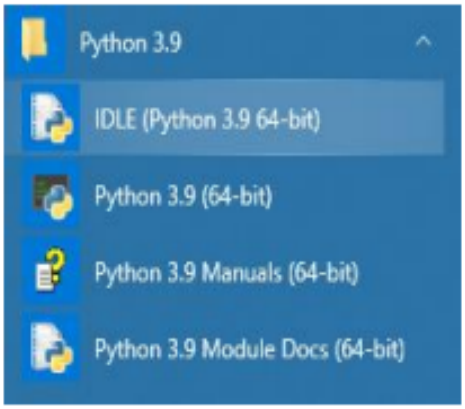
Android үшін Python бейімделуі QPython деп аталады және оны PlayMarket-тен жүктеуге болады. IOS үшін Python интерпретаторын AppStore-ден жүктеуге болады.

Windows операциялық жүйесінде жүктелген python-3.x.x.exe файлын басу арқылы сізге компьютерде Python орнатылғанша бірнеше минут күту керек болады (7-ші сурет).



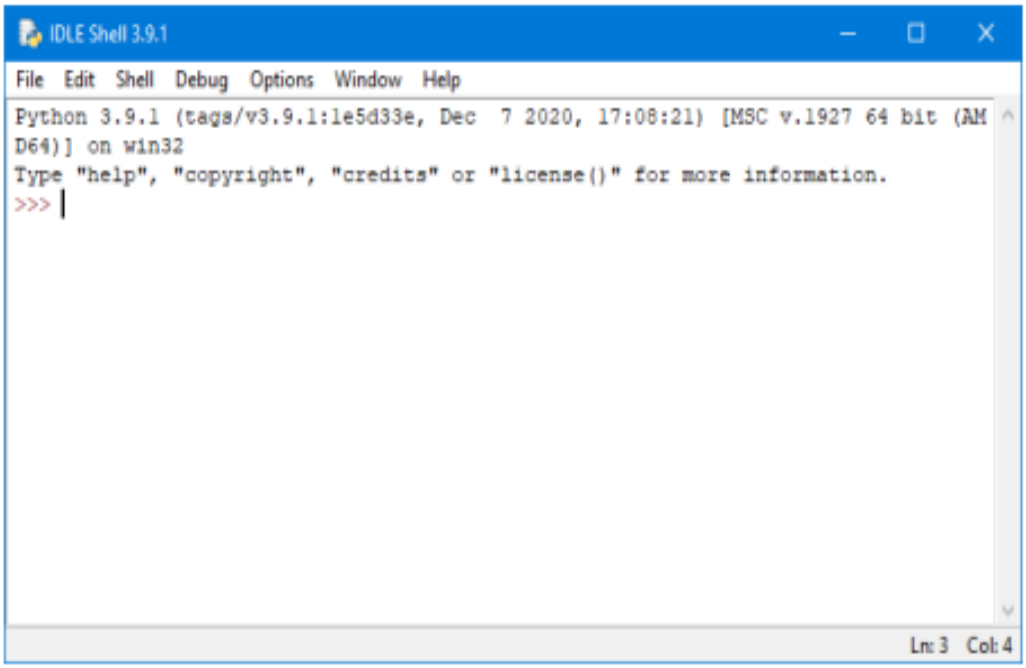
Сурет 7 –Python-ды орнату

Содан кейін, **Пуск/ Python 3.x** командасын орындағаннан кейін 8-ші суретте көрсетілген **IDLE (Python 3.x 32- bit)** жолын таңдап басу керек.



Сурет 8 – Орнатылған бағдарламаның таңбашасы(ярлык)

9-ші суретте көрсетілген терезе ашылады.



Сурет 9 –IDLE терезесінің ортасы

IDLE (Integrated Development Environment) - бұл Python тіліндегі интеграцияланған даму ортасы, оның көмегімен біз Python-да жазылған бағдарламаларды қарап шығамыз, өңдейміз, іске қосамыз және қалпына келтіреміз. IDLE Linux және UNIX тәрізді операциялық жүйелерде кеңінен қолданылатын **Tkinter** (Tk интерфейсі) деп аталатын графикалық кітапхананың көмегімен жасалған. IDLE - бұл Python-да жазылған белсенді терезе құралы, қайтару, синтаксисті пен ретке келтірушіні жарықтандыру, жұмысын жасайтын көп терезелі редактордың функцияларын сүйемелдеушісі.

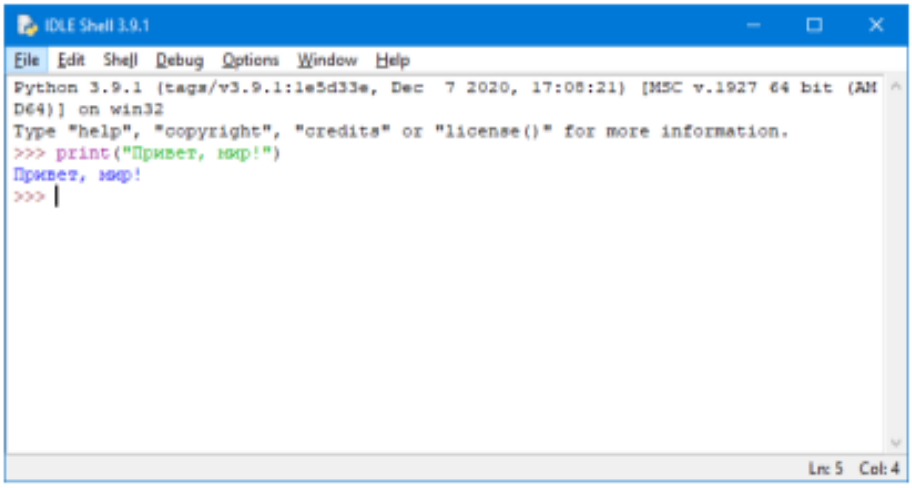
Қабықша мәзірінің элементтері осындай бағдарламалар үшін стандартты болып табылады, олар құрылған бағдарламаларды сақтау, ашу, редакциялау және күйін келтірудің стандартты функцияларын қолдайды, сондықтан біз оларға жеке сипаттама бермейміз, бірақ олармен тікелей IDLE-мен жұмыс істей отырып танысамыз.

Python-да алғашқы бағдарламамызды құрайық, яғни кез-келген тілде бағдарламалауға алғашқы қадам жасаған адамның басқа адамдарға бейресми сәлемі болып табылатын «**Сәлем, әлем!**» хабарламасын шығаратын бағдарлама. Біз оны **print** операторы мен осы оператордың синтаксисі арқылы жазамыз. Бағдарламалаудағы оператор (нұсқаулық) - бұл компьютердің бағдарламаның орындалуына жауап ретінде орындайтын әрекеті. Әрбір оператордың өзіндік синтаксисі бар, яғни белгілі бір бағдарламалау ортасында берілген операторды орындауға болатын жазу ережелері. **Print** операторында келесі синтаксис бар:

print("Хабарлама").

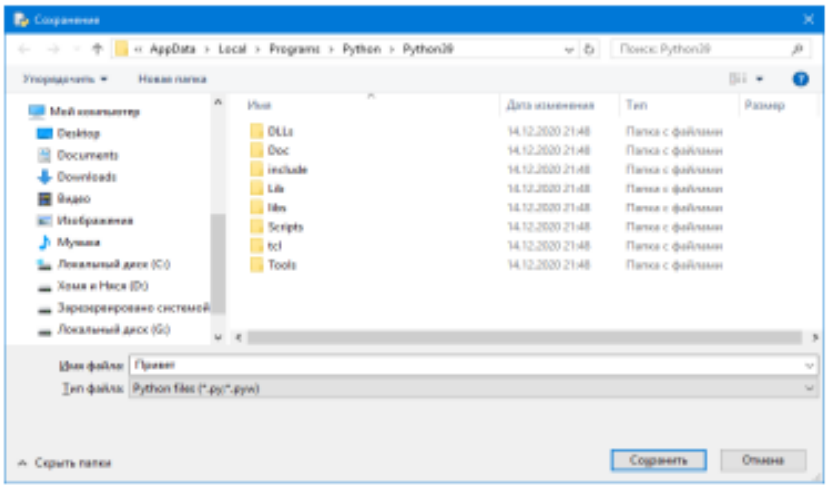
Python регистрді ескеретін болғандықтан, **print** операторы кіші әріппен жазылуы керек екенін ескеру қажет. Сонымен, **Print** немесе **PRINT** операторлары қате жазылған операторлар болып табылады.

Ендеше, сіз IDLE-де шақыру белгісі пайда болғанда >, яғни >>> үш еселенген белгіден кейін **print**(«Сәлем, әлем!») деп жазасыз. Нәтижені көру үшін Enter пернесін басу керек. Нұсқаулықтың нәтижесі, әрине, «Сәлем, әлем!» хабарламасын көрсетеді. (10-шы сурет). Python-да бағдарламалауға алғашқы қадам жасалды.



Сурет 10 – Бірінші жазылған бағдарламаның нәтижесі

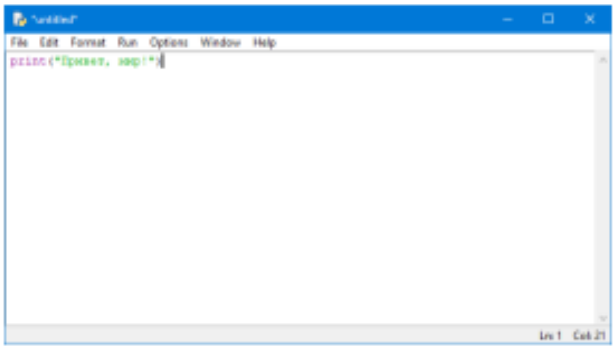
Бірінші бағдарламамызды сақтап алайық. Ол үшін **File/Save** командасын орындаңыз. 11-ші суретте көрсетілген терезе ашылады.Өз бағдарламаларыңызды дұрыс сақтау үшін бөлек папка жасағаныңыз жөн, бұл жағдайда оның **Мои проекты**(Менің жобаларым) деген атауы болады. Үнсіз келісім бойынша , егер сіз Windows операциялық жүйесінде болсаңыз, онда Python бағдарламасы келесі жолға орнатылады**: C: \ Users \ Computer Username \ AppData \ Local \ Programs \ Python \ Python39 \.**



Сурет 11 –«Қалай сақтау» терезесі

Сонымен, егер сіз болашақта өз бағдарламаларыңызды оларға бағыт сілтей отырып тапқыңыз келсе, онда сіз жоғарыда көрсетілген адрес бойынша орындайсыз. Енді өз бағдарламаларымызды IDLE Python-да құрудың тағы бір әдісін қарастырайық. Бірінші әдіс бізге жазған бағдарламаның нәтижесін бірден көруге мүмкіндік берді. Сонымен, жүйенің жедел реакциясы пайдаланушының операторлардың жазуына жауап ретінде пайда болатын мұндай интерактивті режим аса маңызды бағдарламаларды әзірлеу кезінде мүлдем ыңғайлы болмайды, өйткені болашақта жазылған кез келген бағдарлама осындай режимді түзету кезеңі қажет ететін болады. Сондықтан алдымен жоспарланған әрекеттерді бағдарламалық операторлар түрінде жазып, содан кейін программаны орындау үшін «іске қосу» керек.

Ол үшін IDLE-ді бұрын сипатталған тәсілмен ашып, File/New командасын орындаңыз. 12-ші суретте көрсетілген терезе ашылады.



Сурет 12 – Программасы бар терезе

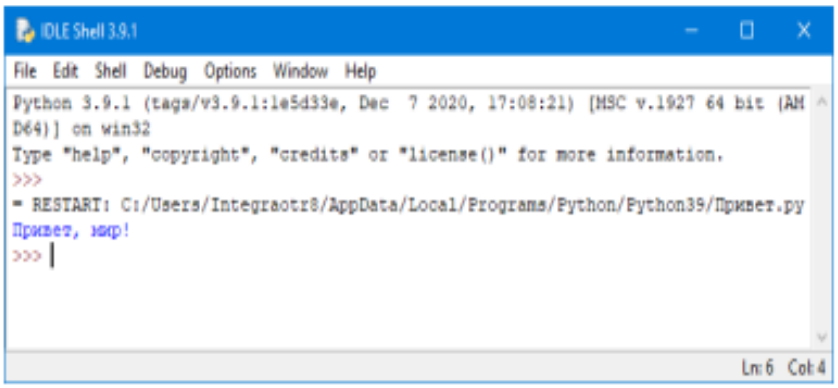
Дәл осы жерде біз print(«Сәлем әлем!») қайта жазамыз.

Ескерту. Операторды жазу кезінде терезенің сол жақ шекарасынан бос жол жасамаңыз. Мұның немен байланысты екендігі кейінірек түсіндіріледі.

Біздің бағдарламаны орындау үшін сізге **Run**(Іске қосу) мәзірінің пунктін және онда Run модулінің пунктін таңдау керек болады. Алайда, осыдан кейін бағдарламаның нәтижесі емес, жобаны сақтау туралы өтініш пайда болады (13-ші сурет). Болашақта бағдарламаңызға кез-келген өзгеріс енгізгеннен кейін, егер ол бұрын сақталған болса да, сіз өзгертулерді сақтауды сұрайтын терезе пайда болатынын және сіз тек Ok батырмасын басуыңыз керек екенін ескеріңіз.

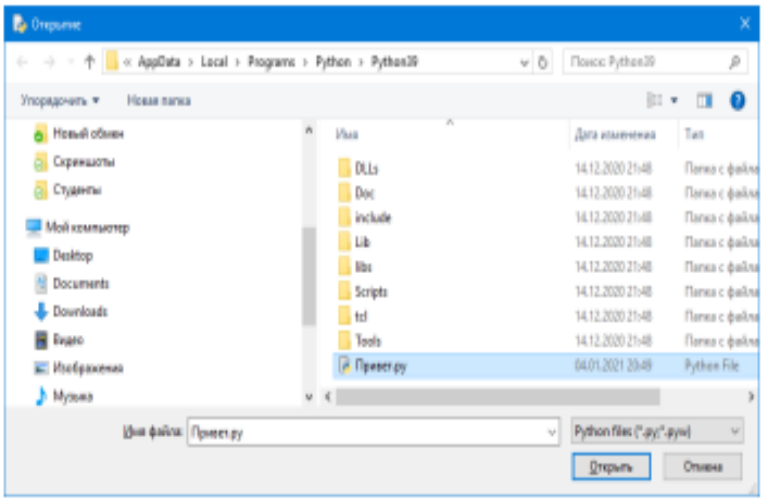
Сурет 13 – Программаны сақтау туралы сұрақ

Ok батырмасын басу арқылы сіз болашақ жобаларыңыз үшін бұрын жасаған бумаға(папака) жолды көрсетуіңіз керек. Сіз алдыңғы жобаны сақтап қойғандықтан, жоба атауын бұрынғыдай көрсете аласыз. Ұқсас жоба қазірдің өзінде бар екендігі және сіздің әрекеттеріңізді растауыңыз керек екендігі туралы терезе пайда болады. Енді Python ортасы (қабығы) ашылады, онда сіз бағдарламаның нәтижелерін көресіз, атап айтқанда «Сәлем әлем!» хабарламасын. (14-ші сурет).

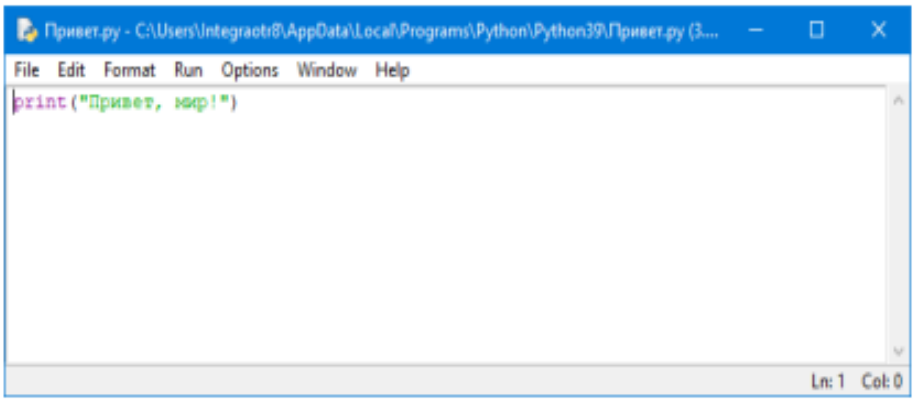


Сурет 14 – Программаның қорытынды нәтижесі

Қазірдің өзінде жасалған бағдарламаны ашу процесін қарастырайық. Start / Python 3.x командасын орындаңыз және IDLE (Python 3.x 32-разряд) жарлығын басыңыз. Пайда болған терезеде **File/Open** командасын орындаңыз, Python-да жазылған жобалар сақталатын бумаға жолды көрсетіп, «Сәлем, әлем!» файлын таңдаңыз. Осыған ұқсас жағдай 15-ші суретте көрсетілген. Біздің алдымызда сценарий режимі деп аталатын терезе тағы ашылады, онда біз Python-да бағдарламалар жазамыз (16-шы сурет). Онда сіз бұрын жасалған бағдарламаның мәтінін көресіз.



Сурет 15 – Ашу «Открыть» терезесі



Сурет 16 – Сценариалық режимнің терезесі

Сіз бұдан әрі не істеу керектігін білесіз, бірақ бағдарламаны іске қосу үшін **Run** мәзірінің тармағын таңдап, оның ішінен **Run** модулінің тармағын таңдаңыз. Бұл команда бойынша бағдарлама орындалады және оның нәтижесін көруге болады.

2.2. Деректерді енгізу және шығару әдістері және ерекшеліктерді өңдеу

Күрделі бағдарламаны жазу уақыты келді, оның нәтижесі пайдаланушы енгізген екі санның қосындысы болады. Оның мысалында біз деректерді енгізуге және айнымалыға қандай да бір мән тағайындауға мүмкіндік беретін жаңа операторларды қарастыра аламыз.

Меншіктеу операторы жүзеге асыратын әрекеттерін біз 1.6-параграфты оқу кезінде таныстық. Естеріңізге сала кетейік, Python тілінде оның түрі (=) белгісіне ие.

Input функциясы пайдаланушы пернетақтадан енгізетін мәндерді (деректерді енгізу) алу үшін бағдарлама кодтарында қолданылады. Оның келесі синтаксисі бар:

айнымалы атауы = **input** ("шақыру")

Мұндай жазба **print** операторының синтаксисіне өте ұқсас. Алайда айырмашылықтары бар. Ең алдымен, біз "функция" ұғымы туралы сөйлесетін боламыз. Python бағдарламалау тілін үйрену барысында біз функцияларды қолдану негізінде бағдарламалауды жеткілікті меңгеруіміз керек, функцияларды қарастырайық *:* sin(x) функциясын есептеу кезінде, мысалы, Microsoft Excel-мен жұмыс істейтін қолданушы **sin** функциясын формула/кірістіру/Математика және тригонометрия санатынан таңдайды және онымен жұмыс істей бастайды, осы функцияның мәнін есептейді.

Сонымен бірге, **SIN** функциясы-бұл формула бойынша есептеуге болатын күрделі ауыспалы математикалық қатар:

$$sinx = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \cdots$$

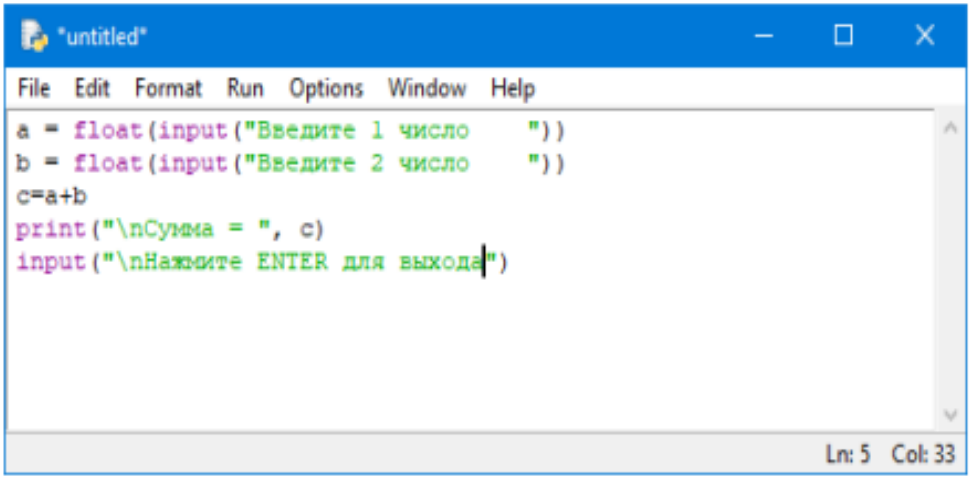
Бағдарламалау ортасында осы қатардың қосындысын есептеу үшін өте күрделі бағдарлама жазу керек. Математикалық ішкі бағдарламалар кітапханасында орналасқан бағдарламашылар жазған SIN функциясын шақыру және оның аты - SIN көмегімен жұмыс істеу оңайырақ.

Алайда, оны бағдарламалау ортасында қолданудан нәтижесін алу үшін Сіз оның **sin** атауын және жақшадағы аргументтің мәнін (sin(x)) көрсетіп қана қоймай, функцияның сол жағында айнымалы атауды, сонымен қатар меншіктеу операторын, яғни функцияны шақыру y=sin(x) операторы ретінде жүзеге асырылатындығын түсінуіңіз керек.

Енді "функция" ұғымы математикалық әрекеттер санатына ғана емес, басқаларға да қатысты екенін есте ұстаған жөн. Жедел жад ұяшығына пайдаланушыдан алынған кез-келген мәнді орналастыру үшін **input** функциясын қолданған кезде (бұл тапсырма меншіктеу операторының әрекетін еске түсіреді), сол оператордың сол жағында біз айнымалы атауды көрсетуіміз керек.

Жоғарыда айтылғандарды ескере отырып, = **input**("шақыру") айнымалы атауы ретінде жазылған **input** функциясының синтаксисі ешқандай сұрақтар тудырмауы керек.

Сценарий редакторында 17-ші суретте көрсетілген бағдарлама кодын жазыңыз.



Сурет 17 – «Екі санның қосындысына » арналған программа мәтіні

Бағдарламада нәтиженің шығуы синтаксисі экранға тек бір жолды шығарғанға қарағанда біршама өзгеше болатын **print** операторымен шығарылады, атап айтқанда **print**(«Шақыру», идентификатор), мұнда

1. *Шақыру* - шығару ақпараттың сипаттамасын қамтитын жол;

2. идентификатор - нәтижені басып шығаруға арналған айнымалының атауы. Бағдарламаның соңғы екі жолында қамтылған белгі **\n** интерпретаторға курсорды жаңа жолға жылжытуға нұсқау беретіндігін тағы да атап өтуге болады.

Екі санды қосыңыз бағдарламаның кодында, бұл кезеңде тек float функциясын пайдалану және жоғарыда келтірілген ойларға қарамастан, **input** функциясы айнымалы атауын сол жақта айтпастан шақырылатындығы түсініксіз

болуы мүмкін, ол жолда **input("\nНажмите ENTER для выхода")** меншіктеу операторы жоқ

Python-да қолданылатын мәліметтер типтерін түсіндіргенде (1.7 бөлімін қараңыз), сонымен қатар нақты бөлшектер түрі**(float)** де аталды, ол бағдарлама бөлшек сандармен жұмыс істеу керек болған кезде қолданылады. Сонымен, **input** функциясы жолдың деректер түрін қайтарады. Яғни, пайдаланушыдан алынатын санды компьютер сан ретінде емес, жол мәні ретінде қабылдайды, сондықтан біз жолдарда ешқандай математикалық амалдар орындай алмаймыз. Осылайша, **float** функциясы мәліметтердің бір түрін екіншісіне түрлендіру функциясы рөлін атқарады (бұл жағдайда жол түрін нақты түріне түрлендіру).

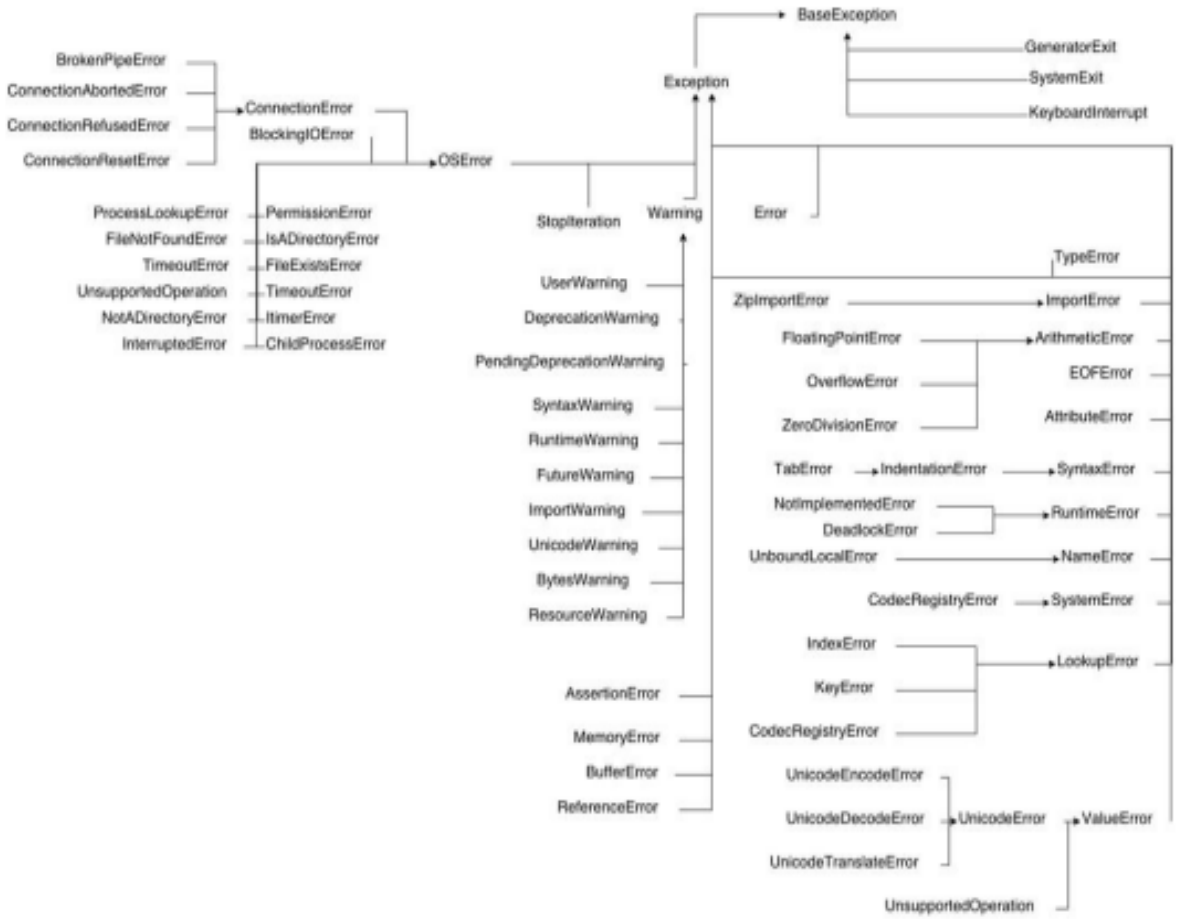
Оператор input("\nНажмите ENTER для выхода") міндетті емес. Егер ол бағдарламада болмаса, онда, әрине, оның нәтижесі математика тұрғысынан өзгермейді, сонымен қатар, нәтижені алып, ENTER пернесін басқаннан кейін бағдарламадан мұндай шығу болмайды. Ол тек шақыруға барады >>>. Осыған қарамастан, бұл жол пайдаланушыға оның бағдарламасы аяқталғанын және қандай да бір шешім қабылдау керек екенін айтады: не бағдарламалау ортасынан шығу керек, не бағдарламамен әрекеттерді жалғастыруды.

Сонымен қатар, осы операторда **input** функциясын қолдану қолданушыдан сандық мән алуға әкелмейді, тек ENTER пернесінің ішкі кодын оқумен ғана байланысты болады, ол сандарды қосудың математикалық операциясының нәтижесін тікелей өзгерте алмайды. Сонымен, енгізу функциясының басқа синтаксисі де мүмкін, ол мысалда көрсетілген. Мәліметтерді **input** және нәтижені шығару нәтижесі сурет 18 көрсетілген.

Сурет 18 – Программа жұмысының нәтижесі

Осы бөлімде әрі қарай қарастырылатын үзулер ішкі үзілістер класына жатады және **ерекшеліктер**(exceptions) деп аталады. Олар бағдарламаның орындалуымен синхронды түрде пайда болады және белгілі бір команданы орындау кезінде төтенше жағдай туындағанда пайда болады. Ерекшеліктердің мысалдары - нөлге бөлу, толып кету, жоқ файлға қол жеткізу және т.б.

Python-да қателерді өңдеуші **try...except...finally** блогын пайдаланады. Ерекше жағдай орын алған кезде **try...except блогы** кодтың бір бөлігін қоршау керек .**Finally** блок әрдайым орындалады, сондықтан оған қарамастан орындалуы керек операторлар нұсқаулығы орналастырылады. Бағдарлама әртүрлі себептермен өз жұмысын тоқтатуы мүмкін, сондықтан ерекшеліктердің саны өте аз. Python3-те ерекше иерархия 19-шы суретке көрсетілген.



Сурет 19 – Python3-тің сатылық(иерархиялық) ерекшеліктері

5ші -кестеде олардың арасында ең көп кездесетін ерекшеліктер келтірілген.

Кесте 5 - Ерекшеліктер түрлері

Тип исключений	Сипаттамасы
IOError	Енгізу-шығару операцияларында қателіктер пайда болған кезде пайда болады
IndexError	Тізбекте көрсетілген индексі бар элемент табылмаған кезде пайда болады
NameError	Егер айнымалы атауы, функция атауы табылмаса пайда болады
SyntaxError	Бағдарламада синтаксистік қате болған жағдайда пайда болады
TypeError	Стандартты операция сәйкес емес типтегі объектіге қолданылған кезде пайда болады
ValueError	Стандартты операция ,мәні сәйкес емес, тиісті типтегі объектіге

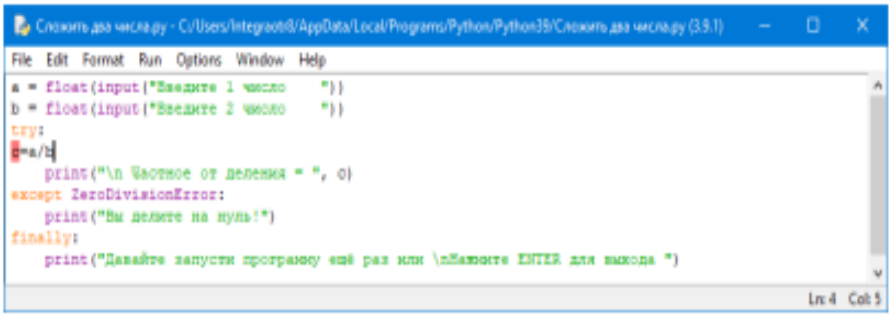
	қолданылған кезде орын алады
ZeroDivisionError	Нөлді бөлу операцияда орындалғанда пайда болады

try...except...finally әрекетті пайдаланғанға мысал келтірейік , екі санды бөлуден алынған мәнді алуға арналған программа құрыңыз және жазыңыз. Оның коды төмендегі листингте(тізімде) көрсетілген:

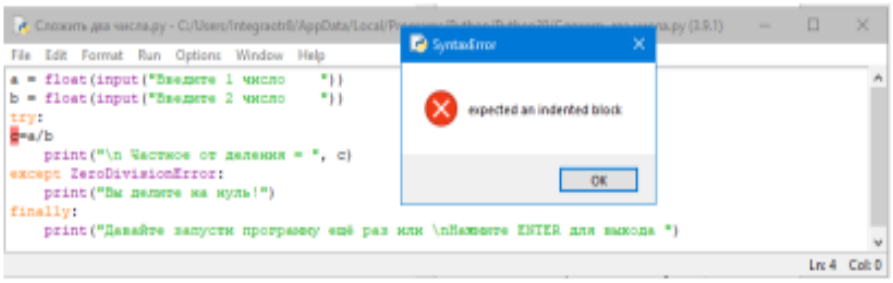
```
a=float(input("Введите 1 число  "))
b=float(input("Введите 2 число  "))
try:
   c=a/b
   print("\n Частное от деления = ", c)
except ZeroDivisionError:
   print("Вы делите на нуль!")
finally:
   print("Давайте запустим программу еще раз или \n Нажмите ENTER для выхода")
```

Осы кодтағы шегіністе назар аударыңыз. Мәселе мынада: C #, Microsoft Visual Basic, Паскальға бағытталған тілдер сияқты басқа танымал бағдарламалау тілдерінен айырмашылығы, мұнда бағдарламаны оқуды және қалпына келтіруді жеңілдету үшін код блоктарын шегіндіруге кеңес беріледі, бірақ олардың принципі қосымша; Python-да шегініс бағдарламалау құрылымдарының синтаксисінің бөлігі болып табылады. Кодтың сызығын **шегіндіру** үшін **Tab** пернесін басыңыз немесе **Space(пробел)** пернесін пернені төрт рет басыңыз.

Жоғарыда келтірілген кодтағы негізгі бағдарламалық құрылым - бұл **try...except...finally**. құрылымы болып табылады, сондықтан бұл блоктың ең болмағанда бір жолын шегіндірмеу керек, мысалы, 20-ші суретте көрсетілгендей (шегініс жоқ c=a/b операторында), ал бағдарлама 21-ші суретте көрсетілген қатені жіберіп, жұмысын тоқтатады.



Сурет 20 –c=a/b операторында шегініс жоқ

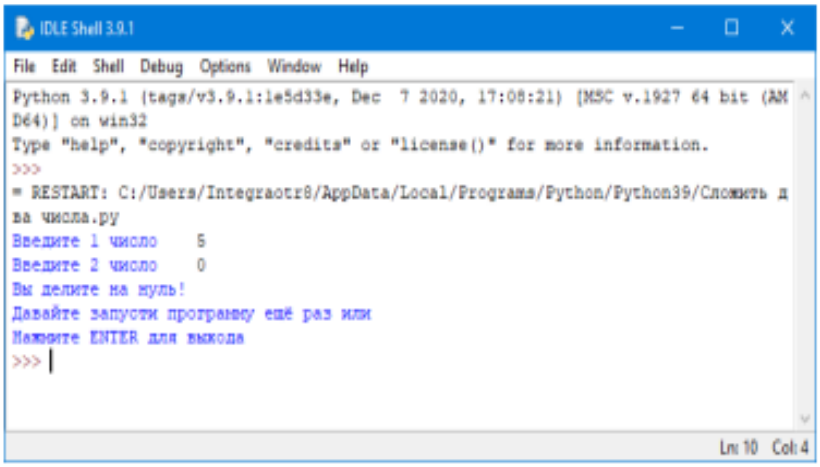


Сурет 21 – Шегініс болмағандықтан туындаған қате

Сонымен қатар, егер сіз сөйлемдер блогына жатпайтын бағдарлама кодының сызығын шегіндірсеңіз, мысалы, **a=float(input(«1 сан енгізіңіз»)),** онда бағдарлама сонымен қатар жұмысын тоқтатады, ал аудармашы қате көрсетеді. Демек, егер сіз осындай ережелерді сақтамасаңыз, жаңадан бағдарламма жазғанда көмексіз түсіну өте қиын болады. Болашақта біз белгілі бір бағдарламалық құрылыстарды шегіндіру қажеттілігі туралы арнайы түсініктемелер береміз.

Бағдарлама кодының түсініктемесіне оралыңыз. Біз кодты қоршап алдық, онда нөлге бөлу қатесі пайда болуы мүмкін, **try...except...finally** блогымен, онда күтілетін ерекшелік түрін қолдандық **ZeroDivisionError** (кестені 5 қараңыз.). Бұл блок нөлге бөлудің нәтижесінде пайда болатын белгілі бір толып кету қатесін, ұстап қалады.

Біз 5 санын енгіздік делік, ал екінші сан 0-ге тең. Бұл код орындалған кезде хабарлама пайда болады: «Сіз нөлге бөлесіз!». Бағдарламаның нәтижесі 22-ші суретте көрсетілген.



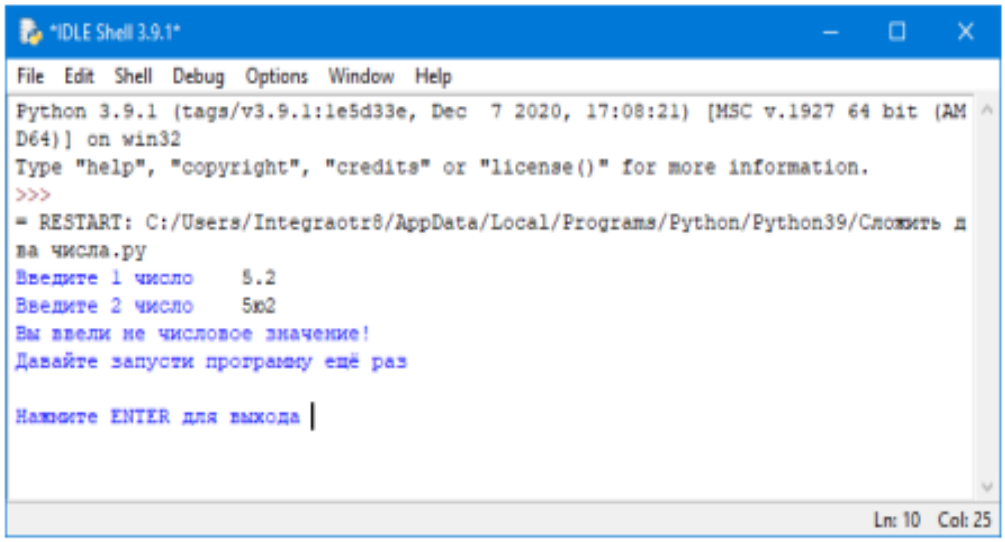
Сурет 22 – Программа хабарлама шығарады «Сіз нөлге бөлесіз!»

Бағдарламаны жақсартуға тырысайық. Біздің бағдарлама толығымен дұрыс емес екендігінде: пайдаланушы қате арқылы сандардың орнына пернетақтада орналасқан қарапайым белгілерді енгізуі мүмкін. Тізімде көрсетілген төмендегі код дұрыс енгізілмеген форматқа байланысты тағы бір қате жібереді. Онда, біріншіден, біз оператор **try**-ді осындай қате орын алуы мүмкін жерге орналастырамыз - бұл мәліметтер енгізу нұсқаулары. Екіншіден, біз бірнеше

ерекше жағдайларды өңдеуді **except** конструкциядан басқа бірнеше қосымшалармен ұстауға болатындығын және біздің кодқа **ValueError** ерекше жағдайын қосамыз (5-ші кестені қараңыз), ол егер мәні сәйкес емес объектіге, сәйкес типтегі стандартты операцияға қолданылса пайда болады, бірақ. Осылайша, біз келесі бағдарламалық кодты аламыз:

```
try:
   a=float(input("Введите 1 число  "))
   b=float(input("Введите 2 число  "))
   c=a/b
   print("\n Частное от деления = ", c)
except ZeroDivisionError:
   print("Вы делите на нуль!")
except ValueError:
   print("Вы ввели не числовое значение!")
finally:
   print("Давайте запустим программу еще раз")
   input("\n Нажмите ENTER для выхода")
```

Енді бағдарлама қате енгізуден сенімді қорғалған. Пайдаланушы бірінші санның мәнін 5.2 дұрыс енгізді делік, ал екінші санды енгізген кезде қолданушы қате жіберіп, нүктені бөлгіш ретінде емес, **ю** таңбасын енгізеді (өйткені нүкте де, **ю** таңба да бір пенеде орналасқан). Алайда, мұндай қате бағдарламаны тоқтату тұрғысынан алып тастауға әкелмейді. Ол орындалған кезде «Сіз сансыз мән енгіздіңіз!» деген хабарлама шығады (23-ші сурет).



Сурет 23 – Программа «Сіз сандық емес мән енгіздіңіз!» хабарламасы шығады

Сонымен, бірнеше ерекшеліктерді өңдеу мүмкін бірнеше кірістілген **except** конструкциясының көмегі арқылы. Бұл жағдай алдыңғы кодта көрсетілген

болатын. Тағы бір әдіс - келесі бағдарламада көрсетілгендей, үтірлермен бөлінген ерекше жағдайлардың түрлерін тізімдеу.

```
try:
   a=float(input("Введите 1 число  "))
   b=float(input("Введите 2 число  "))
   c=a/b
   print("\n Частное от деления = ", c)
except (ZeroDivisionError, ValueError):
   print("Вы делите на нуль или вы ввели не числовое значение!")
finally:
   print("Давайте запустим программу еще раз")
   input("\n Нажмите ENTER для выхода")
```

Бағдарламаның нәтижесі 23-ші суретте көрсетілгенге ұқсас болады, бірақ бағдарлама коды қысқартылған.

2.3 Бақылау сұрақтары

1. IDLE мақсаты туралы айтыңыз. IDLE бағдарламаларын құрудың қандай әдістерін білдіңіз?
2. Python-да жасалған қосымшалар үшін қандай мәліметтер енгізу және шығару операторлары қолданылады? Пайдаланылған операторларға синтаксис жазыңыз.
3. Дүниежүзілік қателермен жұмыс істеу тұжырымдамасын әзірлеу кезінде *try...except...finally* қашан қолданылады? Мысалдармен ерекше жағдайларды өңдеушілердің жұмысын түсіндіріңіз.
4. Ерекше жағдайлардың негізгі түрлерін атаңыз және олардың пайда болу себептерін көрсетіңіз.
5. Python-да жазылған бағдарламаларда шегініс қандай рөл атқарады?