**Дәрістер кешені**

**Дәріс №1**

**Дәріс тақырыбы:** Python бағдарламалау тілі туралы жалпы деректер. Python бағдарламалау тілінің шығу тарихы.

**Дәрістің мақсаты:** Python бағдарламалау тілі туралы жалпы деректер беру, тіл мүмкіндіктерін қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. Python бағдарламалау тілінің шығу тарихы.

2. Python тілінің мүмкіндіктері.

Python - жоғары дәрежелі кодтың оқылуын және әзірлеушінің өнімділігін арттыруға мақсатталған жалпы мақсаттағы [бағдарламалау](https://kk.wikipedia.org/wiki/%D0%91%D0%B0%D2%93%D0%B4%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B0%D0%BB%D0%B0%D1%83) тілі. Python тілі аз синтаксисті талап етеді. Бірақ сол уақытта стандартты кітапханасы үлкен көлемді пайдалы функцияларды қамтиды.

Тілдің кеңейткіштері: .py, .pyw, .pyc, .pyo, .pyd болып келеді.

Python тілін әзірлеу [голланд институтының қызметкері [Гвидо ван Россуммен](https://ru.wikipedia.org/wiki/%D0%93%D0%B2%D0%B8%D0%B4%D0%BE_%D0%B2%D0%B0%D0%BD_%D0%A0%D0%BE%D1%81%D1%81%D1%83%D0%BC) 1980 жылдың соңында басталған. Ол оны бос уақытында жаза бастаған. 1991 жылыдың 20 ақпанында алғашқы мәтіндерін жаңалықтар топтамасында жарыққа шығара бастады.](https://ru.wikipedia.org/wiki/1980-%D0%B5) Python тілі бастапқыдан объектіге бағытталған бағдарламалау тілі ретінде жобаланды.

Автор бағдарламаны [1970-жылдардағы](https://ru.wikipedia.org/wiki/1970-%D0%B5) танымал британ комедиялық «Монти Пайтон [Ұшатын](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D1%82%D0%B0%D1%8E%D1%89%D0%B8%D0%B9_%D1%86%D0%B8%D1%80%D0%BA_%D0%9C%D0%BE%D0%BD%D1%82%D0%B8_%D0%9F%D0%B0%D0%B9%D1%82%D0%BE%D0%BD%D0%B0) циркі» телешоуының құрметіне атаған. Көбі оны жыланның атымен байланыстырады. Бағдарламаның python.org сайтында (2.5 нұсқасына дейін) жыланның басы бейнеленген. Python әзірлеушісінің негізгі мақсаты - оны қолданушыға қызықты етіп жасау болды. Оны атауынан да байқауға болады. Оның бұл мақсаты бағдарламаны үйретуді ойын түрінде ұйымдастырып, ақпараттық материалдармен жабдықтағандығында. Бұл тілге деген қолданушылардың жақсы сын пікірлерінен Гвидоның дизайнерлік құрылымының да ұтымды болғанын дәлелдейді [13].

2008 жылдың 3 желтоқсанында ұзақ тестілеуден кейін Python 3000 (немесе Python 3.0) бірінші нұсқасы шыққан. Python 3000 бағдарламалау тілінде Python ескі нұсқаларымен сәйкестікті максимум сақтауға тырыса отырып, архитектурасы бойынша кемшіліктерді жойған. Қазіргі күні (Python 3.x және 2.x) екі даму бұтақтары да қолданып келе жатыр.

Python тілінің негізгі және аралық нұсқаларының уақыты:

* Python 1.0-Қаңтар 1994;
* Python 1.5 - 31 желтоқсан 1997;
* Python 1.6-5 қыркүйек 2000;
* Python 2.0 - 16 қазан 2000;
* Python 2.1 - 17 сәуір 2001;
* Python 2.2 - 21 желтоқсан 2001;
* Python 2.3-29 шілде, 2003;
* Python 2.4 - 30 қараша 2004;
* Python 2.5-19 қыркүйек, 2006;
* Python 2.6-1 қазан, 2008;
* Python 2.7-3 шілде, 2010;
* Python 3.0-3 желтоқсан, 2008;
* Python 3.1-27 маусым 2009;
* Python 3.2-20 ақпан, 2011;
* Python 3.3-29 қыркүйек, 2012;
* Python 3.4-16 наурыз, 2014.

Python тіліне басқа тілдердің ықпалы.

Python тілінің кешірек шыққандықтан оған көптеген тілдердің ықпалы болды. Мысалы келесі тілдердің:

* [ABC](https://ru.wikipedia.org/wiki/ABC_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) — операторларды топтаудың шегіністері, жоғары деңгейлі деректер құрылымы. (Python тілі шындығында келегенде, ABC ОББ тілін жобалауда кеткен қателіктерді түзету үшін құрылған тіл болатын);
* [Modula-3](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-3) — бумалары, модульдері;
* [С](https://ru.wikipedia.org/wiki/%D0%A1_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [C++](https://ru.wikipedia.org/wiki/C%2B%2B) — біршама синтаксистік конструкциялары;
* [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk) — объектіге бағытталған программалау;
* [Lisp](https://ru.wikipedia.org/wiki/Lisp) — (lambda, map, reduce, filter и другие) [функционалды программалаудың](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) айрықша белгілерін;
* [Fortran](https://ru.wikipedia.org/wiki/Fortran) —массивтер, кешенді арифметика;
* [Miranda](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D1%80%D0%B0%D0%BD%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) — [тізімдік өрнектер](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%9F%D0%B8%D1%82%D0%BE%D0%BD%D0%B5#%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F);
* [Java](https://ru.wikipedia.org/wiki/Java) — logging, unittest, threading модульдерін, xml.sax стандартты библиотекасын, finally және except-тің ескерпелерді өңдеудегі біріктірілген қолданылуын;
* [Icon](https://ru.wikipedia.org/wiki/Icon_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) — [генераторларын](https://ru.wikipedia.org/wiki/Python#%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B) [14].

Python тілінің басым бөлігі (мысалы, бастапқы кодтың байт-компиляциясы) бұрынырақта басқа бағдарламалау тілдерінде іске асырылатын.

2. Python тілінің мүмкіндіктері

Python тілін бағдарламалауда қолданудың қажеттігін дәлелдейтін көптеген мүмкіндіктері бар. Олар келесілер:

* xml/html файлдарымен жұмыс жасау;
* http сұраныстар жасау;
* GUI (графикалық интерфейс)
* Веб-сценарийлер құру;
* FTP-мен жұмыс жасау;
* Кескіндермен, аудио және видеомен жұмыс жасау;
* Робототехникада қолдану;
* Математикалық және ғылыми есептеулерді бағдарламалау және т.б.

Соған орай python бағдарламасы көптеген міндеттерді шешеді: резервті көшіру болсын, электронды поштаны оқу болсын немес қандай да ойынды құру болсын. Python бағдарламалау тілі ештеңемен шектелмегендіктен оны үлкен жобаларды пайдалануға болады. Мысалы ретінде, python тілі IT-дің алыптары болып табылатын, Google және Yandex жобаларында кеңінен қолданылады. Сонымен қатар, оның қарапайымдылығы мен әмбебаптығы бұл тілді ең үздік бағдарламалау тілдері қатарына қосады [15].

Тілін жүктеу және орнату. Python 3 бағдарламасын өз компьютеріңізге жүктеуді қарастырамыз. Ол тегін және еш тіркелусіз орындалады.

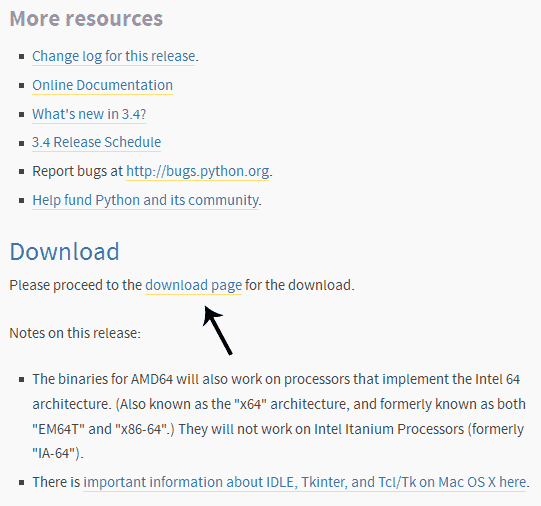
## Windows жүйесіне Python тілін орнату

Алдымен оны ресми сайттан жүктеп алу керек. Басқа сайттардан көшіру қауіпті, яғни вирустық программалар болуы мүмкін. Программа тегін. Келесі сайттың адресіне кіру керек: <https://python.org/downloads/windows/>, ішенен "latest python release" және python 3 таңдау керек.



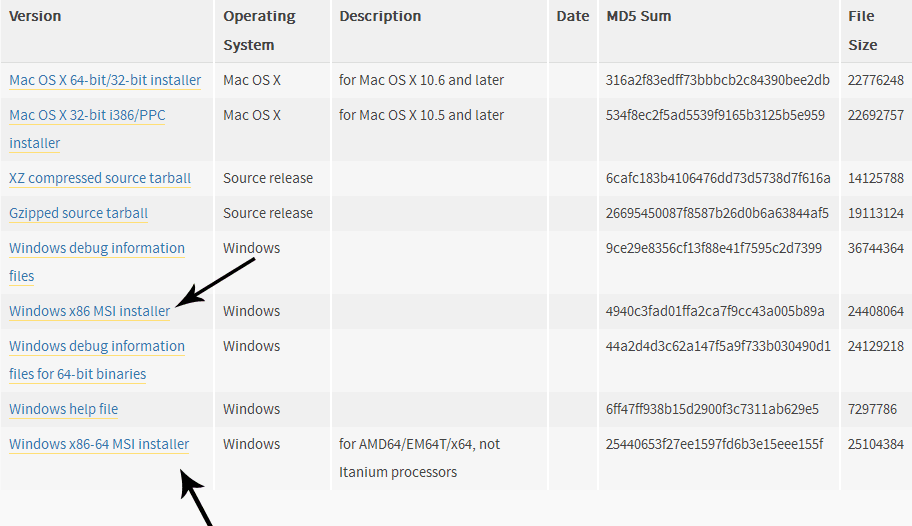
Сурет 1 Бағдарламаны көшіруге арналған сайттың алғашқы бетінен үзінді

Таңдаған нұсқа бойынша сипаттамалары жазылған терезе пайда болады. Беттің соңына дейін көшіп, "download page" ашамыз.



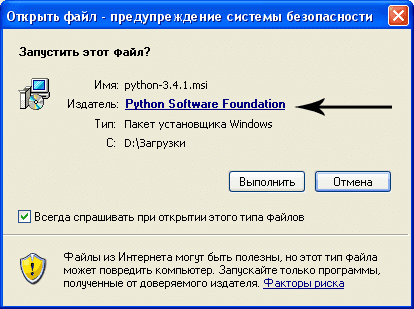
Сурет 5 Жүктеуге болатын файлдар тізімі

Бұл жерден жүктеуге болатын файлдар тізімін байқауға болады. Сіздерге егер жүйе 32- биттік болса, онда Windows x86 MSI installer, егер жүйе 64-биттік болса, онда Windows x86-64 MSI installer файлын таңдау керек.



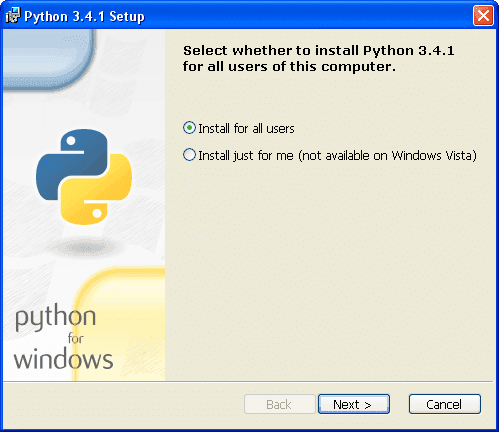
Сурет 2 Жүктеуге арналған файлдар тізімі

Python тілі жүктелгенше күтеміз. Жүктелген файлды ашу керек. Шығарушы қатарында Python Software Foundation жазуын көрсеңіздер, онда дұрыс таңдалды. Басқа жазу тұрса, ондай файлды ашпаған жөн [15].



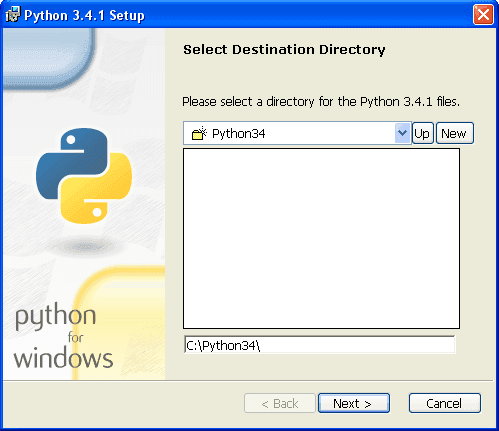
Сурет 3 Хабарлама терезесі

Барлық қолданушыларға немесе тек бір қолданушыға орнатуды орындаймыз (өз қалауларыңызбен орнату керек).



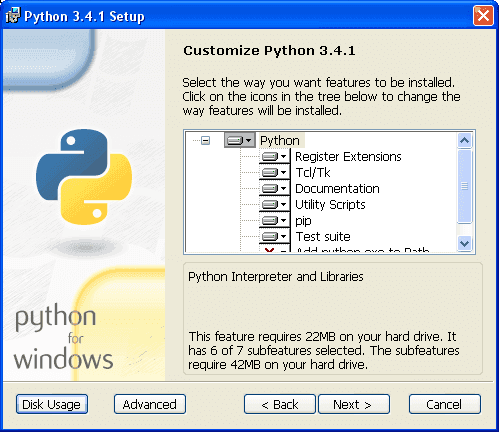
Сурет 4 Орнату жағдайын таңдау терезесі

Орнатуға арналған буманы таңдау керек. Дискіден кез келген буманы таңдауға болады.



Сурет 5 Орнататын буманы таңдау терезесі

Компонентерін таңдауға болады. Егер таңдау білмеген жағдайда үнсіз келісім бойынша қалдыру керек.



Сурет 6 Компоненттерді таңдау терезесі

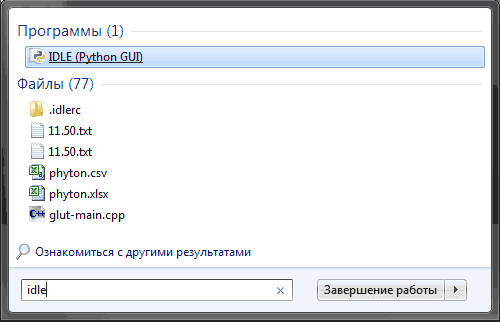
Python бағдарламасының орнатылғанын күту керек.

Finish батырмасын басқан соң бағдарлама орнатылды деп есептеу керек. Бұл нұсқада IDLE «зірлеу ортасы ендірілген. Дәл қазір алғашқы бағдарлама жазуға болады.

Енді алғашқа бағдарламаны кез келген мәтіндік редакторда немесе IDLE әзірлеу ортасында жазуға болады.

# IDLE әзірлеу ортасы. Алғашқы бағдарлама

Python бағдарламасын іске қосқан соң IDLE ортасын ашу керек (Python бағдарламалау тіліндегі әзірлеу ортасы).



Сурет 7 IDLE ортасын іске қосу терезесі

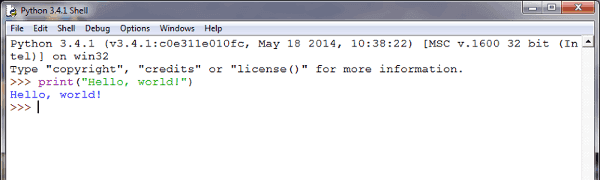
IDLE ортасы бастапқыда интерактивті режімде ашылады. Кейіннен программаны жазуды бастауға болады. Салт бойынша алғашқы программа "hello world" болады.

Python тілінде "hello world" сөз тіркесін жазу үшін тек бір ғана жолдың жазылуы жеткілікті:

print("Hello world!")

Сурет 8 Python тілінде жазылған код

Бұл кодты IDLE ортасына енгізіп Enter батырмасына шертеміз.Нәтиже келесә суретте бейнеленген:



Сурет 9 Интерактивті режім терезесі

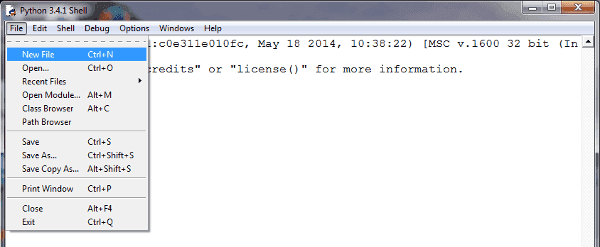
Осымен интерактивті режіммен таныс болдық. Келесі кодтарды да жазып тексеріп көруге болады.

print(3 + 4)

print(3 \* 5)

print(3 \*\* 2)

Негізінде интерактивті режім ең негізгісі болып табылмайды, сондықтан көбінде бағдарламалық кодтарды файлға жазып, файлды тексеруге жіберіп отырып жұмыс жаслады. IDLE интерактивті режімінде жаңа файл құру үшін File → New File (немесе Ctrl + N пернелер комбинациясын басу керек) [16].



Сурет 10 Жаңа файл құру терезесі

Ашылған терезеде келесі 15-суреттегі код теріледі:

name = input("Атыңыз кім? ")

print("Сәлем,", name)

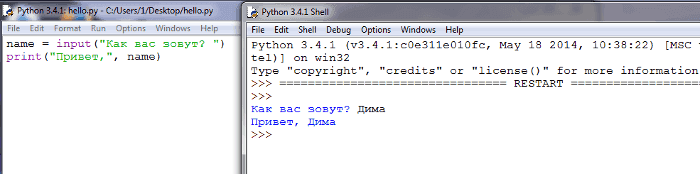
Сурет 11 Алғашқы бағдарлама коды

Бірінші жол енгізуді талап етеді, яғни пернетақтадан жазған мәліметті енгізіп, Enter пернесін шерткенде енгізілген мәліметті name айнымалысына сақтайды..

Келесі жолда print функциясы қолданылады. Ол экранға Сәлем сөзінен кейін "name" айнымалысында сақталған сөзді шығарады.

Функционалды пернелерден F5 (немесе мәзірден IDLE Run → Run Module командасын таңдап) басу және шыққан нәтиженің дұрыстығын тексеру керек.

Келесі суретте бейнеленген скриншотта оң бағанында нәтиже, сол жақ бағанында жазған бағдарлама көрсетіледі.



Сурет 12 Бағдарламаның нәтижесін шығару терезесі

Сонымен қарапайым бағдарламаның құрылуы іске асырылды. Енді тілдің синтаксисін қарастыруға болады.

# Python тілінің синтаксисі. Негізгі ережелері:

* Жолдың соңы интрукцияның соңы болып табылады (нүктелі үір қажет емес);
* Ендірілген инструкциялар шегіністер өлшемі (көлемі) бойынша блоктарға біріктіріледі. Шегініс кезкелген болуы мүмкін, бастысы бір ендірілген блокқа бірдей шегініс өлшемі қолданылса болғаны. Кодтың оқылымы туралы да ұмытпау керек. 1 бос орын шегініс - ол ешқандай шешім болып табылмайды. Тым болмағанда 4 бос орын шегініс немесе табуляция белгісін қолданған дұрыс болады.
* Python тілінде әр инструкциялар бір шаблонға сәйкес жазылады, негізгі инструкция қос нүктемен аяқталған соң, оның соңынан инструкцияның ендірілген блогы шегініс арқылы орналасады. Келесі суретте жазылуы бейнеленген:
* Негізгі инструкция:

Инструкцияның ендірілген блогы

## Сурет 13 Инструкцияның жазылуының бейнесі

## Бірнеше арнайы жағдайлар бар. Олар:

1. Кейбір жағдайда бірнеше инструкцияны бір жолға нүктелі үтір арқылы жазуға болады:

a = 1; b = 2; print(a, b)

## Сурет 14 Бірнеше инструкцияны бір жолға жазу

Бұндайды көп қолдануға болмайды, әрқашан да оқылым туралы есте сақтау керек.

1. Бір инструкцияны бірнеше жолға да бөліп жазуға болады. Ол үшін бірнеше доғал, квардатты немесе жүйелі жақшаны қолдану керек.

if (a == 1 and b == 2 and

c == 3 and d == 4): *# Не забываем про двоеточие*

print('spam' \* 3)

Сурет 15 Бір инструкцияны бірнеше жолға бөліп жазу коды

1. Құрамдас интсрукцияның денесі сол негізгі дененің жазылған жолына орналасуы мүмкін, егер негізгі инструкцияның денесі құрамдас денені қамтымаса. Мысалы:

if x > y: print(x)

Сурет 16 Құрамдас және негізгі инчтрукцияның денесінің бір жолға жазылуы

Бағдарламалау тілінің толық синтаксисін түсіну үшін көптеген мысалдарды қарастыру керек. Бірнеше есептердің шығарылуын қарастырып барып түсінуге болады.

**Дәріс №2**

**Дәріс тақырыбы:** Негізгі Стандартты Python модульдері.

**Дәрістің мақсаты:** Негізгі Стандартты Python модульдерін қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

* 1. Модуль түсінігі.

1. Модуль түсінігі.

Стандартты кітапхананың модульдерін зерттеуге кіріспес бұрын, Python-да модуль деп аталады.

Модульдік тәсілге сәйкес бағдарламалау үлкен міндет бірнеше ұсақ болып бөлінеді, олардың әрқайсысын (Идеалда) жеке модуль шешеді. Әр түрлі әдістемелерде модуль өлшеміне әртүрлі шектеулер беріледі, алайда бағдарламаның модульдік құрылымын құру кезінде модульдердің композициясын құру маңызды, ол олардың арасындағы байланысты барынша азайтуға мүмкіндік береді. Өз элементтері арасында көптеген байланыстары бар сыныптар мен функциялар жиынтығы бір модульде қисынды орналастыру болар еді. Тағы бір пайдалы ескерту бар: модульдерді қайта жазудан гөрі пайдалану оңай болуы керек. Бұл дегеніміз, модуль ыңғайлы интерфейс болуы керек: функциялар жиынтығы, сыныптар және тұрақты, ол өз пайдаланушыларына ұсынады.

Python тілінде бір мәселеге арналған модульдер жиынтығын пакетке қоюға болады. Мұндай пакеттің жақсы мысалы-XML пакеті, онда XML өңдеудің әр түрлі аспектілеріне арналған модульдер жинақталған.

Python бағдарламасында модуль Модуль-нысан-модуль, Оның атрибуттары модульде анықталған атаулар болып табылады:

>>> import datetime

>>> d1 = datetime.date(2004, 11, 20)

Бұл мысалда datetime модулі импортталады. Import операторының жұмысы нәтижесінде ағымдағы атау кеңістігінде datetime атымен объект пайда болады.

Python тіліндегі бағдарламаларда қолдану үшін модульдер өзінің шығу тегі бойынша кәдімгі (Python-да жазылған) және басқа бағдарламалау тілінде жазылған кеңейту модульдеріне (әдетте C-да) бөлінеді. Пайдаланушы тұрғысынан олар жылдам әрекетпен ерекшеленуі мүмкін. Стандартты кітапханада модульдің екі нұсқасы бар: Python және C.бұл, мысалы, pickle және cPickle модульдері. Әдетте Python модульдері кеңейту модулдеріне қарағанда икемді.

Python модульдері

Модуль бастапқы коды бар жеке файл түрінде ресімделеді. Стандартты Модульдер тілдің тиісті интерпретаторын таба алатын каталогта болады. Python модульдерді іздейтін каталогтарға жолды sys айнымалысы мәнінен көруге болады.path:

>>> sys.path

['', '/usr/local/lib/python23.zip', '/usr/local/lib/python2.3',

'/usr/local/lib/python2.3/plat-linux2', '/usr/local/lib/python2.3/lib-tk',

'/usr/local/lib/python2.3/lib-dynload',

'/usr/local/lib/python2.3/site-packages']

Соңғы Python нұсқаларында модульдерді жинақы сақтау үшін zip-мұрағатына (Java-дағы jar-мұрағаттарға ұқсас) орналастыруға болады.

Модульдерді іздеу бағдарламасын іске қосқан кезде, ағымдағы каталогта жүреді. (Стандартты немесе қосымша орнатылған модульдермен аттар қақтығысы болмас үшін өз модульдерін Мұқият атауыңыз керек.)

Python бағдарламасына модульді қосу import операторының көмегімен жүзеге асырылады . Оның екі нысаны бар : import және from-import:

import os

import pre as re

from sys import argv, environ

from string import \*

Бірінші пішін көмегімен ағымдағы көріну аймағымен тек модуль объектісіне сілтеме жасайтын атау ғана байланысады, ал екіншісін пайдаланған кезде модуль объектілерінің көрсетілген аттары (немесе қолданылса, барлық аттары \* ) ағымдағы көріну аймағымен байланысады. Импорттау кезінде, as арқылы элементтің атын өзгертуге болады. Бірінші жағдайда модуль атаулары кеңістігі бөлек атауда қалады және модульден нақты атауына кіру үшін нүктені қолдану қажет. Екінші жағдайда аттар ағымдағы модульде анықталғандай қолданылады:

os.system("dir")

digits = re.compile("\d+")

print argv[0], environ

Модульді қайта импорттау әлдеқайда жылдам жүреді , себебі Модульдер интерпретатормен кэштеледі. Жүктелген модульді reload функциясы арқылы тағы да жүктеуге болады (мысалы, модуль дискіде өзгерсе)():

import mymodule

. . .

reload(mymodule)

Бірақ бұл жағдайда модульдің ескі нұсқасынан сыныптардың даналары болып табылатын барлық объектілер өз мінез-құлқын өзгертпейді.

Модульдермен жұмыс істеу кезінде басқа да қырлары бар . Мысалы, модульді импорттау процесін қайта анықтауға болады. Бұл туралы толығырақ түпнұсқа құжаттамадан білуге болады.

**Дәріс №3**

**Дәріс тақырыбы:** Python тіліндегі жолдар.

**Дәрістің мақсаты:** Python тіліндегі жолдар қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

* 1. Python тіліндегі жолдар.

1. Python тіліндегі жолдар

Жолдар*.* Жолдағы бос орындарды немесе бірнеше бос орындарды «\*» таңбасымен алмастыруды қарастырайық.

Бұндай есепті шығару үшін бірінші алгоритмін құрып алу керек. Егер жолда бірнеше бос орын кездессе оны бір «\*» таңбасымен алмастырып, ал жолдың басы мен соңындағы бос орындарды ою керек.

**Есепті шығару алгоритмі:**

Жолды басынан бастап әр таңбасын қарап шығу. Бірінші тұрған бос орындық емес таңбаның нөмірін есте сақтау қажет. Жолдың босорындық емес бірінші таңбасынан соңына дейін бір бөлікті алу.

Жолды соңынан бастап қарап шығу. Жолдың соңынан бастағандағы босорындық емес таңбаның нөмірін есте сақтау. Жолдың басынан бастап соңынан санағандағы бірінші босорындық емес таңбаға дейінгі жолдың бір бөлігін алу.

Жаңа жолға ескі жолдың бірінші таңбасын меншіктеу (ол барлық жағдайда бос орын емес).

Ескі жолдың әр таңбасын қарап шығу. Егер келесі таңба бос орын болмаса, онда жаңа жолды осы таңбаны қосу арқылы форматтау керек. elif  бұтасына бос орын таңбалары орналасады, бірақ бұл кезде бұрынғы таңба бос орынға тең болды ма сол тексеріледі. Егер олай болмаса, онда тек қана сонда жаң жолға "\*" қосылады. Артық бос орындар жаңа жолға қосылмай алып тасталады.

Мысалы:

s = input()

 i = 0

while s[i] == ' ': i+=1

s = s[i:]

i = len(s)

while s[i-1] == ' ': i-=1

s = s[:i]

s1 = s[0]

i = 1

while i < len(s):

if s[i] != ' ':

s1 += s[i]

elif s[i-1] != ' ':

s1 += '\*'

i += 1

print(s1+'!')

Скриптің орындалу мысалы:

one two three four f\*ve six seven

one\*two\*three\*four\*f\*ve\*six\*seven!

Бірақ Python бағдарламалау тілінде жолды тізімге айналдыру арқылы есепті шешу немесе тұрақты өрнектер арқылы шығару оңайға түседі.

Бұндағы split() әдісі арқылы жолдар бос орындар бойынша сөздерге бөлінеді. Бұл жерде олардың саны маңызды емес. Ары қарай тек сөздерді қайта жолға "\*" таңбасы арқылы жинақтау қалады:

s = input()

l = s.split()

s1 = ''

for i in l:

s1 += i + '\*'

s1 = s1[:-1] *#* соңғысын жоямыз "\*"

print(s1)

Жолдардан санды таңдап алу. Натурал сандар мен сөздерден тұратын жолдар берілген. Бұл жолдардағы сандардан тізім құрау керек. Ол үшін жолдарды әрбір таңбасы бойынша қарап шығу керек. Егер келесі таңба сан болса оны жаңа жолға қосу керек. Ары қарай одан кейінгі таңбаларды қарап шығу керек. Егер олар да сандар болса, оларды да жолдың соңына тіркеп отыру қажет. Келесі таңба сан емес болып шыққан жағдайда немесе жолдың соңына жетсек цифрларландан тұрған жолды сандарға айналдырып тізімне қосу керек.

Осы бағдарламаның листингі келесі суретте бейнеленген:

|  |
| --- |
| s = input()  l = len(s)  integ = []  i = 0  while i < l:  s\_int = ''  a = s[i]  while '0' <= a <= '9':  s\_int += a  i += 1  if i < l:  a = s[i]  else:  break  i += 1  if s\_int != '':  integ.append(int(s\_int))    print(integ)  5 dkfj78df 9 8 dkfj8  Жауабы: [5, 78, 9, 8, 8] |

Сурет 17 Жолдардан санды таңдап алу бағдарламасының листингі

Нөмірі бойынша сөзді табу. Мәтіннен арнайы нөмірі бойынша сөзді табу (мысалы, реттік нөмірі бесінші сөзді табу). Экранға ол сөздің бірінші әрпін шығару керек.

Сөзді санау үшін төмендегі программада count айнымалысы қолданылады, ал f – біздің сол сөзде тұрымыз ба, әлде келесі жаң сөз басталып кетті ме соны анықтайды.

Егер жаңа сөз басталса, сөздер есептегіші арта түседі, ал егер есептегіш ізделінудегі сөздің нөміріне тең болса цикл аяқталады.

s = input()

l = len(s)

n = int(input("Номер слова: "))

count = 0

f = 0

i = 0

while i < l:

if s[i] != ' ' and f == 0:

count += 1

f = 1

if count == n:

break

elif s[i] == ' ':

f = 0

i += 1

print(s[i])

Кодтың орындалу мысалы:

Келесі мәндер енгізіледі: один два три четы пять шесть

Сөздің нөмірі: 4

Жауабы: ч

Алайда бұл есепті Python тілінің көмегімен шығару әлдеқайда ауқымды емес, яғни төмендегі кестеде көрсетілгендей жазып шығуға болады.

Кесте 1

Python тіліндегі есептің қысқаша жазылу кестесі

|  |
| --- |
| s = input()  l = len(s)  n = int(input("Номер слова: "))  s = s.split()  print(s[n][0]) |

Бұндағы split() әдісі аргументсіз болған жағдайда, жолды бос орындары бойынша сөздерге бөледі. s[n][0] өрнегі тізімнен n- элементті алып тастап, содан соң жолдан бірінші таңбаны алады.

**Дәріс №4**

**Дәріс тақырыбы:** Python тіліндегі сандар.

**Дәрістің мақсаты:** Python тіліндегі сандар түрлерін қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. Python тіліндегі сандар.

1. Python тіліндегі сандар.

Python тілінде сандар келесі түрлері бар: бүтін, нақты, комплексті. Бүтін сандар (int). Python тіліндегі сандардың қарапайым сандардан еш айырмашылығы жоқ. Олар қарапайым математикалық операцияларды қолдайды және 5-кестеде көрсетілген.

Кесте 2

Математикалық операциялар

|  |  |
| --- | --- |
| x + y | Қосу |
| x – y | Айырма |
| x \* y | Көбейту |
| x / y | Бөлу |
| x // y | Бөлгендегі бүтін бөлік |

2-кестенің жалғасы

|  |  |
| --- | --- |
| x % y | Бөлгендегі қалдық |
| -x | Санның белгісін ауыстыру |
| abs(x) | Сан модулі |
| divmod(x, y) | (x // y, x % y) жұп |
| x \*\* y | Дәрежеге шығару |
| pow(x, y[, z]) | xy модулі бойынша (егер модулі берілсе) |

Тағы бір айта кететіні, Python тіліндегі бүтін сандар, басқа бағдарламалау тілдеріне қарағанда ұзақ арифметиканы қолдайды (бірақ ол жадыдан көп орын талап етеді). Мысалы келесі 22-суретте бағдарламаның бөлігінің листингі көрсетілген.

**>>>** 255 + 34

289

**>>>** 5 \* 2

10

**>>>** 20 / 3

6.666666666666667

**>>>** 20 // 3

6

**>>>** 20 % 3

2

**>>>** 3 \*\* 4

81

**>>>** pow(3, 4)

81

**>>>** pow(3, 4, 27)

0

**>>>** 3 \*\* 150

369988485035126972924700782451696644186473100389722973815184405301748249

Сурет 18 Бүтін сандарға қолданылған операциялар

### Биттік операциялар. Бүтін сандармен де биттік операцияларды орындауға болады.

### Кесте 3

### Биттік операциялар тізімі

|  |  |
| --- | --- |
| x | y | Биттік немесе |
| x ^ y | Биттік «алып тастағыш» немесе |
| x & y | Биттік *и* |
| x << n | Солға биттік жылжу |
| x >> y | Оңға биттік жылжу |
| ~x | Биттер инверсиясы |

### Қосымша әдістер. int.bit\_length() - белгісі мен жетекші нөлдерін алып тастағандағы екілік жүйеде санды беру үшін қажетті биттер саны. Төменде көрсетілгендей:

>>>

>>> n = -37

>>> bin(n)

'-0b100101'

>>> n.bit\_length()

6

int.to\_bytes(length, byteorder, \*, signed=False) – осы санды беретін байттар жолын қайтарады. Ол келесі 23-суретте көрсетілді.

|  |
| --- |
| >>> (1024).to\_bytes(2, byteorder='big')  b'\x04\x00'  >>> (1024).to\_bytes(10, byteorder='big')  b'\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00'  >>> (-1024).to\_bytes(10, byteorder='big', signed=True)  b'\xff\xff\xff\xff\xff\xff\xff\xff\xfc\x00'  >>> x = 1000  >>> x.to\_bytes((x.bit\_length() // 8) + 1, byteorder='little')  b'\xe8\x03' |

Сурет 19 Int.to\_bytes әдісі

Келесі classmethod  int.from\_bytes(bytes, byteorder, \*, signed=False) әдісі аталған байттар жолынан сандарды қайтарады.

>>> int.from\_bytes(b'\x00\x10', byteorder='big')

16

>>> int.from\_bytes(b'\x00\x10', byteorder='little')

4096

>>> int.from\_bytes(b'\xfc\x00', byteorder='big', signed=True)

-1024

>>> int.from\_bytes(b'\xfc\x00', byteorder='big', signed=False)

64512

>>> int.from\_bytes([255, 0, 0], byteorder='big')

16711680

### Санау жүйелері. Сандар тек ғана ондық санау жүйесінде ғана емес сонымен қатар, басқа да санау жүйелерінде беріледі. Мысалы, компьютерде екілік санау жүйесі қолданылады. 19 саны екілік санау жүйесінде 10011 деп бейнеленеді. Кейбір жағдайда бір санау жүйесінен екіншіге көшу қажет болады. Оны орындауға Python тілі бірнеше функцияларды ұсынады:

* int([object], [санау жүйесінің негізі]) – ондық санау жүйесіндегі бүтінге айналдыру. Бұл жерде үнсіз келісім бойынша ондық санау жүйесі қолданылады, бірақ негізін 2-ден 36-ға дейін таңдай отырып кез келген санау жүйесінде беруге болады.
* bin(x) – бүтін санды екілік жолға айналдыру.
* hex(х) - бүтін санды он алтылық жолға айналдыру.
* oct(х) - бүтін санды сегіздік жолға айналдыру.

Келесі 20-суретте мысалдар көрсетілген.

|  |
| --- |
| >>> a = int('19') # жоды санаға айналдыру  >>> b = int('19.5') # жол бүтін сан болып табылмайды  Traceback (most recent call last):  File "", line 1, in  ValueError: invalid literal for int() with base 10: '19.5'  >>> c = int(19.5) # жылжымалы нүктесі бар санға қолданылғанда бөлшек бөлігін алып тастайды  >>> print(a, c)  19 19  >>> bin(19)  '0b10011'  >>> oct(19)  '0o23'  >>> hex(19)  '0x13'  >>> 0b10011 # сандық тұрақтылықарды ослай да жазуға болады.  19  >>> int('10011', 2)  19  >>> int('0b10011', 2)  19 |

Сурет 20 Санау жүйесіне арналған мысалдар листингі

## Нақты сандар (float). Нақты сандар да бүтін сандардағы сиқты операцияларды қолдайды, бірақ сандарды компьютерде бергендіктен нақты сандар нақты болмауы және қателіктерге соқтыруы мүмкін және оның өзі қателіктерге әкелуі мүмкін.

>>> 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1

0.9999999999999999

Жоғары нақтылыққа қол жеткізу үшін басқа да объектілерді (Мысалы: Decimal және [Fraction](https://pythonworld.ru/moduli/modul-fractions.html)) қолдануға болады.

Сонымен қатар, нақты типтер ұзақ арифметиканы қолдамайды.

>>>

>>> a = 3 \*\* 1000

>>> a + 0.1

Traceback (most recent call last):

File "", line 1, in

OverflowError: int too large to convert to float

Сандармен қарапайым мысалдар:

>>>

>>> c = 150

>>> d = 12.9

>>> c + d

162.9

>>> p = abs(d - c) # сан модулі

>>> print(p)

137.1

>>> round(p) # Дөңгелектеу

137

### Қосымша әдістері:

* float.as\_integer\_ratio() - қатынасы осы санға тең болатын жұп сандар.
* float.is\_integer() - мән бүтін сан бола ма.
* float.hex() – float-ты hex-ке түрлендіреді (он алтылық жүйеде).
* classmethod float.fromhex(s) - он алтылық жолдағы float.

**>>>** (10.5).hex()

'0x1.5000000000000p+3'

**>>>** float.fromhex('0x1.5000000000000p+3')

10.5

Python тілінде сандармен жұмысқа қатысты стандартты өрнектерден басқа бірнеше қажетті модульдер бар. Келесі [math](https://pythonworld.ru/moduli/modul-math.html) модулі күрделі математикалық функцияларды қамтамасыз етеді. Оның қолданылуы төмендегідей.

>>> import math

>>> math.pi

3.141592653589793

>>> math.sqrt(85)

9.219544457292887

Келесі random [модулі](https://pythonworld.ru/moduli/modul-random.html)  - кездейсоқ сандар генераторы мен кездейсоқ функцияны таңдауды жүзеге асырады.

>>> import random

>>> random.random()

0.15651968855132303

## Комплексті сандар (complex). Python тіліне сонымен қатар, комплекті сандар енгізілген.

## 21-суретте комплекс сандардың программада қолданылуы көрсетілген.

**>>>** x = complex(1, 2)

**>>> print**(x)

(1+2j)

**>>>** y = complex(3, 4)

**>>> print**(y)

(3+4j)

**>>>** z = x + y

**>>> print**(x)

(1+2j)

**>>> print**(z)

(4+6j)

**>>>** z = x \* y

**>>> print**(z)

(-5+10j)

**>>>** z = x / y

**>>> print**(z)

(0.44+0.08j)

**>>> print**(x.conjugate()) *# түйіндес сан*

(1-2j)

**>>> print**(x.imag) *# жорамал бөлігі*

2.0

**>>> print**(x.real) *# нақты бөлігі*

1.0

**>>> print**(x > y) *# Комплексті санды салыстыруға болмайды*

Traceback (most recent call last):

File "", line 1, in

TypeError: unorderable types: complex() > complex()

**>>> print**(x == y) *# бірақ теңдікке тексеруге болады*

False

**>>>** abs(3 + 4j) *# комплексті сан модулі*

5.0

**>>>** pow(3 + 4j, 2) *# санның дәрежесін шығару*

(-7+24j)

Сурет 21 Комплексті сандар

Комплексті сандармен жұмыс жасау үшін сонымен қатар,  [cmath](https://pythonworld.ru/moduli/modul-cmath.html) модулі де қолданылады.

Тізімдер (list). Тізімдердің функциялары мен әдістері. Тізімдер деген типке, олармен орындалатын операциялар және әдістерге, тізімдер генераторы және тізімдерді қолдануға тоқталайық.

# Тізім дегеніміз не? Python-дағы тізім – ерікті түрдегі нысандардың (объектілердің) реттелген өзгертілетін топтамасы (массив сияқты, бірақ типтері әр басқа болуы мүмкін). Тізімдерді қолдану үшін оларды құрып алу керек. Тізімді бірнеше жолмен құруға болады. Мысалы, кез келген қайталанатын нысанды өңдеуге болады (мысалы, [жолды](https://pythonworld.ru/tipy-dannyx-v-python/stroki-literaly-strok.html) list ендірілген функциясымен):

**>>>** list('тізім')

['с', 'п', 'и', 'с', 'о', 'к']

Тізімді литералдардың да көмегімен құруға болады:

**>>>** s = [] # Бос список

**>>>** l = ['s', 'p', ['isok'], 2]

**>>>** s

[]

**>>>** l

['s', 'p', ['isok'], 2]

Мысалда көрсетілгендей, тізім кез-келген нысанның (кірістірілген тізімдерді қоса алғанда) кез келген санын қамтуы немесе ештеңені қамтымауы да мүмкін.

Тізімді құрудың тағы бір жолы - тізімдер генераторы. тізімдер генераторы – тізбектегі элементтің әрқайсысына өрнекті қолдана отырып жаңа тізімді құру тәсілі. Тізімдер генераторы [for](https://pythonworld.ru/osnovy/cikly-for-i-while-operatory-break-i-continue-volshebnoe-slovo-else.html) цикліне өте ұқсайды.

>>> c = [c \* 3 for c in 'list']

>>> c

['lll', 'iii', 'sss', 'ttt']

Тізімдер генераторының бұдан да күрделі конструкциясы болуы мүмкін.

>>> c = [c \* 3 for c in 'list' if c != 'i']

>>> c

['lll', 'sss', 'ttt']

>>> c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a']

>>> c

['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']

Бірақ күрделі жағдайларда тізімдерді жасау үшін әдеттегі for циклы қолданған жөн.

Тізімдердің функциялары мен әдістері. Тізімді құрып алған соң олармен жұмыс істеуге болады. Тізімдер үшін негізгі ендірілген функциялар мен әдістер бар.

Кесте 4

Тізімдер әдістері

| Әдіс | Атқаратын қызметі |
| --- | --- |
| list.append(x) | Тізімнің соңына элемент қосады |
| list.extend(L) | L тізімінің барлық элементтерін соңына қосып list тізімін кеңейтеді |
| list.insert(i, x) | i-нші элементке x мәнін қояды |
| list.remove(x) | Х мәні бар тізімдегі бірінші элементті жояды. Егер ондай элемент жоқ болса ValueError деп шығады |
| list.pop([i]) | i-ый элементті жояды және оны қайтарады. Егер индекс көрсетілмесе, соңғы элементті жояды |
| list.index(x, [start [, end]]) | x мәні бар бірінші элементтің жағдайын қайтарады (бұнда іздеу start-тан end-қа дейін жүргізіледі) |
| list.count(x) | X мәні бар элементтердің санын қайтарады. |
| list.sort([key=функция]) | Функциялар негізінде тізімді сұрыптайды |
| list.reverse() | Тізімді кеңейтеді (ашып жазады) |
| list.copy() | Тізімнің үстіңгі көшірмесі |
| list.clear() | Тізімді тазартады |

Айта кету керек, тізімдер әдісі жолдың әдісіне қарағанда, тізімнің өзін өзгертеді, сондықтан орындаудың нәтижесін осы айнымалыға жазбауға болады.

>>> l = [1, 2, 3, 5, 7]

>>> l.sort()

>>> l

[1, 2, 3, 5, 7]

>>> l = l.sort()

>>> print(l)

None

Тізімдермен жұмыстардың мысалдары::

>>> a = [66.25, 333, 333, 1, 1234.5]

>>> print(a.count(333), a.count(66.25), a.count('x'))

2 1 0

>>> a.insert(2, -1)

>>> a.append(333)

>>> a

[66.25, 333, -1, 333, 1, 1234.5, 333]

>>> a.index(333)

1

**>>>** a.remove(333)

**>>>** a

[66.25, -1, 333, 1, 1234.5, 333]

**>>>** a.reverse()

**>>>** a

[333, 1234.5, 1, 333, -1, 66.25]

**>>>** a.sort()

**>>>** a

[-1, 1, 66.25, 333, 333, 1234.5]

Кейбір кезде өнімділікті арттыру үшін тізімдерді әлдеқайда икемді массивтермен алмастырады (бұндай жағдайда әдетте ендірілген кітапханаларды қолданады. Мысалы [NumPy](https://pythonworld.ru/numpy)).

# Кортеждер (tuple). Келесі деректер типі кортеждер (tuple) деп аталады. Олар негізінен өзгертілмейтін тізімдер болып табылады.

## Олар не үшін қажет?

* Қорғау үшін. Кортеж әдейі және кездейсоқ өзгертулерден қорғалған.
* Өлшемі аз. Негізі болмас үшін:

**>>>** a = (1, 2, 3, 4, 5, 6)

**>>>** b = [1, 2, 3, 4, 5, 6]

**>>>** a.\_\_sizeof\_\_()

36

**>>>** b.\_\_sizeof\_\_()

44

Кортеждерді сөздіктердің кілттері ретінде қолдану мүмкіндігі:

>>> d = {(1, 1, 1) : 1}

>>> d

{(1, 1, 1): 1}

>>> d = {[1, 1, 1] : 1}

Traceback (most recent call last):

File "", line 1, in

d = {[1, 1, 1] : 1}

TypeError: unhashable type: 'list'

## Кортеждармен қалай жұмыс істеу керек? Кортеждермен жұмыс тізімдермен сияқты жүзеге асырылады. Бос кортеж құру:

>>> a = tuple() # tuple() ендірілген функциясы арқылы

>>> a

()

>>> a = () # кортеждің литералы арқылы

>>> a

()

>>>

Бір элементтен кортеж құрамыз:

**>>>** a = ('s')

**>>>** a

's'

Бұл жерде жол пайда болды. Енді кортежді алу үшін келесі өзгерісті енгіземіз:

**>>>** a = ('s', )

**>>>** a

('s',)

Байқағандай барлық мәселе үтірде болды. Өз-өзімен жақшалар ештеңені бермейді. Кортежді келесі түрде де құруға болады.

**>>>** a = 's',

**>>>** a

('s',)

Әдетте жақша қою міндетті болады.

Қайталанатын объектісі бар кортежді де tuple() функциясы көмегімен қуруға болады.

**>>>** a = tuple('hello, world!')

**>>>** a

('h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd', '!')

## Кортеждермен операциялар*.* Тізімдермен орындалатын барлық операциялар, өзгертілмейтін тізімдер(қосу, санға көбейту, index() және count() әдістері және бірнеше басқа да операциялар) орындалады.

## Сонымен бірге, элементтердің орнын әр түрлі түрде ауыстыруға және т.с.с. атқаруға болады.

Мысалы: Python-да екі айнымалының мәнінің орнын ауыстыру :

a, b = b, a

**Дәріс №5**

**Дәріс тақырыбы:** Функциялар және файлдармен жұмыс.

**Дәрістің мақсаты:** Python бағдарламалау тілінде функциялар және файлдармен жұмыс жасауды қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. Функциялар және файлдармен жұмыс.

# Функциялар және файлдармен жұмыс.

# Бұнда атаулы мен анонимді функциялар және def, return және lambda интрукциялары қарастырылады. Сонымен қатар, функцияның міндетті және міндетті емес аргументтері, еркін санды аргументтері бар функциялар қарастырылады.

Атаулы функциялар, def инструкциясы. Python тіліндегі функция аргументтер қабылдайтын және мәнді қайтаратын нысан болып табылады. Әдетте функция def инструкциясы көмегімен анықталады.

Қарапайым функцияны анықтайық:

def add(x, y):

return x + y

return Инструкция мәнді қайтаруды талап етеді. Ал аталған жағдайда функция x пен y қосындысын қайтарады. Енді оны шақыруға болады

>>> add(1, 10)

11

>>> add('abc', 'def')

'abcdef'

Функция кез-келген күрделілікте болуы және барлық нысандарды (тізімдер, кортеждерді және тіпті функцияларды) қайтаруы мүмкін:

>>> def newfunc(n):

... def myfunc(x):

... return x + n

... return myfunc

...

>>> new = newfunc(100) *# new - бұл функция*

>>> new(200)

300

Функция return инструкциясымен аяқталмауы да мүмкін, бірақ соның өзінде функция [None](https://pythonworld.ru/tipy-dannyx-v-python/none.html) мәнін қайтарады:

>>> def func():

... pass

...

>>> print(func())

None

## Функция аргументтері. Функция кез келген сандағы аргументтерді қабылдауы мүмкін немесе мүлдем қабылдамауы да мүмкін. Сондай-ақ, аргументтер саны кез келген, позициялық және атаулы аргументтері бар функциялар, міндетті және міндетті емес функциялар кең таралған.

>>> def func(a, b, c=2): # c – мінддетті емес аргумент

... return a + b + c

...

>>> func(1, 2) # a = 1, b = 2, c = 2 (үнсіз жағдайда)

5

>>> func(1, 2, 3) # a = 1, b = 2, c = 3

6

>>> func(a=1, b=3) # a = 1, b = 3, c = 2

6

>>> func(a=3, c=6) # a = 3, c = 6, b анықталмаған

Traceback (most recent call last):

File "", line 1, in

func(a=3, c=6)

TypeError: func() takes at least 2 arguments (2 given)

Функция позициялық аргументтердің айнымалы санын қабылдауы мүмкін, ондай жағдайда оның атауының алдына \* қойылады:

>>> def func(\*args):

... return args

...

>>> func(1, 2, 3, 'abc')

(1, 2, 3, 'abc')

>>> func()

()

>>> func(1)

(1,)

Мысалдан көргендей, функцияның барлық берілген аргументтерінің ішінен args – бұл  [кортеж](https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html), сонымен қатар айнымалының өзімен де кортежбен сияқты жұмыс істеуге болады.

Функция атаулы аргументтердің кез келген санын қабылдай алады, онда атудың алдына \*\* белгісі қойылады:

>>> def func(\*\*kwargs):

... return kwargs

...

>>> func(a=1, b=2, c=3)

{'a': 1, 'c': 3, 'b': 2}

>>> func()

{}

>>> func(a='python')

{'a': 'python'}

Kwargs айнымалысында барлық әрекеттерді атқаруға болатын сөздік сақталады.

## Анонимді функциялар, lambda инструкциясы. Анонимді функциялар тек бір ғана өрнекті қамтығандықтан тез орындалады. Анонимді функциялар lambda инструкциясының көмегімен құрылады. Сонымен қатар, оларды def func() инструкциясындағы сияқты міндетті түрде айнымалыларға меншіктеудің қажеті жоқ:

>>> func = lambda x, y: x + y

>>> func(1, 2)

3

>>> func('a', 'b')

'ab'

>>> (lambda x, y: x + y)(1, 2)

3

>>> (lambda x, y: x + y)('a', 'b')

'ab'

lambda функциясы басқа функцияларға қарағанда return инструкциясын қажет етпейді, ал басқа жағынан еш айырмашылығы жоқ:

>>>

>>> func = lambda \*args: args

>>> func(1, 2, 3, 4)

(1, 2, 3, 4)

# Файлдар. Файлдармен жұмыс. Төмендеpython тілінің файлдармен әрекет жасауға арналған ендірілген құралдары қарастырылады: ашу / жабу, оқу және жазу.

Файлмен жұмыс жасау үшін алдымен оны ашып алу керек. Ол үшін open ендірілген функциясы қолданылады:

f = open('text.txt', 'r')

Open функциясының көптеген өлшемдері бар, бірақ олардың ішінде ең қажетті 3 аргументі бар. Олардың біріншісі файлдың аты, файлға баратын жолдың өзі салыстырмалы немесе абсолютті болуы мүмкін. Екінші аргумент –ол файлды ашу режімі.

Кесте 4

Файлды ашу режімдері

|  |  |
| --- | --- |
| Режім | Белгіленуі |
| 'r' | Оқу үшін ашу (әдепкі мән болып табылады) |
| 'w' | Жазу үшін ашу, файлдың ішіндегі мәні жойылады, файл жоқ болса, онда жаңадан құрылады |
| 'x' | Жазу үшін ашу, егер файл жоқ болса, онда ерекше жағдай деп саналады |
| 'a' | қосымша жазу үшін ашу, ақпарат файлдың соңына қосылады |
| 'b' | Екілік режимде ашу |
| 't' | мәтіндік режимде ашу (әдепкі мән болып табылады) |
| '+' | оқу және жазу үшін ашу |

Режимдер біріктірілген болуы мүмкін, яғни, мысалы, 'rb' – екілік режімде оқу. Әдепкі жағдайда мәні 'rt' болады.

Соңғы аргумент encoding, ол тек мәтіндік режімде файлды оқу кезінде қажет. Бұл аргумент кодтауды анықтайды.

## Файлдан оқу. Файлды ашқан соң одан ақпаратты оқуымыз керек. Ол үшін оның бірнеше тәсілі бар, бірақ соның ішінде екеуі ғана қажет болады.

Бірінші – егер аргументсіз болса файлды тұтастай, егер n (бүтін санмен) аргументімен шақырылса, онда n символымен оқылатын read әдісі.

**>>>** f = open('text.txt')

**>>>** f.read(1)

'H'

**>>>** f.read()

'ello world!\nThe end.\n\n'

Тағы бір тәсілі – файлды - for циклін қолдану арқылы файлды жол жолымен оқу:

>>> f = open('text.txt')

>>> for line in f:

... line

...

'Hello world!\n'

'\n'

'The end.\n'

'\n'

Файлға жазу*.* Файлға келесі тізімді жазуды орындап көрейік:

>>> l = [str(i)+str(i-1) for i in range(20)]

>>> l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

Файлды жазу үшін ашу:

>>> f = open('text.txt', 'w')

Файлға жазу write әдісі арқылы жүзеге асырылады. Ол келесі 26-суретте көрсетілген:

|  |
| --- |
| >>> for index in l:  ... f.write(index + '\n')  ...  4  3  3  3  3 |

Сурет 22 Write әдісі арқылы файлға жазу

Жоғарыда көрсетілген жазбалардан: write әдісі жазылған символдардың санын қайтарады [17].

Файлмен жұмыс жасап болған соң оны міндетті түрде close әдісі арқылы жабу керек:

>>> f.close()

Енді осы тізімді құрылған файлдан қайта жасауға тырысайық. Ол үшін файлды оқу үшін ашып жолдарын оқу керек.

>>> f = open('text.txt', 'r')

>>> l = [line.strip() for line in f]

>>> l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

>>> f.close()

Жоғарыда көрсетілгендей қайта сол тізім алынды. Күрделірек жағдайда жазу алгоритмін ойлап табу қиынға соғады. Сондықтан Python тілінде ондай жағдайлар үшін баяғыдан [pickle](https://pythonworld.ru/moduli/modul-pickle.html) немесе [json](https://pythonworld.ru/moduli/modul-json.html) сияқты құралдары ойлап табылған. Олар файлда анағұрлым күрделі құрылымдарды сақтауға мүмкіндік береді [17].

**Дәріс №6**

**Дәріс тақырыбы:** Сандық алгоритмдер. Матрицалық есептеулер.

**Дәрістің мақсаты:** Сандық алгоритмдер мен матрицалық есептеулерді қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. Numeric Python.
2. Numeric Модулі.
3. Массивті құру.

1. Numeric Python.

Бұл бөлім жеткілікті дәрежеде ескірген. Қазір модуль numpy деп аталады. Жалпы, Numeric-те болған барлық numpy-да қол жетімді, бірақ аттар сәйкес келмеуі мүмкін. Numeric NumPy/oldnumeric сияқты қол жетімді. Сіздің құжаттама.

Numeric Python - көптеген сандық қосымшалар үшін қажетті көп өлшемді массивтерді есептеу үшін бірнеше модуль. Numeric модулі MatLab, Octave (MATLAB аналогы), APL, J, S+, ID сияқты пакеттер мен жүйелердің Python мүмкіндіктерін енгізеді. Пайдаланушылар Numeric-ті оңай және ыңғайлы табады. Кейбір Python синтаксистік мүмкіндіктері (кесіктерді пайдаланумен байланысты) Numeric үшін арнайы әзірленген.

Numeric Python үшін құралы бар:

* матрицалық есептеулер;
* FFT Фурье жылдам түрлендіру;
* жетіспейтін ma эксперименталдық деректермен жұмыс істеу;
* RNG статистикалық модельдеу;
* MatLab бағдарламасының негізгі функцияларының эмуляциясы.

1. Numeric Модулі.

Numeric модулі толымды түрді анықтайды-массив және массив операцияларына арналған функциялардың көп санын қамтиды. Массив-бұл индекстер бойынша қолжетімді біртекті элементтер жиынтығы. Numeric Модулінің массивтері көп өлшемді, яғни бір мөлшерден артық болуы мүмкін.

1. Массивті құру.

Массивті жасау үшін, массивтің мазмұнын (ішкі тізімдер түрінде) және түрін көрсете отырып, array () функциясын пайдалануға болады. Array() функциясы дәлел - массив болса, көшірмесін жасайды. As array() функциясы ұқсас жұмыс істейді, бірақ оның аргументі массиві болған кезде жаңа массив жасамайды:

>>> from Numeric import \*

>>> print array([[1, 2], [3, 4], [5, 6]])

[[1 2]

[3 4]

[5 6]]

>>> print array([[1, 2, 3], [4, 5, 6]], float)

[[ 1. 2. 3.]

[ 4. 5. 6.]]

>>> print array([78, 85, 77, 69, 82, 73, 67], 'c')

[N U M E R I C]

Массив элементтері ретінде келесі түрлерді пайдалануға болады: Int8-Int32, Unsigned Int 8-Unsigned Int32, Float 8-Float64, Complex8-Complex64 және PyObject. 8, 16, 32 және 64 сандар шаманы сақтауға арналған биттердің санын көрсетеді. Int, Unsigned Integer, Float және Complex түрлері осы платформада қабылданған мәндерге сәйкес келеді . Массивте сондай-ақ еркін нысандарға сілтемелерді сақтауға болады .

Әрбір ось бойынша массив мөлшері мен ұзындығы массив формасы (shape) деп аталады . Массив пішініне қол жеткізу shape атрибуты арқылы жүзеге асырылады:

>>> from Numeric import \*

>>> a = array(range(15), int)

>>> print a.shape

(15,)

>>> print a

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

>>> a.shape = (3, 5)

>>> print a.shape

(3, 5)

>>> print a

[[ 0 1 2 3 4]

[ 5 6 7 8 9]

[10 11 12 13 14]]

**Дәріс №9**

**Дәріс тақырыбы:** Web-қосымшаларды әзірлеу.

**Дәрістің мақсаты:** Web-қосымшаларды әзірлеуді қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. CGI сценарийлері.

2. CGI модулі.

Www-браузерде HTML және XML-құжаттарды басқаратын негізгі интерфейсі жұмыс істейтін бағдарлама web-қосымша деп түсініледі. Сапасын жақсарту үшін пайдаланушы интерфейсінің жиі қолданады JavaScript, алайда бұл бірнеше төмендетеді әмбебаптығы интерфейс. Айта кету керек, интерфейс құруға болады Java - немесе Flash-апплетах, алайда, мұндай қосымшалар қиын деп атауға web-қосымшаларда, себебі Java немесе Flash пайдалана алады өз хаттамалар қарым-қатынас жасау үшін, сервер, ал стандартты емес үшін WWW хаттама HTTP.

Web-қосымшаларды құру кезінде пішін (сыртқы түрі, стилі), мазмұны мен деректерді өңдеу логикасын бөлуге тырысады. Web-сайттарды құрудың заманауи технологиялары осы идеалға өте жақын болуға мүмкіндік береді. Дегенмен, көп деңгейлі қосымшаларды қолданбай да, басқаларды қозғамай (немесе қозғамай), осы аспектілердің кез келгенін өзгертуге мүмкіндік беретін стильді ұстануға болады. Бұл тақырыпты талқылау әзірлеу ортасына арналған бөлімде жалғасатын болады.

1. CGI сценарийлері.

Www үшін қосымшаларды жасаудың классикалық жолы-CGI сценарийлерін жазу (кейде скрипттер айтады). CGI ( Common Gateway Interface, жалпы шлюздік интерфейс) - бұл сервердің сыртқы қосымшалармен өзара әрекеттесуін регламенттейтін стандарт. WWW болған жағдайда, web-сервер белгілі бір сценарий бойынша бетті генерациялау сұрауын жібере алады. Бұл сценарий web-серверден (сол, өз кезегінде, оларды пайдаланушыдан ала алады) кіріс деректерді алып, дайын нысанды (сурет, аудиодер, стильдер кестесі және т.б.) генерациялайды.

Сценарийді шақырған кезде Web-сервер оған ақпаратты стандартты енгізу, орта айнымалылары арқылы және ISINDEX үшін командалық жолдың аргументтері арқылы (олар sys арқылы қол жетімді.argv ).

Web-серверге (және CGI-сценарийге) - GET және POST браузерде толтырылған формадан деректерді берудің екі негізгі әдісі. Әдісіне байланысты деректер әртүрлі беріледі. Бірінші жағдайда олар кодталады және тікелей URL-ге орналастырылады, мысалы: http://example.com/cgi-bin/a.cgi?a=1&b=3. Сценарий оларды QUERY\_STRING атымен айнымалыға алады. POST әдісі жағдайында олар стандартты енгізуге беріледі.

Ол үшін, егер бұл сервердің конфигурациясы рұқсат етілсе, HTML құжаттары арасында кез-келген орында. Сценарий орындалуының белгісі болуы тиіс. Unix жүйесінде оны chmod a+x пәрменімен орнатуға болады.

Келесі қарапайым сценарий os сөздігінен мәндерді шығарады.environ және оған не берілгенін көруге мүмкіндік береді:

#!/usr/bin/python

import os

print """Content-Type: text/plain

%s""" % os.environ

Оның көмегімен Web-сервермен орнатылған орта айнымалыларын көруге болады. Web-сервердің CGI сценарийімен берілетін файл метаақпараттық өрістер (мазмұн түрі, құжаттың соңғы жаңартылу уақыты, кодтау және т.б .) көрсетілген тақырып бөлігін қамтиды.

Сценарийлерді жасау үшін жеткілікті негізгі орта айнымалылары:

QUERY\_STRING

Сұраныс жолы.

REMOTE\_ADDR

Клиенттің IP-мекенжайы.

REMOTE\_USER

Клиенттің аты (егер ол сәйкестендірілсе).

SCRIPT\_NAME

Сценарий атауы.

SCRIPT\_FILENAME

Сценарийі бар Файл атауы.

SERVER\_NAME

Сервер аты.

HTTP\_USER\_AGENT

Клиент браузерінің атауы.

REQUEST\_URI

Сұрау жолы (URI).

HTTP\_ACCEPT\_LANGUAGE

Құжаттың қажетті тілі.

Мұнда os сөздігі болуы мүмкін.environ CGI сценарийінде:

{

'DOCUMENT\_ROOT': '/var/www/html',

'SERVER\_ADDR': '127.0.0.1',

'SERVER\_PORT': '80',

'GATEWAY\_INTERFACE': 'CGI/1.1',

'HTTP\_ACCEPT\_LANGUAGE': 'en-us, en;q=0.50',

'REMOTE\_ADDR': '127.0.0.1',

'SERVER\_NAME': 'rnd.onego.ru',

'HTTP\_CONNECTION': 'close',

'HTTP\_USER\_AGENT': 'Mozilla/5.0 (X11; U; Linux i586; en-US;

rv:1.0.1) Gecko/20021003',

'HTTP\_ACCEPT\_CHARSET': 'ISO-8859-1, utf-8;q=0.66, \*;q=0.66',

'HTTP\_ACCEPT': 'text/xml,application/xml,application/xhtml+xml,

text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,

image/gif;q=0.2,text/css,\*/\*;q=0.1',

'REQUEST\_URI': '/cgi-bin/test.py?a=1',

'PATH': '/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin',

'QUERY\_STRING': 'a=1&b=2',

'SCRIPT\_FILENAME': '/var/www/cgi-bin/test.py',

'HTTP\_KEEP\_ALIVE': '300',

'HTTP\_HOST': 'localhost',

'REQUEST\_METHOD': 'GET',

'SERVER\_SIGNATURE': 'Apache/1.3.23 Server at rnd.onego.ru Port 80',

'SCRIPT\_NAME': '/cgi-bin/test.py',

'SERVER\_ADMIN': 'root@localhost',

'SERVER\_SOFTWARE': 'Apache/1.3.23 (Unix) (Red-Hat/Linux)

mod\_python/2.7.8 Python/1.5.2 PHP/4.1.2',

'SERVER\_PROTOCOL': 'HTTP/1.0',

'REMOTE\_PORT': '39251'

}

Келесі CGI-сценарий қара квадрат береді (онда суреттерді өңдеу үшін Image модулі қолданылады):

#!/usr/bin/python

import sys

print """Content-Type: image/jpeg

"""

import Image

i = Image.new("RGB", (10,10))

i.im.draw\_rectangle((0,0,10,10), 1)

i.save(sys.stdout, "jpeg")

2. CGI модулі.

Python CGI модулі түрінде CGI қолдауы бар. Келесі мысал оның кейбір мүмкіндіктерін көрсетеді:

#!/usr/bin/python

# -\*- coding: cp1251 -\*-

import cgi, os

# анализ запроса

f = cgi.FieldStorage()

if f.has\_key("a"):

a = f["a"].value

else:

a = "0"

# обработка запроса

b = str(int(a)+1)

mytext = open(os.environ["SCRIPT\_FILENAME"]).read()

mytext\_html = cgi.escape(mytext)

# формирование ответа

print """Content-Type: text/html

<html><head><title>Решение примера: %(b)s = %(a)s + 1</title></head>

<body>

%(b)s

<table width="80%%"><tr><td>

<form action="me.cgi" method="GET">

<input type="text" name="a" value="0" size="6">

<input type="submit" name="b" value="Обработать">

</form></td></tr></table>

<pre>

%(mytext\_html)s

</pre>

</body></html>""" % vars()

Бұл мысал түрінде берілген санға 1 қосылады. Сонымен қатар, сценарийдің бастапқы коды шығады. Таңбаларды экрандау>,<, & cgi функциясы пайдаланылғанын байқағаныңыз жөн. escape(). Web-бетті қалыптастыру үшін пішімдеу операциясы қолданылды. Сөздік ретінде барлық жергілікті айнымалылармен vars () сөздігі қолданылады. Пайыз белгілері пішімдеу командасымен түсіндірілмеуі үшін екі еселенді. Пайдаланушыдан мәні қалай алынғанына назар аудару керек. FieldStorage нысаны "дерлік" сөздік, әдеттегі мәнді алу үшін қосымша мән төлсипатын қарау керек. Сценарийге тек мәтіндік мәндерді ғана емес, сондай-ақ файлдарды да, сондай-ақ сол атауымен көпше мәндерді де бере алады.

Абайлаңыз!

CGI сценарийлердің кіріс мәндерін өңдегенде, рұқсат етілген мәндерді мұқият және мұқият тексеру қажет. Клиент кез келген нәрсені кіре алады деп санаған дұрыс. Осының барлығы сценарийді күтетін нәрсені таңдау және тексеру қажет.

Мысалы, пайдаланушыдан алынған деректерді eval() және оған ұқсас функцияларға аргументтер ретінде файл жолына; командалық жолдың параметрлері; деректер базасына SQL-сұранымындағы бөліктер ретінде қоюға болмайды. Сондай-ақ, егер бұл беттер тек URL мекен - жайына тапсырыс берген клиент ғана емес (мысалы, мұндай жағдай web-чаттарда, форумдарда, қонақ кітаптарында әдеттегі), және тіпті осы ақпараттың жалғыз оқырманы-сайт әкімшісі болған жағдайда да, алынған деректерді тікелей қалыптастырылатын беттерге енгізуге болмайды. HTML-коды тексерілмеген, пайдаланушыдан тікелей келіп түскен беттерді қарайтын адам өзінің браузерінде оны қорғауда пайдаланатын зиянды кодты өңдеу қаупі бар.

CGI сценарийін URL-ге сұрау арқылы басқа сценарийлер ғана пайдаланса да, пайдаланушы деректерді енгізгеннен кейін кіріс мәндерін мұқият тексеру қажет. (Өйткені теріс пайдаланушы web-серверге кез келген мәндерді бере алады).

Жоғарыда мысалда int () функциясын шақырған кезде рұқсат етілген тексеру жүргізілді: егер сансыз мән қойылса, сценарий авариялық аяқталды, ал Пайдаланушы Internal Server Error көрді.

Кіріс деректерін талдағаннан кейін оларды өңдеу фазасын бөліп көрсетуге болады. Бұл бөлімде CGI-сценарий одан әрі шығару үшін айнымалылар есептеледі. Бұл жерде берілген айнымалылардың мәндерін ғана емес, сонымен қатар олардың болу немесе болмау фактілерін де ескеру қажет, себебі бұл да сценарий логикасына әсер етуі мүмкін.

Және, соңында, дайын нысанды (мәтін, HTML-құжат, сурет, мультимедиа-объект және т.б.) шығару фазасы. Беттің үлгісін (немесе оның ірі бөліктерін) алдын ала дайындау оңай, содан кейін жай ғана айнымалылардан мазмұнды толтыру.

Келтірілген мысалдарда аттар сұраныс жолында тек бір рет пайда болды. Кейбір нысандар бір атау үшін бірнеше мәндерді тудырады. Барлық мәндерді getlist әдісі арқылы алуға болады():

lst = form.getlist("fld")

Lst тізімі fld аты бар өрістер web-формадан алынған ( берілген аты бар бірде- бір өріс толтырылмаған болса, ол бос болуы мүмкін) сияқты мәндерден тұрады .

Кейбір жағдайларда серверге файлдарды жіберу керек (upload жасау). Келесі мысал және оған түсініктеме осы міндетті шешуге көмектеседі:

#!/usr/bin/env python

import cgi

form = cgi.FieldStorage()

file\_contents = ""

if form.has\_key("filename"):

fileitem = form["filename"]

if fileitem.file:

file\_contents = """<P>Содержимое переданного файла:

<PRE>%s</PRE>""" % fileitem.file.read()

print """Content-Type: text/html

<HTML><HEAD><TITLE>Загрузка файла</TITLE></HEAD>

<BODY><H1>Загрузка файла</H1>

<P><FORM ENCTYPE="multipart/form-data"

ACTION="getfile.cgi" METHOD="POST">

<br>Файл: <INPUT TYPE="file" NAME="filename">

<br><INPUT TYPE="submit" NAME="button" VALUE="Передать файл">

</FORM>

%s

</BODY></HTML>""" % file\_contents

Ең басында сценарий соңында келтірілген web-форманы қарастыру керек: ол CGI-сценарий бойынша жүгінген кезде пайдаланушыға шығарылады. Сонымен қатар, web-браузерде енгізу жолағы мен "Browse"батырмасы бар file түрінің өрісі бар. "Browse" батырмасын басу арқылы пайдаланушы компьютерде ОЖ қол жетімді файлды таңдайды. Осыдан кейін ол файлды серверге жіберу үшін "файлды жіберу" батырмасын басуы мүмкін.

CGI сценарийін баптау үшін cgitb модулін пайдалануға болады. Қате пайда болған кезде бұл модуль ерекшелеуді қоздыру орнын көрсете отырып, бояулы HTML-бетті береді. Жөнделген сценарийдің басында қою керек

import cgitb

cgitb.enable(1)

Немесе браузерде қателерді көрсету қажет болмаса:

import cgitb

cgitb.enable(0, logdir="/tmp")

Сценарий кейінге қалдырылған кезде бұл жолдарды алып тастау керек, өйткені ол сценарий кодының бөлігін береді. Бұл зиянкестер CGI-сценарийде осалдықты табу немесе парольдерді қарау үшін пайдалануы мүмкін (егер ондай сценарийде бар болса).

**Дәріс №10**

**Дәріс тақырыбы:** Python-ғы желілік қосымшалар.

**Дәрістің мақсаты:** Python-ғы желілік қосымшаларды қарастыру, талдау.

Тақырып бойынша қарастырылатын сұрақтар:

1. Сокеттермен жұмыс.

2. Smtplib модулі.

1. Сокеттермен жұмыс.

IP-желілерде қолданылатын архитектура клиент-сервер клиент пен сервер арасындағы байланыс үшін IP-пакеттерді пайдаланады. Клиент сұрауды сол жауап беретін серверге жібереді. TCP / IP болған жағдайда клиент пен сервер арасында байланыс орнатылады (әдетте деректерді екі жақты берумен), ал UDP/IP - клиент пен сервер кепілсіз жеткізумен пакеттермен (дейтаграммалармен) алмасады.

IP-желінің әрбір желілік интерфейсі осы желіде бірегей мекенжайға ( IP-адрес) ие. Интернет желісіндегі әрбір компьютер өзінің IP-адресіне ие. Бұл ретте бір желілік интерфейс аясында бірнеше желілік порттар болуы мүмкін. Желілік қосылымды орнату үшін клиент қолданбасы бос портты таңдап, қашықтағы желілік интерфейстің белгілі бір нөмірімен порт тыңдайтын серверлік қосымшамен (listen) байланыс орнатуы тиіс. IP-адрес және порт сокет (ұя) - желілік коммуникацияның бастапқы (соңғы) нүктесін сипаттайды. TCP / IP байланысын құру үшін екі сокет қажет: біреуі жергілікті машинада, екіншісі - алыстан. Осылайша, әрбір желілік қосылым жергілікті машинада IP-мекен-жайы мен порты, сондай-ақ қашықтағы машинада IP-мекен-жайы мен порты бар.

Socket модулі Python сокеттерімен жұмыс істеуге мүмкіндік береді. Сокеттер OSI (Open Systems Interconnection, ашық жүйелердің өзара әрекеттесуі) жеті деңгейлі моделіне сәйкес көлік деңгейін пайдаланады, яғни осы бөлімде сипатталған хаттамалардың көпшілігіне қарағанда төмен деңгейге жатады.

OSI моделі деңгейлері:

Физикалық

Физикалық сызықпен берілетін биттердің ағыны. Физикалық желі параметрлерін анықтайды.

Арна (Ethernet, PPP, ATM және т. б.).)

Физикалық бірыңғай желі шегінде Физикалық деңгейде туындайтын қателерді жеңе отырып, биттер ағыны түріндегі деректерді кодтайды және кодтайды.

Желілік (IP)

Түйіннен түйінге ақпараттық пакеттерді маршруттайды.

Көлік (TCP, UDP және т. б.).)

Екі байланыс нүктелері арасында деректерді мөлдір беруді қамтамасыз етеді.

Сеанс

Желі қатысушылары арасындағы байланыс сеансын басқарады. Қосылыстарды бастайды, үйлестіреді және аяқтайды.

Ұсыну

Деректер форматтарын түрлендіру арқылы оларды ұсыну формасынан тәуелсіз болуын қамтамасыз етеді. Бұл деңгейде деректерді ашық шифрлеу және шифрлеу орындалуы мүмкін.

(HTTP, FTP, SMTP, NNTP, POP3, IMAP және т. б.).)

Нақты желілік бағдарламаларды қолдайды. Хаттама сервис түріне байланысты.

Әрбір сокет байланыс домендерінің біріне жатады. Socket модулі UNIX және Internet домендерін қолдайды. Әрбір домен өзінің отбасы хаттамалар мен адресацияны білдіреді. Бұл мазмұндама тек Internet доменін ғана қозғайды, атап айтқанда TCP/IP және UDP/IP хаттамалары, сондықтан сокет жасаған кезде коммуникациялық доменді көрсету үшін socket.AF\_INET константасы көрсетіледі.

Мысал ретінде қарапайым клиент-серверлік жұпты қарастыру керек. Сервер жолды қабылдап, клиентке жауап береді. Желілік құрылғы кейде хост деп аталады (host), сондықтан бұл термин желілік бағдарлама жұмыс істейтін компьютерге қатысты қолданылады.

Сервер:

import socket, string

def do\_something(x):

lst = map(None, x);

lst.reverse();

return string.join(lst, "")

HOST = "" # localhost

PORT = 33333

srv = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

srv.bind((HOST, PORT))

while 1:

print "Слушаю порт 33333"

srv.listen(1)

sock, addr = srv.accept()

while 1:

pal = sock.recv(1024)

if not pal:

break

print "Получено от %s:%s:" % addr, pal

lap = do\_something(pal)

print "Отправлено %s:%s:" % addr, lap

sock.send(lap)

sock.close()

Клиент:

import socket

HOST = "" # удаленный компьютер (localhost)

PORT = 33333 # порт на удаленном компьютере

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.connect((HOST, PORT))

sock.send("ПАЛИНДРОМ")

result = sock.recv(1024)

sock.close()

print "Получено:", result

Ескертпе: Мысалда орыс әріптері қолданылады: кодтауды көрсету қажет.

Ең алдымен, серверді іске қосу керек. Сервер сокетті 33333 портында және 127.0.0.1 адресінде жергілікті машинада ашады. Осыдан кейін ол (listen () портты тыңдайды. Портта деректер пайда болған кезде кіріс байланыс ( accept) қабылданады. Accept әдісі () қосылымды орнататын қашықтан компьютердегі socket нысаны мен мекен - жайын қайтарады(жұп-IP мекен-жайы, қашықтағы машинадағы порт). Осыдан кейін клиентпен сөйлесу үшін recv() және send() әдістерін қолдануға болады. Recv-да () келесі порциядағы байттар саны беріледі. Клиенттен аз деректер келуі мүмкін.

Клиент-бағдарламаның коды анық. Байланыс әдісі() қашықтағы хостпен байланысты орнатады (келтірілген мысалда ол сол машинада орналасқан). Деректер send() әдісімен беріледі және серверде болатынына ұқсас recv () әдісімен қабылданады.

Socket модулі бірнеше қосалқы функцияларға ие. Атап айтқанда, домендік атаулар жүйесімен (DNS ):

>>> import socket

>>> socket.gethostbyname('www.onego.ru')

('www.onego.ru', [], ['195.161.136.4'])

>>> socket.gethostbyaddr('195.161.136.4')

('www.onego.ru', [], ['195.161.136.4'])

>>> socket.gethostname()

'rnd.onego.ru'

Жаңа Python нұсқаларында socket сияқты функция пайда болды .getservbyname (). Ол интернет-сервистердің атауын порттардың жалпы қабылданған нөмірлеріне түрлендіруге мүмкіндік береді:

>>> for srv in 'http', 'ftp', 'imap', 'pop3', 'smtp':

... print socket.getservbyname(srv, 'tcp'), srv

...

80 http

21 ftp

143 imap

110 pop3

25 smtp

Socket Модулінің басқа функцияларын қажет болған жағдайда құжаттама бойынша зерттеуге болады.

2. Smtplib модулі.

Интернеттегі электрондық пошта хабары клиенттен серверге және серверлер арасында негізінен SMTP ( S imple m ail T ransfer P rotocol, қарапайым пошта жіберу ХАТТАМАСЫ) хаттамасы бойынша жіберіледі . SMTP және ESMTP протоколы (SMTP кеңейтілген нұсқасы) RFC 821 және RFC 1869- да сипатталған. Үшін SMTP стандартты кітапхана модульдер бар модуль smtplib. Электрондық пошта серверімен SMTP байланысын бастау үшін алдымен SMTP класты конструкторының көмегімен SMTP сессиясын басқару үшін нысанды жасау қажет:

smtplib.SMTP([host[, port]])

Host және port параметрлері пошта жіберілетін SMTP серверінің мекен-жайы мен портын береді . Әдепкі, port=25. Егер host орнатылса, конструктор өзі қосылымды орнатады, әйтпесе connect () әдісін бөлек шақыруға тура келеді . SMTP класының даналары барлық кең таралған SMTP протоколдары үшін әдістерге ие, бірақ поштаны жіберу үшін sendmail() және quit конструктор шақыру жеткілікті.():

# -\*- coding: cp1251 -\*-

from smtplib import SMTP

fromaddr = "student@mail.ru" # От кого

toaddr = "rnd@onego.ru" # Кому

message = """From: Student <%(fromaddr)s>

To: Lecturer <%(toaddr)s>

Subject: From Python course student

MIME-Version: 1.0

Content-Type: text/plain; charset=Windows-1251

Content-Transfer-Encoding: 8bit

Здравствуйте! Я изучаю курс по языку Python и

отправляю письмо его автору.

"""

connect = SMTP('mail.onego.ru')

connect.set\_debuglevel(1)

connect.sendmail(fromaddr, toaddr, message % vars())

connect.quit()

Toaddr хабарламасында (To өрісінде ) және жіберу кезінде сәйкес келмеуі мүмкін екенін байқаған жөн . SMTP-сессия барысында алушы мен жөнелтуші SMTP-хаттама командаларымен беріледі. Жоғарыда көрсетілген мысалдарды іске қосқан кезде экранда жөнге келтіру ақпараты пайда болады (себебі жөндеу деңгейі 1-ге тең):

send: 'ehlo rnd.onego.ru\r\n'

reply: '250-mail.onego.ru Hello as3-042.dialup.onego.ru [195.161.147.4], pleased to meet you\r\n'

send: 'mail FROM:<student@mail.ru> size=270\r\n'

reply: '250 2.1.0 <student@mail.ru>... Sender ok\r\n'

send: 'rcpt TO:<rnd@onego.ru>\r\n'

reply: '250 2.1.5 <rnd@onego.ru>... Recipient ok\r\n'

send: 'data\r\n'

reply: '354 Enter mail, end with "." on a line by itself\r\n'

send: 'From: Student <student@mail.ru>\r\n . . . '

reply: '250 2.0.0 iBPFgQ7q028433 Message accepted for delivery\r\n'

send: 'quit\r\n'

reply: '221 2.0.0 mail.onego.ru closing connection\r\n'

Бұл (бірнеше қысқартылған) жөндеу ақпаратынан клиент SMTP-серверіне (EHLO, MAIL FROM, RCPT TO, DATA, QUIT) командаларды жіберетінін көруге болады, ал ол командаларды орындайды және қайтару кодын қайтару арқылы жауап береді (reply).

Бір SMTP-сессия барысында quit () шақырмаса, қатарынан бірнеше хат жіберуге болады.

Ол үшін-қосылыстар ( helo(), ehlo(), esmtp(), helo(), mail(), rcpt(), vrfy(), send(), noop(), data (), SMTP-протоколының аттас командаларына сәйкес келетін әдістер бар.

DOCMD () әдісі арқылы SMTP серверіне еркін пәрменді де орнатуға болады. Келесі мысалда кейбір домен үшін хабарламалар кезегі сақталатын пошта серверінен SMTP протоколы бойынша өз серверіне поштаны уақыт өте келе қабылдайтын адамдар пайдалана алатын қарапайым сценарий көрсетілген:

from smtplib import SMTP

connect = SMTP('mx.abcde.ru')

connect.set\_debuglevel(1)

connect.docmd("ETRN rnd.abcde.ru")

connect.quit()

Бұл қарапайым сценарий серверге ұсынады mx.abcde.ru негізгі домен пошта серверімен байланысуға тырысу rnd.abcde.ru және оған жиналған барлық поштаны жіберіңіз.

Smtplib сыныбымен жұмыс істеу кезінде.SMTP әртүрлі ерекшеліктер қозғалуы мүмкін. Кейбір тағайындау төменде келтірілген:

smtplib.SMTPException

Модульдің барлық ерекшеліктеріне арналған базалық сынып.

smtplib.SMTPServerDisconnected

Сервер кенеттен байланыс үзілді (немесе сервермен байланыс орнатылмаған).

smtplib.SMTP Response Exception

SMTP-сервер жауап коды бар барлық ерекшеліктер үшін базалық класс.

smtplib.SMTP Sender Refused

Жіберуші қабылдамады

smtplib.SMTPRecipientsRefused

Барлық алушылар сервермен қабылданбайды.

smtplib.SMTPDataError

Сервер осы хабарға белгісіз кодпен жауап берді.

smtplib.SMTP Connect Error

Байланыс орнату қатесі.

smtplib.SMTP Helo Error

Сервер HELO командасына дұрыс жауап бермеді немесе оны қабылдамады.

**Дәріс №11**

**Дәріс тақырыбы:** Мәліметтер базасымен жұмыс.

**Дәрістің мақсаты:** Мәліметтер базасымен жұмыс жасауды қарастыру, талдау.

Тақырып бойынша қарастырылатын сұрақтар:

1. Реляциялық ДҚБЖ негізгі ұғымдары.

2. Модуль интерфейсі.

1. Реляциялық ДҚБЖ негізгі ұғымдары.

Реляциялық деректер қоры-бұл деректер кестесі.

Кесте-жолдар мен бағандардан тұратын тікбұрышты матрица. Кесте (relation) қатынасын көрсетеді.

Жол-баған жолдарынан тұратын жазба. Әрбір өрісте кейбір мән немесе арнайы NULL (бос) мәні болуы мүмкін. Кестеде еркін жолдар саны болуы мүмкін. Реляциялық модель үшін жолдардың орналасу тәртібі анықталмаған және маңызды емес.

Кестедегі әр бағанның өз аты мен түрі бар.

DB-API 2 дегеніміз не.

Атауына енгізілген аббревиатуралар екі ұғымды біріктіреді: DB (Database, деректер қоры) және API (Application Program Interface, қолданбалы бағдарламаның интерфейсі).

Осылайша, DB-API деректер базасымен қолданбалы бағдарламаның интерфейсін анықтайды. Бұл интерфейс төменде сипатталған, Python-бағдарламалардың деректер базасымен байланысы үшін қызмет ететін барлық кеңею модульдерін іске асыруы тиіс. Бірыңғай API (қазіргі уақытта оның екінші нұсқасы) пайдаланылатын деректер базасының маркасынан абстракциялауға мүмкіндік береді, Қажет болған жағдайда функциялар мен әдістердің бір жиынтығын зерттей отырып, бір ДББЖ-ны екіншісіне оңай ауыстыруға мүмкіндік береді.

DB-API 2.0 PEP 249-да сипатталған (сайт http://www.python.org/dev/peps/pep-0249/) және төменде берілген сипаттама оған негізделген.

Сипаттамасы DB API 2.0

DB API 2.0 деректер базасымен жұмыс істеу үшін кеңейту Модулінің интерфейстерін, объект-базамен қосылу әдістерін, ағымдағы өңделетін жазбаның объект-курсорын, деректер типтері мен олардың конструкторларына арналған әр түрлі объектілерді регламенттейді, сондай-ақ модульдерді іске асыру бойынша әзірлеушілерге арналған ұсынымдарды қамтиды. Бүгінгі күні Python көптеген белгілі деректер базасын кеңейту модульдері арқылы қолдайды (мекен-жай бойынша web-бетте нақтылауға болады) http://www.python.org/topics/database/). төменде жаңа модульдерді әзірлеушілерге арналған ұсынымдарды қоспағанда, DB-API-дің барлық дерлік ережелері қарастырылады.

2. Модуль интерфейсі.

Мұнда DB-API 2.0 талаптарын қанағаттандыру үшін модульді ұсыну керек.

Деректер қорына кіру қосылым объектісі (connection object) арқылы жүзеге асырылады. DB-API-үйлесімді модуль қосылымдар класы үшін connect() конструктор функциясын ұсынуы тиіс. Конструкторда келесі атаулы параметрлер болуы керек:

* dsn деректер көзінің атауы
* user пайдаланушы аты
* Password Пароль
* host ДББЖ жұмыс істейтін хост Мекен-жайы
* database деректер қорының атауы.
* Нысан әдістері-қосылыстар кейінірек қарастырылады.

Модуль оның негізгі сипаттамалары бар константаларды анықтайды:

* apilevel нұсқасы DB-API ("1.0" немесе "2.0").
* threadsafety басқару ағындарын пайдалану кезінде модуль мүмкіндіктерін сипаттайтын бүтін тұрақты:
* 0 Модуль ағындарды қолдамайды.
* 1 ағындар модульді бірге пайдалана алады, бірақ қосылыстар емес.
* 2 ағындар модуль мен қосылыстарды бірлесіп пайдалана алады.
* 3 ағындар модульді, қосылымдарды және меңзерлерді бірлесіп пайдалана алады. (Бірлесіп пайдалану деп аталған ресурстарды семафорларды қолданбай пайдалану мүмкіндігі түсініледі).
* paramstyle параметрлерді қойған кезде қолданылатын белгі түрі. Осы константаның келесі мәндері мүмкін:
* "format" ANSI C стилінде пішімдеу (мысалы, "% s", "% i").
* "pyformat "Python ("%(item)s стилінде пішімде аталған спецификаторларды пайдалану" )
* "qmark" белгілерді пайдалану "?"параметрлерді орнату орындарын белгілеу үшін.
* "numeric" позициялар нөмірлерін пайдалану ( ": 1").
* "named "орнатылған параметрлердің атын пайдалану (": name").

Модуль типтік ерекше жағдайларды белгілеу үшін бірқатар ерекшеліктерді анықтауы тиіс: Warning (ескерту), Error (Қате), InterfaceError (интерфейс қатесі), DatabaseError (деректер базасына қатысты қате). Сондай-ақ, осы соңғы ерекшеліктің Ішкі сыныбы: DataError (деректерді өңдеу қатесі), OperationalError (жұмыс қатесі немесе деректер қоры қатесі), IntegrityError (деректер қоры тұтастығының қатесі), InternalError (деректер қорының ішкі қатесі), ProgrammingError (бағдарламалық қате, мысалы, SQL-сұранымның синтаксисіндегі қате), NotSupportedError (сұралған сипаттың қолдауы болмаған жағдайда).

Нысан-қосылым

Объект-connect () функциясын сәтті шақыру нәтижесінде алынған қосылым келесі әдістерге ие болуы тиіс:

* close () деректер базасымен байланысты жабады.
* commit () транзакцияны аяқтайды.
* rollback () басталған транзакция (бастапқы күйін қалпына келтіреді). Аяқталмаған транзакция кезінде байланысты жабу транзакцияны автоматты түрде қайтарады.
* cursor () осы қосылымды пайдаланатын курсорды қайтарады. Егер деректер базасы меңзерді қолдамаса, жұптастыру модулі оларды имитациялауы тиіс.

Транзакция деп Деректер базасын өзгертетін бір немесе бірнеше операциялардан тұратын топ түсініледі. Транзакция деректер қорына логикалық бөлінбейтін операцияларға сәйкес келеді, ал транзакцияны ішінара орындау ДБ тұтастығын бұзуға әкеледі. Мысалы, бірінші шотты азайту және екіншісін ұлғайту бойынша бір шоттан басқа операцияларға ақша аудару кезінде транзакция болып табылады. Commit() және rollback() әдістері транзакцияның басталуы мен аяқталуын айқын көрсетеді. Айтпақшы, барлық деректер қоры транзакция механизмін қолдамайды.

Модульдің DB-API 2.0 іске асырылуына байланысты осы қосылыс курсорларының жұмысын жалғастыратын объект-қосылысқа сілтемені сақтау қажет. Атап айтқанда, бұл нысан-курсорды бірден алуға болмайтынын білдіреді. Сондай-ақ, элементті-қосылымды жергілікті айнымалы функциядан немесе нысан-меңзерден қайтару арқылы қалдыруға болмайды.

Объект-курсор

Курсор (ағылш . cursor-current Set Of Records, ағымдағы жазбалар жиынтығы) сұрау нәтижесімен жұмыс істеу үшін қызмет етеді. Сұраныс нәтижесі әдетте бір немесе бірнеше тікбұрышты кестелер-өрістер және жол-жазбалары бар. Қолданба алынған кестелер мен жазбаларды бір кестеден оқи және өңдей алады, сондықтан курсорда ағымдағы кесте мен жазбалар туралы ақпарат сақталады. Кез келген уақытта белгілі бір курсор бір SQL-нұсқаулықты орындауға байланысты.

Курсордың атрибуттары да DB-API анықталған:

arraysize Fetchmany () әдісімен қайтарылатын жазбалар санына тең Атрибут. Әдепкі бойынша 1 тең.

callproc (procname[, params]) сақталатын procname процедурасын params тізбегінің өзгермелі параметрлерімен шақырады. Сақталған рәсім кейбір реттілік параметрлерінің мәнін өзгерте алады. Әдіс fetch-әдістері арқылы қол жеткізу нәтижесін қайтаруы мүмкін.

close () курсорды жабады.

description бұл тек оқу үшін қол жетімді атрибут-семиэлементтік тізбектерден тізбектілік. Осы тізбектердің әрқайсысы нәтиженің бір бағанын сипаттайтын ақпаратты қамтиды:

(name, type\_code, display\_size, internal\_size, precision, scale, null\_ok)

Алғашқы екі элемент (аты және түрі) міндетті, ал қалғандарының орнына (шығару өлшемі, ішкі өлшемі, дәлдігі, масштабы, бос мәнді орындау мүмкіндігі) None мәні болуы мүмкін. Бұл атрибут мәндерді қайтармайтын операциялар үшін None тең болуы мүмкін.

execute (operation[, parameters]) деректер базасына немесе ДББЖ командасын сұранымды орындайды. Параметрлер (parameters ) деректер базасында жоғарыда сипатталған paramstyle атрибуттарына сәйкес ұсынылуы мүмкін.

executemany (operation, seq\_of\_parameters) берілген үлгіге параметрлерді орната отырып, сұраулар немесе командалар сериясын орындайды. Seq\_of\_parameters параметрі параметрлер жиынтықтарының ретін анықтайды.

fetchall () сұрау нәтижесінің барлық (немесе барлық қалған) жазбаларын қайтарады.

fetchmany ([size]) сұрау нәтижесінен келесі бірнеше жазбаларды кезектілік ретімен қайтарады. Бос бірізділік деректер жоқ дегенді білдіреді. Қосымша size параметрі қайтарылатын жазбалар санын көрсетеді (нақты қайтарылатын жазбалар аз болуы мүмкін). Әдепкі size курсордың arraysize атрибутына тең.

fetchone () деректер болмаған кезде сұрау нәтижесінен немесе None келесі жазбаны (ретпен) қайтарады.

nextset () курсорды сұрау нәтижесінде алынған келесі деректер жиынтығының басына ауыстырады (бұл ретте алдыңғы жиынтықтағы жазбалардың бөлігі оқылмаған күйінде қалуы мүмкін). Егер жиынтық болмаса, None қайтарады. Барлық деректер қоры бір операция үшін бірнеше нәтиже жиынтығын қайтаруды қолдамайды.

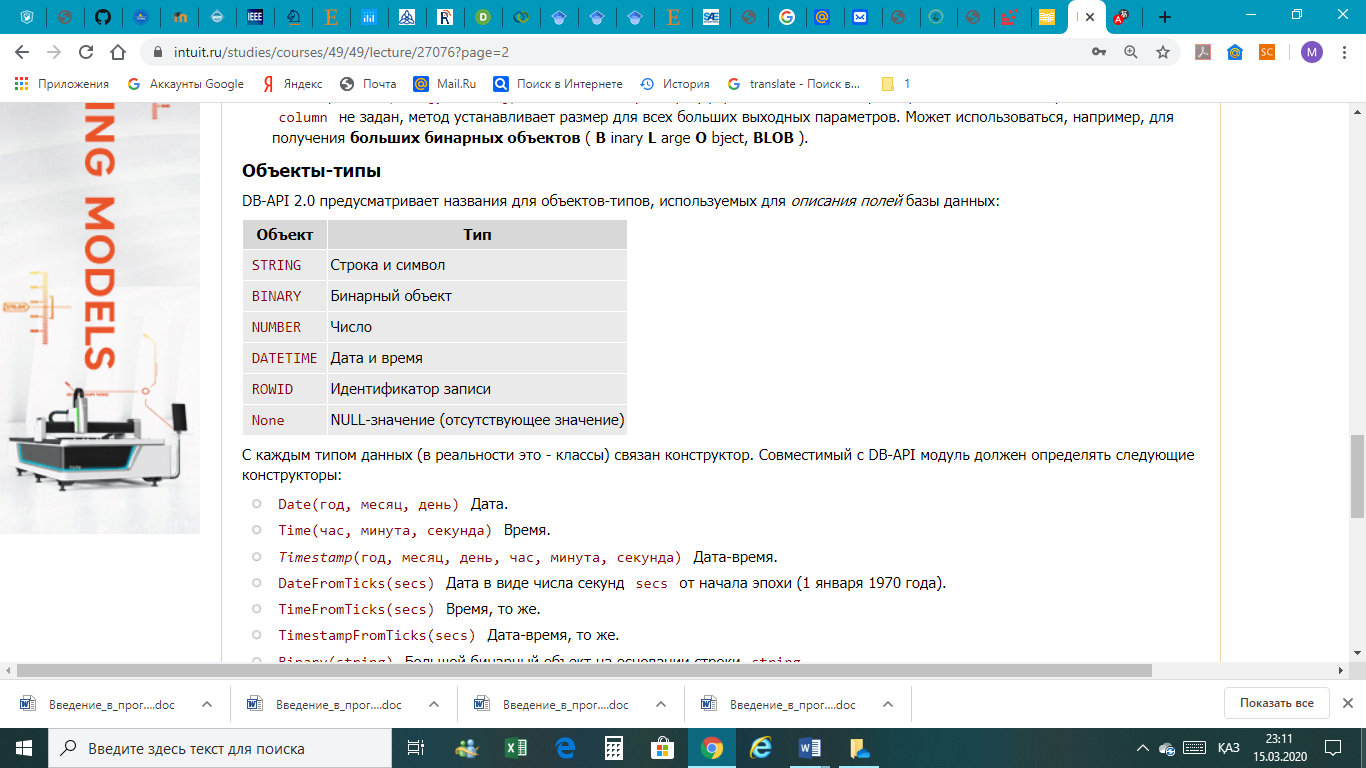
rowcount соңғы сұрау нәтижесінде алынған немесе қозғалған жазбалар саны. Execute сұраулары болмаған немесе жазбалардың санын көрсету мүмкін болмаған жағдайда -1 тең.

setinputsizes (sizes) операцияларда пайдаланылатын параметрлер үшін жад аймағын алдын ала анықтайды. Sizes аргументі әрбір элемент бір кіріс параметріне сәйкес келетін тізбекті анықтайды. Элемент тиісті параметрдің түрі немесе жолдың ұзындығын білдіретін бүтін сан болуы мүмкін. Егер кіріс параметрінің мөлшері туралы ештеңе алдын ала айтылмаса немесе ол өте үлкен деп болжанса, ол сондай-ақ None мәні болуы мүмкін. Әдіс execute әдістеріне дейін туындауы тиіс.

setoutputsize (size[, column]) column нөмірі бар бағаннан шығу параметрі үшін буфер өлшемін орнатады. Егер column орнатылмаса, әдіс барлық Үлкен Шығыс параметрлеріне өлшем орнатады. Мысалы, үлкен екілік нысандарды ( B inary L arge O bject, BLOB) алу үшін пайдаланылуы мүмкін.

Нысандар-түрлері

DB-API 2.0 деректер қорының өрістерін сипаттау үшін пайдаланылатын нысандар-типтері үшін атауларды қарастырады:



Әрбір деректер түріне (шын мәнінде бұл - сыныптар) конструктор байланысты. DB-API үйлесімді модуль келесі конструкторларды анықтауы тиіс:

* Date(год, месяц, день) Дата.
* Time(час, минута, секунда) Время.
* *Timestamp*(год, месяц, день, час, минута, секунда) Дата-время.
* DateFromTicks(secs) Дата в виде числа секунд secs от начала эпохи (1 января 1970 года).
* TimeFromTicks(secs) Время, то же.
* TimestampFromTicks(secs) Дата-время, то же.
* Binary(string) Большой бинарный объект на основании строки string.

**Дәріс №12**

**Дәріс тақырыбы:** Көп ағынды есептеулер.

**Дәрістің мақсаты:** Көп ағымды есептеулерді қарастыру, талдау.

Тақырып бойынша қарастырылатын сұрақтар:

1. Басқару ағыны туралы.

2. Threading Модулінің функциялары.

3. Thread Класы.

4. Таймер.

1. Басқару ағыны туралы.

Қазіргі заманғы операциялық жүйеде, тіпті ерекше ештеңе орындамайтын, бір мезгілде бірнеше процесс (processes) жұмыс істей алады. Мысалы, Бағдарламаны іске қосқан кезде жаңа процесс іске қосылады. Процестерді басқару функциялары Python тілінің стандартты os модулінде табуға болады. Мұнда ағындар туралы сөз болады.

Басқару ағындары (threads) бір процесс шеңберінде қалыптасады және жұмыс істейді. Бір ағынды қосымшада (қосымша ағындарды пайдаланбайтын бағдарламада) тек бір ғана басқару ағыны бар. Айтқанда упрощенно, іске қосу кезінде бағдарламаның бұл ағыны дәйекті орындайды встречаемые бағдарламасында операторлар, бағыт ала отыра бірі бойынша баламалы тармақтарының оператордың таңдау арқылы өтеді цикл денесін керегін саны таңдалады орнына өңдеуге алып тастау қозғау кезінде алып тастау. Кез келген уақытта Python интерпретаторы келесі орындау үшін қандай команданы біледі. Команда орындалғаннан кейін қандай командаға басқаруды беретіні белгілі болады. Бұл жіп бағдарламаны орындау барысында үздіксіз және ол аяқталғаннан кейін ғана үзіледі.

Енді бағдарламаның кейбір нүктесінде жіп ажыратылып, әр ағын өз жолымен жүретінін елестете аласыз. Пайда болған ағындардың әрқайсысы одан әрі бірнеше рет үлестірілуі мүмкін. (Бұл ретте ағындардың бірі әрқашан басты болып қалады және оның аяқталуы бүкіл бағдарламаның аяқталуын білдіреді.) Әр уақытта интерпретатор қандай команданы орындау керектігін біледі және әр ағынға уақыт кванттарын бөледі. Мұндай, бағдарламаны орындау тетігінің шамалы күрделенуі іс жүзінде бағдарламада сапалы өзгерістерді талап етеді - өйткені ағындардың қызметі келісілуі тиіс. Ағындарды бір уақытта бір нысанды өзгертіп, осындай өзгерістің нәтижесін, ең алдымен, объектінің тұтастығын бұзуына жол беруге болмайды.

Ағындарды келісудің классикалық құралдарының бірі Семафор деп аталатын объектілер болып табылады. Семафорлар кодтың кейбір бөлігін бірнеше ағындармен бір уақытта орындауға жол бермейді. Ең қарапайым семафор-құлып (lock) немесе mutex (ағылшын тілінен mutually exclusive, өзара ажыратқыш). Ағын кодты орындауды жалғастыру үшін, ол алдымен құлыпты басып алуы керек. Құлыпты басып алғаннан кейін ағын кодтың белгілі бір бөлігін орындайды және содан кейін құлыпты босатады, содан кейін басқа ағынның оны алуы және құлыппен қорғалатын бағдарлама учаскесін одан әрі орындауы үшін. Басқа құлып ағысымен соқтығысқан ағыс, әдетте оның босатылуын күтеді.

Python тілінде көп дәлдікті қолдау бірнеше модульдерді пайдалану арқылы қол жетімді. Threading стандартты модулінде көп ағынды (multithreading) бағдарламаларды әзірлеу үшін қажетті сыныптар анықталған: семафорлардың бірнеше түрі (lock, RLock және semaphore класстары ) және ағындар арасындағы өзара әрекеттесудің басқа механизмдері (Event және Condition класстары ), біраз уақыттан кейін функцияны іске қосу үшін Timer классы. Queue модулі бірден бірнеше ағындарды пайдалана алатын кезекті іске асырады. Thread стандартты модулінде ағындарды құру және (төмен деңгейлі) басқару үшін Thread класы анықталды.

Көп ағынды бағдарламаның үлгісі

Келесі мысалда екі қосымша ағын жасалады, олар әр түрлі стандартты шығаруға шығарылады:

import threading

def proc(n):

print "Процесс", n

p1 = threading.Thread(target=proc, name="t1", args=["1"])

p2 = threading.Thread(target=proc, name="t2", args=["2"])

p1.start()

p2.start()

Алдымен Thread класының екі нысаны алынады, содан кейін түрлі дәлелдермен іске қосылады. Бұл жағдайда ағындарда бір proc () функциясы жұмыс істейді, онда Thread класты конструктордың args атаулы параметрінде берілген бір дәлел беріледі. Start әдісі() жаңа ағысты іске қосу үшін қызмет ететінін болжау қиын емес. Осылайша, келтірілген мысалда үш ағын жұмыс істейді: негізгі және екі қосымша ("t1" және "t2" аттарымен).

2. Threading Модулінің функциялары.

Мұнда қолданылатын threading модулінде ағындар туралы ақпарат алуға мүмкіндік бар:

activeCount () қазіргі уақытта Thread класының белсенді даналарының санын қайтарады. Шын мәнінде, бұл len(threading.enumerate ()).

currentThread () ағымдағы нысанды қайтарады-ағын, яғни осы функцияны тудырған тиісті басқару ағыны. Егер ағын threading модулі арқылы құрылмаса, объект-ағын қысқартылған функциясымен (dummy thread object) қайтарылады.

enumerate () белсенді ағын тізімін қайтарады. Аяқталған және әлі басталған ағындар тізімге кірмейді.

3. Thread Класы.

Threading класының даналары.Thread Python-бағдарламалар ағындарын ұсынады. Ағында орындалатын әрекеттерді екі жолмен орнатуға болады : орындалатын нысанды және оның дәлелдерін сынып конструкторына беру немесе мұрагерлік арқылы run () қайта анықталған әдісімен жаңа класс алу. Бірінші әдіс жоғарыда мысалда қаралды. Класс конструкторы threading.Thread келесі дәлелдерге ие:

Thread(group, target, name, args, kwargs)

Бұл жерде group-ағын тобы (әзірге пайдаланылмайынша, None тең болуы тиіс), target - run () әдісіне байланысты болатын нысан, name - ағын аты, args және kwargs - target параметрінде берілген нысанды шақыру үшін позициялық және атаулы параметрлердің (тиісінше) реті мен сөздігі. Жоғарыда мысалда тек позициялық параметрлер ғана пайдаланылды, бірақ осы параметрлерді қолдану арқылы да орындауға болады:

import threading

def proc(n):

print "Процесс", n

p1 = threading.Thread(target=proc, name="t1", kwargs={"n": "1"})

p2 = threading.Thread(target=proc, name="t2", kwargs={"n": "2"})

p1.start()

p2.start()

Сол сияқты threading класынан мұрагерлік арқылы жасауға болады.Өз конструкторы мен run әдісін анықтау Thread():

import threading

class T(threading.Thread):

def \_\_init\_\_(self, n):

threading.Thread.\_\_init\_\_(self, name="t" + n)

self.n = n

def run(self):

print "Процесс", self.n

p1 = T("1")

p2 = T("2")

p1.start()

p2.start()

3. Таймер.

Threading класы.Timer берілген уақыттан кейін орындалуы тиіс әрекетті білдіреді. Бұл сынып threading Сынып Ішкі сыныбы .Thread, сондықтан start () әдісімен іске қосылады. Стандартты Hello, world тұжырымында басып шығаратын келесі қарапайым мысал! айтқанын түсіндіреді:

def hello():

print "Hello, world!"

t = Timer(30.0, hello)

t.start()

**Дәріс №13**

**Дәріс тақырыбы:** Пайдаланушының графикалық интерфейсі бар қосымшаларды жасау.

**Дәрістің мақсаты:** Пайдаланушының графикалық интерфейсі бар қосымшаларды жасауды қарастыру, талдау.

Тақырып бойынша қарастырылатын сұрақтар:

1. Графикалық кітапханаларға шолу.

2. Tk Негіздері.

3. Виджеттер сыныптары.

4. Виджетті жасау және конфигурациялау.

1. Графикалық кітапханаларға шолу.

Python тіліндегі бағдарламалар үшін пайдаланушының графикалық интерфейсін (GUI, G raphical U ser I nterface) құру графикалық интерфейс компоненттерінің тиісті кітапханаларының көмегімен немесе кальканы ағылшын тілінен, виджет кітапханаларын пайдалана отырып болады.

Келесі тізім толық емес, бірақ бар шешімдердің алуан түрлілігін көрсетеді:

* Tkinter көп платформалы пакет компоненттердің орналасуын жақсы басқаруға ие. Интерфейс әртүрлі платформаларда (Unix, Windows, Macintosh) бірдей көрінеді. Python стандартты жеткізіліміне кіреді. Құжаттама ретінде Фредрик Лунд жазған "An Introduction to Tkinter" ("Tkinter-ге кіріспе") нұсқаулығын пайдалануға болады: http://www.pythonware.com/library/tkinter/introduction/
* wxPython wxwidgets көп ақылы кітапханасында салынған (бұрын wxWindows деп аталды). Барлық платформалар үшін Ана болып көрінеді, белсенді жетілдірілуде, GL қолдауы жүзеге асырылды. Барлық негізгі платформалар үшін бар. Мүмкін, болашақ Python нұсқаларында Tkinter орын алады. Сайт: http://www.wxpython.org/
* PyGTK gtk+ және Gnome үшін визуалды компоненттер жиынтығы. Тек GTK платформасы үшін.
* PyQT / PyKDE Qt (UNIX немесе Windows астында) немесе KDE қолданатын адамдар үшін жақсы пакеттер.
* Pythonwin MFC айналасында салынған, win32all пакетінде қабықпен бірге жеткізіледі; тек Windows үшін.
* pyFLTK Xforms аналогы, OpenGL қолдауы. Windows және Unix платформалары үшін бар. Сайт: http://pyfltk.sourceforge.net/
* AWT, JFC, Swing Jython бірге жеткізіледі, ал Jython үшін Java пайдаланатын құралдар қол жетімді. Java платформасын қолдайды.
* anygui Python бағдарламасына арналған графикалық интерфейсті құру үшін төменгі платформадан тәуелсіз пакет. Сайт: http://anygui.sourceforge.net/
* Pythoncard HyperCard/MetaCard идеологиясы бойынша ұқсас графикалық интерфейс құрастырушысы. WxPython базасында жасалған. Сайт: <http://pythoncard.sourceforge.net/>.

Python-дан Қол жетімді түрлі графикалық кітапханаларға арналған өзекті сілтемелер тізімін мына мекен-жай бойынша табуға болады: http://phaseit.net/claird/comp.lang.python/python\_GUI.html

Кітапхана банк-клиент. Мысалы, PythonCard Wxpython пайдаланады, мысалы, Linux платформасында wxwindows көп платформалды GUI-кітапханасына негізделген, ол өз кезегінде GTK+ немесе Motif - ға негізделген, ал сол да X Window шығару үшін қолданылады. Айтпақшы, Motif үшін Python-да өз байланыстары бар.

Дәрісте TCL сценарий тілі үшін белгілі графикалық пакет - TCL/Tk үшін орама болып табылатын Tkinter пакеті қарастырылады. Бұл пакеттің мысалында пайдаланушының графикалық интерфейсін құрудың негізгі принциптерін үйрену оңай.

Графикалық интерфейс туралы.

Жалпы мақсаттағы барлық заманауи графикалық интерфейстер WIMP - Window, Icon, Menu, Pointer (терезе, иконка, мәзір, көрсеткіш) моделі бойынша құрылады. Терезелердің ішінде Графикалық интерфейстің элементтері бейнеленеді, олар қысқа виджеттер деп аталады (widget - штучка). Мәзір терезенің әр түрлі бөліктерінде орналасуы мүмкін, бірақ олардың мінез-құлықтары бірдей: олар алдын ала анықталған әрекеттер жиынтығынан әрекетті таңдау үшін қызмет етеді. Графикалық интерфейсті пайдаланушы" түсіндіреді " қажетті әрекеттерді компьютерлік бағдарламаға көрсеткіш көмегімен түсіндіреді. Әдетте көрсеткіш тінтуір меңзері немесе джойстик болып табылады, бірақ басқа да "сілтегіш" құрылғылар бар. Көмегімен иконок графикалық интерфейс иеленеді тәуелсіздігі тілі және кейбір жағдайларда мүмкіндік береді тез бағдарлай интерфейс.

Графикалық интерфейстің негізгі міндеті пайдаланушы мен компьютер арасындағы коммуникацияны жеңілдету болып табылады. Бұл туралы интерфейсті жобалау кезінде үнемі есте сақтау керек. Программисте (немесе дизайнерде) бар құралдарды қолдану графикалық интерфейсті жасау кезінде әрбір нақты жағдайда пайдаланушыға ыңғайлы виджеттерді таңдай отырып, минимумға дейін қою керек. Сонымен қатар, ең аз таң қаларлық принципін ұстану пайдалы: интерфейс формасынан оның мінез-құлқы түсінікті болуы керек. Нашар ойластырылған интерфейс, тіпті интерфейстің қасбетінде тиімді алгоритм жасырылса да, пайдаланушының бағдарламадан сезінуін бұзады. Интерфейс пайдаланушының типтік әрекеттері үшін ыңғайлы болуы керек. Көптеген қолданбалар үшін мұндай әрекеттер "шеберлер" (wizards) деп аталатын экрандардың жеке серияларына бөлінген. Алайда, егер қосымша-пайдаланушы өзіне қажетті шешімдерді құра алатын конструктор болса, типтік әрекет шешімді құру болып табылады. Типтік әрекеттерді анықтау оңай емес, сондықтан компромиссом "шеберлер" бар гибрид болуы мүмкін және өз құрылыстары үшін жақсы мүмкіндіктер. Дегенмен, графикалық интерфейс барлық жағдайларда ең тиімді интерфейс емес. Көптеген пәндік салалар үшін шешімді белгілі бір формальды тілде немесе сценарий тілінде алгоритм арқылы мәлімдеу оңайырақ.

2. Tk Негіздері.

Графикалық интерфейсі бар кез келген бағдарламаның негізгі ерекшелігі-интерактивтілік. Бағдарлама жай ғана бір нәрсе (пакеттік режимде) өзінің іске қосылуынан аяғына дейін санайды: оның әрекеттері пайдаланушының араласуына байланысты. Іс жүзінде, графикалық қосымша оқиғаларды өңдеудің шексіз циклын орындайды. Графикалық интерфейсті іске асыратын бағдарлама оқиғалы-бағытталған. Ол өзінің ішкі жағдайына сәйкес өңделетін оқиғалар интерфейсінен күтеді.

Бұл оқиғалар графикалық интерфейс элементтерінде (виджеттерде) пайда болады және осы виджеттерге бекітілген өңдеушілермен өңделеді. Виджеттердің өздері көптеген қасиеттерге ие( түсі, өлшемі, орналасуы), тиісті иерархияға (бір виджет басқасының иесі болуы мүмкін), жай-күйіне қол жеткізу әдістері бар.

Виджеттердің орналасуы (басқа виджеттердің ішінде) орналасу менеджерлері деп аталады. Виджет орналасу менеджерінің ережелері бойынша орынға орнатылады. Бұл ережелер виджет координаттарын ғана емес, оның өлшемдерін де анықтай алады. Tk-да орналасқан менеджерлердің үш түрі бар: қарапайым ораушы (pack), тор (grid) және еркін орналасуы (place).

Бірақ бұл графикалық бағдарлама үшін жеткіліксіз. Себебі, графикалық бағдарламадағы кейбір виджеттер белгілі бір жолмен өзара байланысты болуы керек. Мысалы, жылжыту жолағы мәтіндік виджетпен өзара байланысты болуы мүмкін: жолақты пайдаланған кезде виджеттегі мәтін қозғалуы тиіс және керісінше, мәтін бойынша жылжыған кезде жолақтың ағымдағы жағдайын көрсетуі тиіс. TK виджеттері арасындағы байланыс үшін виджеттер мен параметрлерді бір-біріне беретін айнымалылар қолданылады.

3. Виджеттер сыныптары.

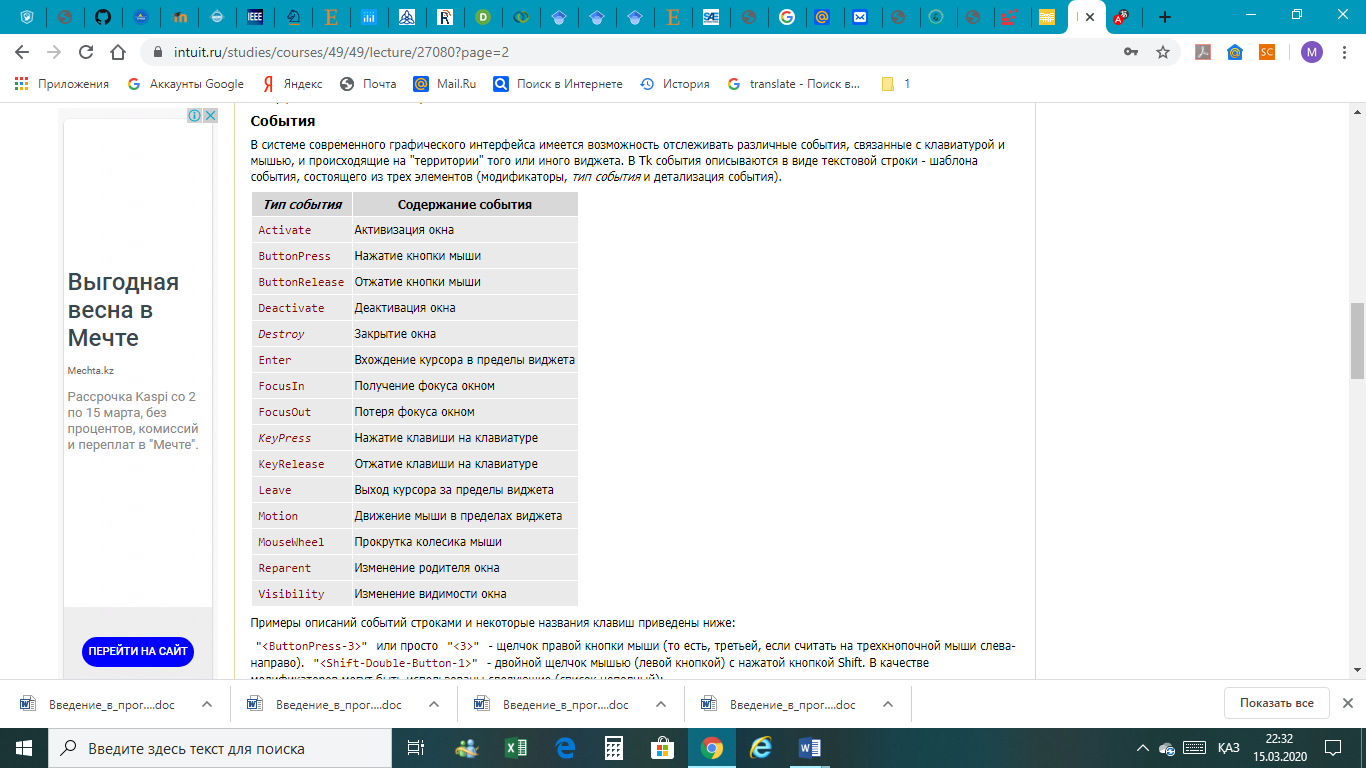
Tk кітапханасында графикалық интерфейсті құру үшін виджеттердің келесі сыныптары таңдалып алынды (алфавиттік ретпен):

* Button (түйме) кейбір әрекеттерді шақыру (белгілі бір пәрменді орындау) үшін қарапайым түйме.
* Canvas (сурет) графикалық примитивтерді шығару негізі.
* Checkbutton (құсбелгі) оған басқан кезде екі күй арасында ауыса алатын түйме.
* Entry (енгізу өрісі) мәтін жолын енгізуге болатын көлденең өріс.
* Frame (Рамка) басқа көрнекі компоненттерді қамтитын Виджет.
* Label (жазу) Виджет мәтінді немесе графикалық суретті көрсете алады.
* Listbox (тізім) пайдаланушы бір немесе бірнеше элементтерді таңдай алатын тізімді тікбұрышты рамка.
* Menu (мәзір) қалқымалы (popup) және төмен түсетін (pulldown) мәзірлерді жасауға болатын Элемент.
* Menubutton (мәзір түймесі) төмендеу мәзірі бар түйме.
* Message (хабар) жазуларға ұқсас, бірақ ұзын жолдарды бұрауға және орналасу менеджерінің талабы бойынша өлшемін өзгертуге мүмкіндік береді.
* Radiobutton (селекторлы түйме) балама мәндердің бірін ұсынуға арналған түйме. Мұндай түймелер әдетте топта әрекет етеді. Басқан кезде бір түйме тобы таңдалған, бұдан бұрын "секіріп".
* Scale (Шкала) белгілі бір диапазонда жылжу арқылы сандық мәнді орындау үшін қызмет етеді.
* Scrollbar (айналдыру жолағы) айналдыру жолағы басқа виджеттерде айналдыру көлемін көрсету үшін қолданылады. Мүмкін тік және көлденең.
* Text (пішімделген мәтін) бұл тікбұрышты виджет әр түрлі стильдерді пайдаланып мәтінді өңдеуге және пішімдеуге, мәтінге суреттер мен тіпті терезелерді енгізуге мүмкіндік береді.
* Toplevel (жоғарғы деңгей терезесі) бөлек терезе ретінде көрсетіледі және басқа виджеттер ішінде болады.

Бұл сыныптардың барлығы бір - бірімен мұрагерлік қатынастары жоқ-олар тең. Бұл жинақ көптеген жағдайларда интерфейсті құру үшін жеткілікті.

Оқиғалар.

Қазіргі заманғы графикалық интерфейс жүйесінде пернетақта мен тінтуірмен байланысты және қандай да бір виджеттің" аумағында" болып жатқан түрлі оқиғаларды бақылау мүмкіндігі бар. Оқиға елінде мәтіндік жол - үш элементтен (модификаторлар, оқиғаның түрі және оқиғаның детализациясы) тұратын оқиға үлгісі түрінде сипатталады.



**4. Виджетті жасау және конфигурациялау.**

Виджетті жасау тиісті сынып конструкторының шақыруы арқылы жүзеге асырылады . Конструктор шақыру келесі синтаксис бар:

Widget([master[, option=value, ...]])

Мұнда widget - виджет класы, masterвиджет-иесі, option және value - конфигурациялық опция және оның мәні (мұндай жұптар бірнеше болуы мүмкін).

Әрбір виджет config () (немесе configure ())) әдістерімен орнатуға және сөздіктермен жұмыс істеу әдістеріне ұқсас әдістермен оқуға болатын қасиеттерге ие. Төменде қасиеттермен жұмыс істеу үшін мүмкін синтаксис:

widget.config(option=value, ...)

widget["option"] = value

value = widget["option"]

widget.keys()

Сипат атауы Python тілінің негізгі сөзімен сәйкес келген жағдайда, атаудан кейін бір астын сызу қолданылады. Сонымен, class қасиетін class\_ сияқты, to\_ сияқты қою керек.

Виджет конфигурациясын кез келген уақытта өзгертуге болады. Бұл өзгеріс оқиғаны өңдеу циклына қайта оралғанда немесе update\_idletasks () айқын қоңырауы кезінде экранда сызылады.

Келесі мысал ішінде екі виджеті бар терезені көрсетеді. Айнымалы жазба енгізу өрісімен тікелей байланысты. Бұл мысал конфигурациялау мүмкіндіктерін көрсету үшін көптеген қасиеттерді пайдаланады:

from Tkinter import \*

tk = Tk()

tv = StringVar()

Label(tk,

textvariable=tv,

relief="groove",

borderwidth=3,

font=("Courier", 20, "bold"),

justify=LEFT,

width=50,

padx=10,

pady=20,

takefocus=False,

).pack()

Entry(tk,

textvariable=tv,

takefocus=True,

).pack()

tv.set("123")

tk.mainloop()

Нәтижесінде экранда көруге болады:



Виджеттер жасау кезінде теңшеледі. Сонымен қатар, виджеттер аттармен байланыспайды, оларды тек виджет-терезе ішінде орналастырады. Бұл мысалда textvariable (мәтіндік айнымалы), relief (рельеф), borderwidth (шекараның ені), justify (туралау), width (ені, таныстарда), padx және pady (виджет шегі мен мазмұн арасындағы пикселдегі қабат), takefocus (Tab пернесін басқан кезде фокусты қабылдау мүмкіндігі), font (қаріп, оның тапсырмасының бір тәсілі) қасиеттері қолданылған. Бұл қасиеттер көптеген виджеттер үшін өте типтік, бірақ кейде өлшем бірліктері өзгеше болуы мүмкін, мысалы, Canvas виджеті үшін ені танысу емес пикселде беріледі.

Келесі мысалда фонға, алдыңғы жоспарға (мәтінге), виджетті (шекараны жарықтандыру) белсенді күйде және фокус болмаған кезде бөлу мүмкіндіктері көрсетіледі:

from Tkinter import \*

tk = Tk()

tv = StringVar()

Entry(tk,

textvariable=tv,

takefocus=True,

borderwidth=10,

).pack()

mycolor1 = "#%02X%02X%02X" % (200, 200, 20)

Entry(tk,

textvariable=tv,

takefocus=True,

borderwidth=10,

foreground=mycolor1, # fg, текст виджета

background="#0000FF", # bg, фон виджета

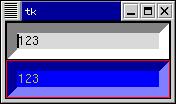
highlightcolor='green', # подсветка при фокусе

highlightbackground='red', # подсветка без фокуса

).pack()

tv.set("123")

tk.mainloop()



Қаласаңыз, барлық виджеттер үшін мәнерлі опцияларды бірден орнатуға болады: tk\_setPalette () әдісі арқылы. Бұл әдіс бойынша жоғарыда пайдаланылған сипаттардан басқа, selectForeground және select Background (алдыңғы жоспар және бөлу фоны), selectColor (таңдалған күйдегі түс, мысалы, Checkbutton), insertBackground (кірістіру нүктесінің түсі) және т.б. пайдалануға болады.

Ескертпе:

Енгізу өрісінен get () әдісі арқылы да мәнді алуға болады. Мысалы, Entry класының нысанын E деп атасаңыз, мәнді келесідей алуға болады: e.get(). Рас, бұл әдіс Text пішімделген мәтін үшін сынып даналарының get() әдісі сияқты икемділікке ие емес: тек барлық мәнді толығымен алуға болады.

**Дәріс №14**

**Дәріс тақырыбы:** Басқа бағдарламалау тілдерімен Python интеграциясы.

**Дәрістің мақсаты:** Басқа бағдарламалау тілдерімен Python интеграциясын қарастыру, талдау.

Тақырып бойынша қарастырылатын сұрақтар:

1. API.

2. Кеңейту модулін жазу.

3. SWIG пайдалану.

1. API.

Python тілінен қол жетімді Модульдер кеңейту модульдері (extension modules) есебінен кеңейтіледі. Кеңею модулдерін C немесе C++ тілінде жазуға және Python бағдарламаларынан шақыруға болады. Бұл дәрісте CPython деп аталатын Python жүзеге асыру туралы әңгіме болады (Jython, Java платформасында Python жүзеге асыру қарастырылмайды).

C тілін пайдалану қажеттілігі Python бағдарламаланған алгоритмі баяу жұмыс істесе, пайда болуы мүмкін. Мысалы, Numeric Модулінің массивтерімен жоғары өнімді операциялар (алдыңғы дәрістердің бірінде айтылған) C тілінде жазылған. Python үшін кеңейту модульдерін құру үшін қажетті мәліметтер стандартты құжаттамада, атап айтқанда "Python/C API Reference Manual" құжатында ("Python/C API"анықтамалық нұсқаулығы) толық көлемде берілген. Мұнда API туралы егжей-тегжейлі мәліметтерсіз кеңейту модулін құрудың негізгі принциптері ғана қарастырылады. Python мүмкіндіктері C++-да қол жетімді екенін байқаған жөн, олар C++ - да пайдалануға болатын C-декларацияларында көрсетілген.

Кеңею модулі үшін қажетті барлық анықтамалар Python тақырып файлында.C/C++компиляторының тақырып файлдарының жолында болуы тиіс h. Python құрастырылған кітапханалардың сол нұсқаларын пайдалану керек. Мүмкіндігінше, C/C++компиляторының сол маркасымен.

Python интерпретаторымен C кодынан байланыс Python интерпретаторында анықталған функцияларды шақыру арқылы жүзеге асырылады. Барлық функциялар Py немесе \_Py-да басталады, себебі кеңейту модулінде қақтығыстарды болдырмау үшін осындай аттармен функцияларды анықтау қажет емес.

API арқылы Python тілінің барлық кірістірілген мүмкіндіктері бар (қажет болса, бұл мәселені құжаттама бойынша егжей-тегжейлі зерттеуге болады):

* py\_main (), PyRun\_String (), PyRun\_File (), Py\_CompileString (), PyCompilerFlags () және т. б.),
* py\_initialize (), Py\_Finalize (), Py\_NewInterpreter (), Py\_EndInterpreter (), Py\_SetProgramName () және басқалар),
* py\_incref(), Py\_DECREF(), Py\_XINCREF(), Py\_XDECREF(), Py\_CLEAR (). C/C++кодында Python-нысандарды құру немесе жою кезінде талап етіледі.
* ерекшеліктер өңдеу (PyErr\* - функциялар және pyexc\_\* -тұрақты, мысалы, PyErr\_NoMemory () және PyExc\_IOError )
* операциялық жүйенің процесін басқару және сервистері ( Py\_FatalError(), Py\_Exit(), Py\_AtExit(), PyOS\_CheckStack(), және басқа функциялар/pyos макростары\* ),
* модульдерді импорттау (PyImport\_Import () және басқалар),
* сериализацияны қолдау ( PyMarshal\_WriteObjectToFile(), PyMarshal\_ReadObjectFromFile() және т. б.)
* аргументтер жолын талдау қолдау ( PyArg\_ParseTuple(), PyArg\_VaParse(), PyArg\_ParseTupleAndKeywords(), PyArg\_VaParseTupleAndKeywords(), PyArg\_UnpackTuple() және Py\_BuildValue (). Бұл функциялардың көмегімен Python функциясын шақырған кезде берілген c кодында параметрлерді алу міндеті жеңілдетіледі. Pyarg\_parse \* функциялары алынған аргументтер пішімінің жолын дәлел ретінде қабылдайды,
* абстрактты нысандардың протоколдарын қолдау: + объект ХАТТАМАСЫ ( PyObject\_Print(), PyObject\_HasAttrString(), PyObject\_GetAttrString(), PyObject\_HasAttr(), PyObject\_GetAttr(), PyObject\_RichCompare ()..., PyObject\_IsInstance (), Pyobject\_call (), PyObject\_Dir () және басқалар). Кез келген Python нысан жасау керек нәрсе + Сан ХАТТАМАСЫ ( PyNumber\_Check (), PyNumber\_Add (),..., PyNumber\_And (),..., PyNumber\_InPlaceAdd (),..., PyNumber\_Coerce (), PyNumber\_Int (),...). Кез келген элементті не істеу керек, Саны + бірізділік протоколы ( PySequence\_Check (), PySequence\_Size (), Pysequence\_repeat (), PySequence\_InPlaceConcat (),..., PySequence\_GetItem (),..., PySequence\_GetSlice (), PySequence\_Tuple (), PySequence\_Count ()...(Мысалы, сөздік көрініс) (функциялар: PyMapping\_Check (), PyMapping\_Length (), PyMapping\_HasKey (), PyMapping\_Keys ()..., PyMapping\_SetItemString (), PyMapping\_GetItemString () және т. б.) + итератордың ХАТТАМАСЫ (PyIter\_Check(), Pyiter\_next () + буфер ХАТТАМАСЫ (PyObject\_AsCharBuffer (), PyObject\_AsReadBuffer (), PyObject\_AsWriteBuffer (), PyObject\_CheckReadBuffer() )
* кірістірілген деректер түрлерін қолдау. Алдыңғы тармақта сипатталған ұқсас, бірақ нақты енгізілген деректер түрлері үшін. Мысалы: + Булевский объект ( PyBool\_Check () - pybool\_type, Py\_False-false нысаны, Py\_True-True нысаны,
* pymem\_malloc(), PyMem\_Realloc(), PyMem\_Free(), PyMem\_New(), PyMem\_Resize(), PyMem\_Del (). Әрине, C/C++ жады бөлу құралдарын да қолдануға болады, алайда бұл жағдайда Python интерпретаторы жадысын басқару артықшылықтары пайдаланылмайды (қоқыс және т.б. жинау). Сонымен қатар, жадты босату оны таңдау сияқты тәсілмен жүргізілуі керек. Тағы бір рет еске салу керек, бір жад аймағын қайта босату (сонымен қатар оны босатқаннан кейін жад аймағын пайдалану) c компиляторы тануға мүмкіндігі жоқ елеулі қателіктерге ұшырайды.
* кіріктірме типті объектілерді анықтауға арналған құрылымдар ( PyObject, PyVarObject және басқа да көптеген).

Ескертпе.

Бұл жерде хаттама деп өз даналарымен операцияларды ұйымдастыру үшін қандай да бір сыныпты қолдауы тиіс әдістер жиынтығы түсініледі. Бұл әдістер тек Python (мысалы, len(A) тізбектің ұзындығын береді) ғана емес, сонымен қатар C кодынан ( Sequence\_Length () қол жетімді.

2. Кеңейту модулін жазу.

Егер Python бағдарламасын енгізу қажет болса, онда C/C++ модулін жазу арқылы оның кеңеюі - өте кең таралған тәжірибе. Бастапқыда Python кеңею мүмкіндігін көздеді, сондықтан қазіргі уақытта өте көп C / C++-кітапханалар Python байланысына ие.

Python байланыстыру, бірақ бірнеше автоматтандырылған болуы мүмкін, дегенмен бұл шығармашылық процесс. Өйткені, егер кітапхананы Python-да қарқынды пайдалану ұйғарылса, оны байлау мүмкіндігінше мұқият жасау керек. Мүмкін, байланыстыру барысында кітапхананы пайдалануды жеңілдетуге мүмкіндік беретін объектілі-бағытталған қондырма немесе басқа да сәулеттік өзгерістер жасалады.

Мысал ретінде md5 Модулінің бастапқы кодынан үзінді келтіруге болады, ол md5-дайджесті алу үшін функцияны іске асырады. Модуль иллюстрация мақсатында келтіріледі(яғни қысқартулармен). Модуль өз деректер түрін енгізеді, MD5Type, сондықтан функциялардың жүзеге асырылуын ғана емес, сонымен қатар кірістірілген типті сипаттау әдісін де көруге болады. Осы курс аясында кеңейту модульдерін бағдарламалау барлық қырлылығын оқып, ең бастысы осы сабақтың рухын түсіну. Дәріс курсы авторының пікіріне Қос слэш көрсетеді //:

// заголовочные файлы

#include "Python.h"

#include "md5.h"

// В частности, в заголовочном файле md5.h есть следующие определения:

// typedef unsigned char \*POINTER;

// typedef unsigned int UINT4;

// typedef struct {

// UINT4 state[4]; /\* state (ABCD) \*/

// UINT4 count[2]; /\* number of bits, modulo 2^64 (lsb first) \*/

// unsigned char buffer[64]; /\* input buffer \*/

// } MD5\_CTX;

// Структура объекта MD5type

typedef struct {

PyObject\_HEAD

MD5\_CTX md5; /\* the context holder \*/

} md5object;

// Определение типа объекта MD5type

static PyTypeObject MD5type;

// Макрос проверки типа MD5type

#define is\_md5object(v) ((v)->ob\_type == &MD5type)

// Порождение объекта типа MD5type

static md5object \*

newmd5object(void)

{

md5object \*md5p;

md5p = PyObject\_New(md5object, &MD5type);

if (md5p == NULL)

return NULL; // не хватило памяти

MD5Init(&md5p->md5); // инициализация

return md5p;

}

// Определения методов

// Освобождение памяти из-под объекта

static void

md5\_dealloc(md5object \*md5p) { PyObject\_Del(md5p); }

static PyObject \*

md5\_update(md5object \*self, PyObject \*args)

{

unsigned char \*cp;

int len;

// разбор строки аргументов. Формат указывает следующее:

// s# - один параметр, строка (заданная указателем и длиной)

// : - разделитель

// update - название метода

if (!PyArg\_ParseTuple(args, "s#:update", &cp, &len))

return NULL;

MD5Update(&self->md5, cp, len);

// Даже возврат None требует увеличения счетчика ссылок

Py\_INCREF(Py\_None);

return Py\_None;

}

// Строка документации метода update

PyDoc\_STRVAR(update\_doc,

"update (arg)\n\

\n\

Update the md5 object with the string arg. Repeated calls are\n\

equivalent to a single call with the concatenation of all the\n\

arguments.");

// Метод digest

static PyObject \*

md5\_digest(md5object \*self)

{

MD5\_CTX mdContext;

unsigned char aDigest[16];

/\* make a temporary copy, and perform the final \*/

mdContext = self->md5;

MD5Final(aDigest, &mdContext);

// результат возвращается в виде строки

return PyString\_FromStringAndSize((char \*)aDigest, 16);

}

// и строка документации

PyDoc\_STRVAR(digest\_doc, "digest() -> string\n\ ...");

static PyObject \*

md5\_hexdigest(md5object \*self)

{

// Реализация метода на C

}

PyDoc\_STRVAR(hexdigest\_doc, "hexdigest() -> string\n...");

// Здесь было определение метода copy()

// Методы объекта в сборе.

// Для каждого метода указывается название, имя метода на C

// (с приведением к типу PyCFunction), способ передачи аргументов:

// METH\_VARARGS (переменное кол-во) или METH\_NOARGS (нет аргументов)

// В конце массива - метка окончания спиcка аргументов.

static PyMethodDef md5\_methods[] = {

{"update", (PyCFunction)md5\_update, METH\_VARARGS, update\_doc},

{"digest", (PyCFunction)md5\_digest, METH\_NOARGS, digest\_doc},

{"hexdigest", (PyCFunction)md5\_hexdigest, METH\_NOARGS, hexdigest\_doc},

{"copy", (PyCFunction)md5\_copy, METH\_NOARGS, copy\_doc},

{NULL, NULL} /\* sentinel \*/

};

// Атрибуты md5-объекта обслуживает эта функция, реализуя метод

// getattr.

static PyObject \*

md5\_getattr(md5object \*self, char \*name)

{

// атрибут-данное digest\_size

if (strcmp(name, "digest\_size") == 0) {

return PyInt\_FromLong(16);

}

// поиск атрибута-метода ведется в списке

return Py\_FindMethod(md5\_methods, (PyObject \*)self, name);

}

// Строка документации к модулю md5

PyDoc\_STRVAR(module\_doc, "This module implements ...");

// Строка документации к классу md5

PyDoc\_STRVAR(md5type\_doc, "An md5 represents the object...");

// Структура для объекта MD5type с описаниями для интерпретатора

static PyTypeObject MD5type = {

PyObject\_HEAD\_INIT(NULL)

0, /\*ob\_size\*/

"md5.md5", /\*tp\_name\*/

sizeof(md5object), /\*tp\_size\*/

0, /\*tp\_itemsize\*/

/\* methods \*/

(destructor)md5\_dealloc, /\*tp\_dealloc\*/

0, /\*tp\_print\*/

(getattrfunc)md5\_getattr, /\*tp\_getattr\*/

0, /\*tp\_setattr\*/

0, /\*tp\_compare\*/

0, /\*tp\_repr\*/

0, /\*tp\_as\_number\*/

0, /\*tp\_as\_sequence\*/

0, /\*tp\_as\_mapping\*/

0, /\*tp\_hash\*/

0, /\*tp\_call\*/

0, /\*tp\_str\*/

0, /\*tp\_getattro\*/

0, /\*tp\_setattro\*/

0, /\*tp\_as\_buffer\*/

0, /\*tp\_xxx4\*/

md5type\_doc, /\*tp\_doc\*/

};

// Функции модуля md5:

// Функция new() для получения нового объекта типа md5type

static PyObject \*

MD5\_new(PyObject \*self, PyObject \*args)

{

md5object \*md5p;

unsigned char \*cp = NULL;

int len = 0;

// Разбор параметров. Здесь вертикальная черта

// в строке формата означает окончание

// списка обязательных параметров.

// Остальное - как и выше: s# - строка, после : - имя

if (!PyArg\_ParseTuple(args, "|s#:new", &cp, &len))

return NULL;

if ((md5p = newmd5object()) == NULL)

return NULL;

// Если был задан параметр cp:

if (cp)

MD5Update(&md5p->md5, cp, len);

return (PyObject \*)md5p;

}

// Строка документации для new()

PyDoc\_STRVAR(new\_doc, "new([arg]) -> md5 object ...");

// Список функций, которые данный модуль экспортирует

static PyMethodDef md5\_functions[] = {

{"new", (PyCFunction)MD5\_new, METH\_VARARGS, new\_doc},

{"md5", (PyCFunction)MD5\_new, METH\_VARARGS, new\_doc},

{NULL, NULL} /\* Sentinel \*/

};

// Следует заметить, что md5 - то же самое, что new. Эта функция оставлена для

// обратной совместимости со старым модулем md5

// Инициализация модуля

PyMODINIT\_FUNC

initmd5(void)

{

PyObject \*m, \*d;

MD5type.ob\_type = &PyType\_Type;

// Инициализируется модуль

m = Py\_InitModule3("md5", md5\_functions, module\_doc);

// Получается словарь с именами модуля

d = PyModule\_GetDict(m);

// Добавляется атрибут MD5Type (тип md5-объекта) к словарю

PyDict\_SetItemString(d, "MD5Type", (PyObject \*)&MD5type);

// Добавляется целая константа digest\_size к модулю

PyModule\_AddIntConstant(m, "digest\_size", 16);

}

Осы мысал негізінде C/API бойынша құжаттамамен және Python стандартты жеткізуден "Extending and Embedding" ("кеңейту және кірістіру") құжатымен танысып, өз кеңейту модульдерін құруға болады. Өз модулін құруға кірісер алдында, бұл орынды екеніне көз жеткізу керек: қолайлы модуль әлі құрылмаған және таза Python түрінде іске асыру тиімсіз. Егер шын мәнінде пайдалы модуль жасалса, оны Python жеткізу үшін ұсынуға болады. Ол үшін электрондық пошта арқылы әзірлеушілердің біреуімен немесе "патч" түрінде модульді ұсыну қажет. http://sourceforge.net ескерту.

3. SWIG пайдалану.

SWIG (Simplified Wrapper and Interface Generator, оңайлатылған ораушы және интерфейстер генераторы) - бұл C және C++ жазылған кітапханаларды, сондай-ақ басқа да бағдарламалау тілдерінде, оның ішінде (соңғы емес!) Python. Айта кету керек, SWING C++ барлық мүмкіндіктеріне, соның ішінде алдын ала өңдеу, сыныптар, көрсеткіштер, мұрагерлік және тіпті c++үлгілеріне толық қолдау көрсетеді. Соңғы шаблондар кітапханасына интерфейс жасау қажет болса өте маңызды.

SWIG қолдану компилятор мен компиляторды қолдана білсеңіз жеткілікті (C/C++бағдарламалау кезінде қажет).

SWIG пайдалану қарапайым мысал

Мысалы, кейбір функцияны іске асыратын C бағдарламасы бар (бұл жолда әр түрлі таңбалардың пайда болу жиілігін есептеу болсын):

/\* File : freq.c \*/

#include <stdlib.h>

int \* frequency(char s[]) {

int \*freq;

char \*ptr;

freq = (int\*)(calloc(256, sizeof(int)));

if (freq != NULL)

for (ptr = s; \*ptr; ptr++)

freq[\*ptr] += 1;

return freq;

}

Python функциясын пайдалану үшін интерфейс файлын жазу керек (кеңейту .i) мынадай мазмұндағы шамамен:

/\* File : freq.i \*/

%module freq

%typemap(out) int \* {

int i;

$result = PyTuple\_New(256);

for(i=0; i<256; i++)

PyTuple\_SetItem($result, i, PyLong\_FromLong($1[i]));

free($1);

}

extern int \* frequency(char s[]);

Интерфейстік файлдарда SWIG нұсқауы және C/C++-кодтың фрагменттері бар, мүмкін, макровключен (мысалы, жоғарыда: $result, $1 ). Long типті элементтердің кортежіне бүтін сандардың массивін түрлендіру үшін жиіліктер есептелген бастапқы массивтен жадыны босату қажет. Енді (gcc компиляторы пайдаланылады), кеңейту модулін жасау шамамен осылай орындалуы мүмкін:

swig -python freq.i

gcc -c -fpic freq\_wrap.c freq.c -DHAVE\_CONFIG\_H

-I/usr/local/include/python2.3 -I/usr/local/lib/python2.3/config

gcc -shared freq.o freq\_wrap.o -o \_freq.so

Осыдан кейін жұмыс каталогында файл пайда болады \_freq .so және freq.py, олар бірге және қажетті функцияға қол жеткізуге мүмкіндік береді:

>>> import freq

>>> freq.frequency("ABCDEF")[60:75]

(0L, 0L, 0L, 0L, 0L, 1L, 1L, 1L, 1L, 1L, 1L, 0L, 0L, 0L, 0L)

Сонымен қатар, freq\_wrap файлының мазмұнын көруге болады.SWIG туған c: онда SWIG өзіне қажетті басқа да көмекші анықтамалардың арасында md5 Модулінің жоғарыда сипатталған мысалына ұқсас нәрсе көруге болады. Мұнда frequency функциясы үшін орау анықтамасы бар осы файл фрагменті():

extern int \*frequency(char []);

static PyObject \*\_wrap\_frequency(PyObject \*self, PyObject \*args) {

PyObject \*resultobj;

char \*arg1 ;

int \*result;

if(!PyArg\_ParseTuple(args,(char \*)"s:frequency",&arg1)) goto fail;

result = (int \*)frequency(arg1);

{

int i;

resultobj = PyTuple\_New(256);

for(i=0; i<256; i++)

PyTuple\_SetItem(resultobj, i, PyLong\_FromLong(result[i]));

free(result);

}

return resultobj;

fail:

return NULL;

}

Жаттығу ретінде бұл анықтаманы freq файлымен салыстыру ұсынылады.i және\_wrap\_frequency функциясының ішінде не болып жатқанын түсіну (). Кеңес: сіз md5 Модулінің C-кодына тағы да түсініктеме көре аласыз .

Тағы да еске салу керек, Python қарағанда, C/C++ тілінде жадыны басқару анық түрде болуы тиіс. Сондықтан түрі түрлендірілген кезде free() функциясы қосылды. Егер мұны істемесе, жад ағуы пайда болады . Функцияны бірнеше рет орындаған кезде бұл ағуларды табуға болады:

>>> import freq

>>> for i in xrange(1000000):

... dummy = freq.frequency("ABCDEF")

>>>

Егер freq функциясы болса .frequency () жады ағуы бар, процесс өте тез барлық жадты алады.

**Дәріс №15**

**Дәріс тақырыбы:** Python тілінің интерпретаторы құрылғысы.

**Дәрістің мақсаты:** Python тілінің интерпретаторы құрылғысын қарастыру.

Тақырып бойынша қарастырылатын сұрақтар:

1. Лексикалық талдау.

2. Синтаксистік талдау.

3. Байт-кодты алу.

4. Баптау.

1. Лексикалық талдау.

Программалау тілінің лексикалық анализаторы бағдарламаның бастапқы мәтінін (жалғыз таңбадан тұратын) лексемаларға - тілдің бөлінбейтін "сөздеріне" бөледі.

Python лексемасының негізгі категориялары: идентификаторлар және кілт сөздер (NAME), литералдар (STRING, NUMBER және т.б.), операциялар (OP), бөлгіштер, шегіністерді (INDENT, DEDENT) және жолдардың ұштарын (NEWLINE) белгілеу (өзгерту) үшін арнайы лексемалар, сондай-ақ түсініктемелер (COMMENT). Лексикалық анализатор tokenize модулі арқылы қол жетімді, ал лексемалардың кодтарын анықтау Python стандартты кітапханасының token модулінде болады. Келесі мысал лексикалық талдағышты іс-әрекетте көрсетеді:

import StringIO, token, tokenize

prog\_example = """

for i in range(100): # comment

if i % 1 == 0: \

print ":", t\*\*2

""".strip()

rl = StringIO.StringIO(prog\_example).readline

for t\_type, t\_str, (br,bc), (er,ec), logl in tokenize.generate\_tokens(rl):

print "%3i %10s : %20r" % (t\_type, token.tok\_name[t\_type], t\_str)

Бірақ бұл бағдарламаны лексемаға бөліп, мысалдың бастапқы кодын шығарады:

prog\_example:

1 NAME : 'for'

1 NAME : 'i'

1 NAME : 'in'

1 NAME : 'range'

50 OP : '('

2 NUMBER : '100'

50 OP : ')'

50 OP : ':'

52 COMMENT : '# comment'

4 NEWLINE : '\n'

5 INDENT : ' '

1 NAME : 'if'

1 NAME : 'i'

50 OP : '%'

2 NUMBER : '1'

50 OP : '=='

2 NUMBER : '0'

50 OP : ':'

1 NAME : 'print'

3 STRING : '":"'

50 OP : ','

1 NAME : 't'

50 OP : '\*\*'

2 NUMBER : '2'

6 DEDENT : ''

0 ENDMARKER : ''

Іс жүзінде әртүрлі мақсаттар үшін пайдаланылуы мүмкін лексемдер ағыны алынды. Мысалы, Python тілінде синтаксистік "бояу" коды үшін. Token.tok\_name сөздігі нөмірі бойынша лексема түріне арналған мнемоникалық атауларды алуға мүмкіндік береді.

2. Синтаксистік талдау.

Бағдарламаның бастапқы мәтінін интерпретатордың байт-кодына түрлендірудің екінші сатысы бастапқы мәтінді синтаксистік талдаудан тұрады. Parser модулі бағдарламалар коды мен Python өрнектері үшін сәйкесінше синтаксистік талдау ағаштарын құру үшін suite() және Express() функцияларын қамтиды. Symbol модулі Python грамматикасының символдарының нөмірлерін, Python грамматикасының символының атауын алу үшін сөздікті қамтиды.

Келесі бағдарлама қарапайым Python ( рrg ) кодын талдайды және синтаксистік талдау ағашын (AST-нысан) туындатады, ол бірден кортежге айналуға және pprint функциясымен әдемі шығаруға болады.pprint (). Бұдан әрі символдардың нөмірлерін грамматикада олардың мнемоникалық белгілеріне (аттарына) айналдыру функциясы анықталады:

import pprint, token, parser, symbol

prg = """print 2\*2"""

pprint.pprint(parser.suite(prg).totuple())

def pprint\_ast(ast, level=0):

if type(ast) == type(()):

for a in ast:

pprint\_ast(a, level+1)

elif type(ast) == type(""):

print repr(ast)

else:

print " "\*level,

try:

print symbol.sym\_name[ast]

except:

print "token."+token.tok\_name[ast],

print

pprint\_ast(parser.suite(prg).totuple())

Бұл бағдарлама келесілерді шығарады (ағаш құрылымы шегіністермен көрсетілген):

(257,

(264,

(265,

(266,

(269,

(1, 'print'),

(292,

(293,

(294,

(295,

(297,

(298,

(299,

(300,

(301,

(302,

(303, (304, (305, (2, '2')))),

(16, '\*'),

(303, (304, (305, (2, '2')))))))))))))))),

(4, ''))),

(0, ''))

file\_input

stmt

simple\_stmt

small\_stmt

print\_stmt

token.NAME 'print'

test

and\_test

not\_test

comparison

expr

xor\_expr

and\_expr

shift\_expr

arith\_expr

term

factor

power

atom

token.NUMBER '2'

token.STAR '\*'

factor

power

atom

token.NUMBER '2'

token.NEWLINE ''

token.ENDMARKER ''

3. Байт-кодты алу.

Синтаксистік талдау ағашын алғаннан кейін, компилятор оны интерпретатор орындау үшін жарамды байт-кодқа айналдыруы тиіс. Келесі бағдарламада жеке синтаксистік талдау, Python тілінде кодты (және өрнектерді) құрастыру және орындау (есептеу) жүргізіледі:

import parser

prg = """print 2\*2"""

ast = parser.suite(prg)

code = ast.compile('filename.py')

exec code

prg = """2\*2"""

ast = parser.expr(prg)

code = ast.compile('filename1.py')

print eval(code)

Parser функциясы.suite () (немесе parser.Express () compile () әдісімен Python байт кодына жинақталатын және code кодында сақталатын AST нысанын (синтаксистік талдау ағашы) қайтарады. Енді бұл кодты exec (немесе eval () функциясы) операторының көмегімен орындауға болады.

Мұнда Python-да соңғы кодты талдау және кодты генерациялау үшін модульдерді біріктіретін compiler пакеті жақында пайда болғанын ескеру қажет. Бұл дәрісте ол қарастырылмайды,бірақ осы процестерді тереңірек зерттегісі келетіндер Python бойынша құжаттарға жүгіне алады.

Байт-кодты зерттеу.

Python-бағдарламаның байт-кодын зерттеу үшін dos модулін ("дизассемблерден" қысқару) пайдалануға болады, ол байт-кодты мнемоникалық түрде көруге мүмкіндік береді . Келесі мысал осы мүмкіндікті көрсетеді:

>>> def f():

... print 2\*2

...

>>> dis.dis(f)

2 0 LOAD\_CONST 1 (2)

3 LOAD\_CONST 1 (2)

6 BINARY\_MULTIPLY

7 PRINT\_ITEM

8 PRINT\_NEWLINE

9 LOAD\_CONST 0 (None)

12 RETURN\_VALUE

2\*2 өрнегінің мәнін есептейтін және басып шығаратын f () функциясы анықталады. Dis() функциясы dis () функциясы Python байт коды мнемоникалық аттармен ұсынылған "ассемблер" түрінде f () функциясының кодын шығарады. Бұл жағдайда, егер де, егер де, егер де, егер де, егер де, егер де, егер де, егер де, егер де, егер де, онда, онда, онда, онда, онда, онда, егер де, онда, онда, онда, онда, егер де, егер де, онда, егер де, онда, онда, онда, егер де, онда, онда, онда, мысалы, онда, онда, онда, онда, онда, егер де, онда, егер де, онда, онда, егер де, онда, егер де, онда, егер де, Return операторсыз функциясы None мәнін қайтарады. Микропроцессор үшін кодтар сияқты кейбір байт-кодтар параметрлерді қабылдайды.

Мнемоникалық атауларды dis тізімінен көруге болады.opname (төменде тек іске қосылған атаулар басылады):

>>> import dis

>>> [n for n in dis.opname if n[0] != "<"]

['STOP\_CODE', 'POP\_TOP', 'ROT\_TWO', 'ROT\_THREE', 'DUP\_TOP', 'ROT\_FOUR',

'NOP', 'UNARY\_POSITIVE', 'UNARY\_NEGATIVE', 'UNARY\_NOT', 'UNARY\_CONVERT',

'UNARY\_INVERT', 'LIST\_APPEND', 'BINARY\_POWER', 'BINARY\_MULTIPLY',

'BINARY\_DIVIDE', 'BINARY\_MODULO', 'BINARY\_ADD', 'BINARY\_SUBTRACT',

'BINARY\_SUBSCR', 'BINARY\_FLOOR\_DIVIDE', 'BINARY\_TRUE\_DIVIDE',

'INPLACE\_FLOOR\_DIVIDE', 'INPLACE\_TRUE\_DIVIDE', 'SLICE+0', 'SLICE+1',

'SLICE+2', 'SLICE+3', 'STORE\_SLICE+0', 'STORE\_SLICE+1', 'STORE\_SLICE+2',

'STORE\_SLICE+3', 'DELETE\_SLICE+0', 'DELETE\_SLICE+1', 'DELETE\_SLICE+2',

'DELETE\_SLICE+3', 'INPLACE\_ADD', 'INPLACE\_SUBTRACT', 'INPLACE\_MULTIPLY',

'INPLACE\_DIVIDE', 'INPLACE\_MODULO', 'STORE\_SUBSCR', 'DELETE\_SUBSCR',

'BINARY\_LSHIFT', 'BINARY\_RSHIFT', 'BINARY\_AND', 'BINARY\_XOR', 'BINARY\_OR',

'INPLACE\_POWER', 'GET\_ITER', 'PRINT\_EXPR', 'PRINT\_ITEM', 'PRINT\_NEWLINE',

'PRINT\_ITEM\_TO', 'PRINT\_NEWLINE\_TO', 'INPLACE\_LSHIFT', 'INPLACE\_RSHIFT',

'INPLACE\_AND', 'INPLACE\_XOR', 'INPLACE\_OR', 'BREAK\_LOOP', 'LOAD\_LOCALS',

'RETURN\_VALUE', 'IMPORT\_STAR', 'EXEC\_STMT', 'YIELD\_VALUE', 'POP\_BLOCK',

'END\_FINALLY', 'BUILD\_CLASS', 'STORE\_NAME', 'DELETE\_NAME',

'UNPACK\_SEQUENCE', 'FOR\_ITER', 'STORE\_ATTR', 'DELETE\_ATTR', 'STORE\_GLOBAL',

'DELETE\_GLOBAL', 'DUP\_TOPX', 'LOAD\_CONST', 'LOAD\_NAME', 'BUILD\_TUPLE',

'BUILD\_LIST', 'BUILD\_MAP', 'LOAD\_ATTR', 'COMPARE\_OP', 'IMPORT\_NAME',

'IMPORT\_FROM', 'JUMP\_FORWARD', 'JUMP\_IF\_FALSE', 'JUMP\_IF\_TRUE',

'JUMP\_ABSOLUTE', 'LOAD\_GLOBAL', 'CONTINUE\_LOOP', 'SETUP\_LOOP',

'SETUP\_EXCEPT', 'SETUP\_FINALLY', 'LOAD\_FAST', 'STORE\_FAST', 'DELETE\_FAST',

'RAISE\_VARARGS', 'CALL\_FUNCTION', 'MAKE\_FUNCTION', 'BUILD\_SLICE',

'MAKE\_CLOSURE', 'LOAD\_CLOSURE', 'LOAD\_DEREF', 'STORE\_DEREF',

'CALL\_FUNCTION\_VAR', 'CALL\_FUNCTION\_KW', 'CALL\_FUNCTION\_VAR\_KW',

'EXTENDED\_ARG']

LOAD мәндерді стекке жүктеу, STORE - түсіру, PRINT - басып шығару, BINARY - бинарлық операция және т. б. білдіреді.

4. Баптау.

Python тілінің и нтерпретаторында бағдарламаларды жөндеу мүмкіндіктері бар, ал стандартты жеткізуде қарапайым реттеуші - pdb бар. Келесі мысал баптауға жататын бағдарламаны және баптаудың типтік сессиясын көрсетеді:

# File myfun.py

def fun(s):

lst = []

for i in s:

lst.append(ord(i))

return lst

Сондықтан әдеттегі жөндеу процесі көрінуі мүмкін:

>>> import pdb, myfun

>>> pdb.runcall(myfun.fun, "ABCDE")

> /examples/myfun.py(4)fun()

-> lst = []

(Pdb) n

> /examples/myfun.py(5)fun()

-> for i in s:

(Pdb) n

> /examples/myfun.py(6)fun()

-> lst.append(ord(i))

(Pdb) l

1 #!/usr/bin/python

2 # File myfun.py

3 def fun(s):

4 lst = []

5 for i in s:

6 -> lst.append(ord(i))

7 return lst

[EOF]

(Pdb) p lst

[]

(Pdb) p vars()

{'i': 'A', 's': 'ABCDE', 'lst': []}

(Pdb) n

> /examples/myfun.py(5)fun()

-> for i in s:

(Pdb) p vars()

{'i': 'A', 's': 'ABCDE', 'lst': [65]}

(Pdb) n

> /examples/myfun.py(6)fun()

-> lst.append(ord(i))

(Pdb) n

> /examples/myfun.py(5)fun()

-> for i in s:

(Pdb) p vars()

{'i': 'B', 's': 'ABCDE', 'lst': [65, 66]}

(Pdb) r

- Return -

> /examples/myfun.py(7)fun()->[65, 66, 67, 68, 69]

-> return lst

(Pdb) n

[65, 66, 67, 68, 69]

>>>

Интерактивті реттеуші pdb функциясымен туындайды.runcall () және оның шақыруына (Pdb) командаларды енгізу керек. Бұл баптау сессиясының мысалында келесі командалардың кейбірі пайдаланылды: l (кодталған код фрагментін басып шығару), n (келесі жолға дейін бәрін орындау), s (әдіс немесе функцияны шақыруға тереңдей отырып, келесі қадам жасау), p (мәнді басып шығару), r (ағымдағы функциядан қайтарылғанға дейін бәрін орындау).

Әрине, Python үшін кейбір интерактивті әзірлеушінің қабықтары реттеуші функциясын ұсынады. Сонымен қатар, бағдарламаның негізгі жерлерінде print операторлары қызығушылық тудыратын параметрлерді шығару үшін ұйымдастыру оңай. Әдетте бұл мәселені оқшаулау үшін жеткілікті. CGI-сценарийлерде алдыңғы дәрістердің бірінде айтылған cgitb модулін пайдалануға болады.