Дәріс №6. Функциялар. Айнымалылардың ауқымы. Python-да генераторлар. Модулдер

Функциялар белгілі бір тапсырманы шешуге арналған код блоктары деп аталады. Функция ретінде анықталған тапсырманы орындау үшін сол тапсырмаға жауапты функцияны шақырасыз. Тапсырманы бағдарламада қайталап орындау қажет болса, барлық қажетті кодты қайта енгізудің қажеті жоқ; жай ғана қоңырау шалыңыз функция, мәселені шешуге арналған және бұл қоңырау Python-ға функцияның ішіндегі кодты орындауға нұсқайды.

Анықтама функциялары

Мұнда қарапайым функциясы бірге аты сәлемдесу_пайдаланушы (), қай көрсетеді сәлемдесу:

greeter.py

```
❶ def greet_user():
❷       """Выводит простое приветствие."""
❸       print("Hello!")

❹ greet_user()
```

Бұл мысал функцияның ең қарапайым құрылымын көрсетеді. Түзу ❶ def кілт сөзі Python-ға функцияны анықтап жатқаныңызды айтады. IN анықтамасы функциялары көрсетілген Аты функциялары Және Егер керек

— функцияның міндетін шешуге қажетті ақпараттың сипаттамасы. Бұл ақпарат жақшаға алынған. Бұл мысалда функция greet_user() деп аталады және оған қосымша қажет емес мәселеңізді шешуге арналған ақпарат, сондықтан жақшалар бос. (Алайда, бұл жағдайда да олар талап етіледі.) Соңында, анықтама қос нүктемен аяқталады.

Def greet_user(): параметрінен кейінгі барлық шегініс жолдар функцияның денесін құрайды. Нүктедегі мәтін ❷түсініктемені — функцияны сипаттайтын құжаттама жолын білдіреді. Құжат жолдары үш тырнақшаға алынады; Python бағдарламаларыңыздағы функциялар үшін құжаттаманы жасаған кезде оларды осы таңбалар тізбегі арқылы таниды .

Бұл функцияның мәтініндегі «нақты» код бір жолды басып шығарудан тұрады («Сәлеметсіз бе!») - төменде қараңыз. ❸. Осылайша, greet_user() функциясы тек бір тапсырманы шешеді: print("Hello!") командасын орындау.

Бұл функцияны пайдалану қажет болғанда, оны шақырыңыз. Функцияны шақыру Python-ға құрамындағы кодты орындауды айтады. Функцияны шақыру үшін оның атын және жолда көрсетілгендей жақшаға алынған барлық қажетті ақпаратты көрсетіңіз. ❹. Өйткені жоқ қосымша ақпарат Жоқ қажет қоңырау шалу функциялары

greet_user() пәрменін орындауға тең . Күтілгендей, функция Сәлем хабарын басып шығарады:

Сәлеметсіз бе!

Хабар тарату ақпарат функциялары

Бірнеше шамалы өзгерістермен greet_user() функциясы «Сәлеметсіз бе!» деп айтудан көп нәрсені істей алады. пайдаланушы, сонымен қатар атымен сәлемдесу. Бұл әрекетті орындау үшін def greet_user() функциясының анықтамасында жақшаға пайдаланушы атын енгізіңіз. Пайдаланушы атын қосқанда, функция шақырылған кезде жақшаға алынған кез келген мәнді қабылдайды. Функция енді әрбір қоңыраудың пайдаланушы атын беруін күтеді. greet_user() қызметін шақырған кезде жақшаның ішіне атауды (мысалы, 'jesse') енгізіңіз:

```
def greet_user(username):
    """Выводит простое приветствие."""
    print(f"Hello, {username.title()}!")

greet_user('jesse')
```

greet_user('jesse') пәрмені greet_user() функциясын шақырады және оған басып шығару пәрменін орындау үшін қажетті ақпаратты береді. Функция берілген атауды алады және сол ат үшін сәлемдесуді басып шығарады:

```
Hello, Jesse!
```

Сол сияқты, greet_user('sarah') пәрмені greet_user() функциясын шақырады және оған түзу 'сара' В нәтиже не ерік алынып тасталды хабар Сәлем Сара! greet_user() функциясын қалағаныңызша шақыруға және қалағаныңызша атау беруге болады, сонда сіз күткен нәтиже аласыз.

Аргументтер мен параметрлер greet_user() функциясы жұмыс істеу үшін пайдаланушы аты айнымалысының мәнін алуы керек етіп анықталған. Функция шақырылып, қажетті ақпаратты (пайдаланушы аты) алғаннан кейін ол дұрыс сәлемдесуді басып шығарады. greet_user() анықтамасындағы пайдаланушы аты айнымалысы параметр болып табылады, яғни шартты деректер, жұмысын орындау үшін қажетті функцияларды орындайды. greet_user('jesse') ішіндегі 'jesse' мәні аргумент болып табылады, яғни функцияны шақыру кезінде берілген нақты ақпарат. Функцияны шақырған кезде, функция жұмыс істеуі керек мәнді жақшаға қосасыз. Бұл жағдайда аргумент 'jesse' greet_user() функциясына жіберілді және оның мәні пайдаланушы аты айнымалысында сақталды.

Модульдер

Үшін құрастыру модульдер қажетті жасау файл бірге кеңейту .ру құрамында функциялар мен айнымалылар бар.

Функцияларды пайдалану үшін модульді басқа бағдарламаға импорттауға болады.

импорт sys
басып шығару('Аргументтер команда
жолдар:') sys.argv ішіндегі і үшін:
басып шығару(i)
басып шығару('\n\nАйнымалы PYTHONPATH құрамында',
sys.path '\n') Шығару: $ python using_sys.py біз аргументтерміз
Аргументтер команда сызықтар:
using_sys.
ру Біз
аргумент
т ер
болып
табылады
PYTHONPATH айнымалысы ['',
'C:\\Windows\\system32\\python30.zip', 'C:\\Python30\\DLLs',
'C:\\Python30\\lib', 'C: \ \ Python30\\lib\\plat-win',
'C:\\Python30', 'C:\\Python30\\lib\\site-packages']

Python бағдарламаны аударатын аралық пішін болып табылатын .рус кеңейтімі бар ***байт құрастырылған файлдарды (немесе байт кодты)*** жасауға болады .

Мұндай .рус файлы модульді басқа бағдарламаға келесі жолы ***импорттағанда пайдалы болады - ол әлдеқайда жылдамырақ болады*** , өйткені маңызды Модульді импорттау кезінде қажет кейбір өңдеулер ***қазірдің өзінде орындалады*** .

Бұл байт код Сондай-ақ болып табылады ***платформаға тәуелсіз*** .

Әдетте, .рус файлдары сәйкес .ру файлдарымен бірдей каталогта жасалады. Егер Python осы каталогтағы файлдарға жазу рұқсатын ала алмаса,
.рус файлдары жасалмайды.

Оператор бастап ... импорт ...

Argv айнымалы мәнін тікелей бағдарламаға импорттау және жазбау үшін кез келген бір рет sys. сағ емдеу Кімге ол, мүмкін қолдану өрнек **- бастап sys импорт argv‖** .

sys модулінде пайдаланылатын ***барлық атауларды*** импорттау үшін «sys import * файлынан» пәрменін орындауға болады.

Бұл жұмыс істейді ***Үшін кез келген модульдер***

Жалпы, бұл мәлімдемені пайдаланудан аулақ болуыңыз керек және атау қайшылықтарын болдырмау және бағдарламаңызды оқуды жеңілдету үшін оның орнына ***импорттау мәлімдемесін пайдаланыңыз.***

```
бастап математика
импорт * n = енгізу("
Енгізіңіз the ауқым: -
") p = [2, 3]
санау = 2
a = 5
кезінде
    (санау
    < n):
    b=0
    мен үшін жылы
        ауқым(2,a): егер (
        мен <= sqrt(a)):
            егер (a % i == 0):
                басып шығару("a емес a
                Премьер саны ",a) b = 1
            басқа:
    өту егер (б != 1):
        басып шығару("a
        Премьер саны ",a) p
        = p + [a]
    санау =
    санау + 1 a
    = a + 2
басып шығару б
```

Модуль атауы – _____аты_____

У барлығы модуль Сонда бар ***аты*** , Және ***командалар*** В модуль алады білу Аты олардың модулі.

Модуль ***өздігінен*** пайдаланылғанына немесе басқа бағдарламаға ***импортталғанына байланысты оның әрекетін басқаша жасауға болады.***

Бұл мүмкін жету бірге қолдану атрибут модуль астында аты ***__аты __.***

```
егер _____аты _== « _негізгі
    _____»: басып шығару ('Бұл
    бағдарламасы іске
қосылды өзі Авторы өзіңіз.') басқа:
    басып шығару('Мен импортталған В басқа модуль.')
Шығару:
```

$ питон3 пайдалану_аты.ру
Бұл бағдарламасы іске қосылды өзі Авторы өзіңізге.
$ питон3 >>> аты арқылы
импорттау Мен импортталған B
басқа модуль.
>>>

Жасау меншік модульдер

Әрбір Python бағдарламасы да модуль болып табылады. Тек оның .ру кеңейтімі орнатылғанына көз жеткізу керек.

Модульді басқа бағдарламаларда пайдалану үшін модуль не импортталатын бағдарламамен бір каталогта немесе sys.path ішінде көрсетілген каталогтардың бірінде болуы қажет.