PYTHONDA FUNKSIYALAR

VA MODULLAR

FUNKSIYALAR

Funksiya - bu bitta, bog'liq bo'lgan harakatni amalga oshirish uchun ishlatiladigan uyushgan, qayta ishlatilishi mumkin bo'lgan kodlar bloki. Funktsiyalar sizning arizangiz uchun yaxshiroq modullik va kodni yuqori darajada qayta ishlatilishini ta'minlaydi.

Siz allaqachon bilganingizdek, Python sizga **print ()** va shu kabi ko'plab ichki funktsiyalarni beradi. Pythonda mavjud bo'lgan tiplarni o'zgartiruvchi va ba'zi bir qo'shimcha funksiyalar bilan quyida tanishishingiz mumkin:

- **bool(x)-** rostlikka tekshirishni standart usulidan foydalanuvchi bool tipiga o`zgartirish. Agarx yolg`on bo`lsa yoki tushirib qoldirilgan bo`lsa, False qiymatini qaytaradi, aksincha esa True qaytaradi.
- **bytearray([manba, [kodlash[xatolar]]])-** bytearray ga o`zgartirish. Bytearray- 0<=x<256 diapazondagi butun sonlarni o`zgarmas ketma-ketligi. Konstruktor argumentlari bytearray() ga mos ko`rinishga ega bo`ladi.
- **complex([real],[image])-** kompleks songa o`zgartirish.
- **dict(object)-** lug`atga o`zg artirish.
- **float([x])**-haqiqiy songa o`zgartirish. Agar argument ko`rsatilmagan bo`lsa, 0.0 qaytaradi.
- **int([object],[asosiy sanoq sistemasi])-** butun sonni berilgan sanoq sistemasidan o`nlik sanoq sistemasiga o`tkazish.
- **list([object])**-ro`yxat tuzadi.
- **memoryview(object)-** memoryview obyektini tuzadi.
- **object()**-hamma obyektlarga asos bo`lgan bosh obyektni qaytaradi.
- **range([start=0], stop,[step=1])-** step qadamli start dan stop gacha bo`lgan arifmetik progressiya.
- **set(object)**-to`plamni yaratadi.
- **slice([start=0], stop, [step=1])**-step qadamga ega startdan stopgachaga bo`lgan kesmaobekti.
- **tuple(obj)-** kortejga o`zgartiradi.

- **abs(x)-** absolyut raqamni (sonni modulini) qaytaradi.
- **all(ketmaketlik)-** agarda hamma elementlar haqiqiy bo`lsa (yoki ketmaketlik bo`sh bo`lsa) True ni qaytaradi.
- **any(ketmaketlik)-**agarda elementlardan hech bo`lmaganda bittasi haqiqiy bo`lsa True ni qaytaradi. Bo`sh ketmaketlik uchun False qaytaradi.
- **ascii(object)-** repr ga o`xshab obyekt ko`rinishiga mos qatorni ekranga xuddi shunday qaytaradi.
- **bin(x)-** butun sonni ikkilik sanoq sistemasiga o`tkazadi
- **chr(x)-** x ning Unicode ga mos belgini qaytaradi.
- **classmethod(x)-** sinf metodi ko`rsatgan funksiyani taqdim etadi.
- **compile(source, filename, mode, flags=0, don't_inherit=False)-** ketmaketlik eval yoki exec funksiyalari bilan bajariladigan dastur kodiga komplyatsiya qilinishi. Qator karetkani qaytaruvchi belgilar yoki nolga teng baytlarga ega bo`lmasligi kerak.
- **delattr(object, name)-** "name" nomidan atributni qaytaradi.
- **dir([object])-** obyekt nomlarining ro`yxati, agar obyekt ko`rsatilmagan bo`lsa, local maydondagi nomlar ro`yxati.
- **divmod(a,b) –** a ni b ga bo`lganda hosil bo`lgan bo`linmaning butun va qoldiq qismi.
- **enumerate(iterable, start=0)-** nomer va unga mos ketmaketlik a'zosidan tarkib topgan kortejni har bir o`tishda taqdim etuvchi iteratorni qaytaradi.
- **eval(expression, globals=None, locals=None)-** dastur kodi qatorini bajaradi.
- **filter(function, iterable)-** function yordamida rost qiymatni elementlarga qaytaruvchi iteratorni qaytaradi.
- **format(value [,format_spec])-** formatlash (qatorni formatlash).
- **getattr(object, name,[default])-** obyekt atributini yoki default.globals()-global nomlar lugatini chiqaradi.

- **hasattr(object, name)-** “name” nomidagi atribut obyektga ega ekanligini tekshiradi.
- **hash(x)-** ko`rsatilgan obyektning heshini qaytaradi.
- **help([object])-** dasturni yordam qismiga kiritilgan ma’lumotnoma tizimini chaqirish.
- **hex(x)-** butun sonni o`n oltilik sanoq sistemasiga o`tkazish.
- **id(object)-**obyekt manzilini qaytaradi .
- **input([promt])-** foydalanuvchi tomonidan kiritilgan qatorni qaytaradi. Promt-foydalanuvchiga yordam.
- **isinstance(object, ClassInfo)-**agarda obyekt classinfo yoki uning sinfosti ekzemplari bo`lsa rost qiymat qaytaradi. Agarda ekzemplar berilgan tipdagi obyekt bo`lmasa, funksiya yolg`on qiymat qaytaradi.

- **issubclass(sinf, ClassInfo)-**agarda sinf ClasssInfo sinfostisi bo`lsa rost qiymat qaytaradi.Sinf o`z-o`ziga sinfosti bo`ladi.
- **iter(x)-** iterator obyektini qaytaradi.
- **len(x)-**ko`rsatilgan obektni elementlar sonini qaytaradi.
- **locals()-**lokal nomlar lug`ati.
- **map(function, iterator)-**ketmaketlikning har bir elementiga function funksiyasini qo`llash orqali yaratiladigan iterator.
- **max(iter,[args…]*[, key])-**ketma-ketlikning maksimal elementi.
- **min(iter,[args…]*[, key])-**ketmaketlikning minimal elementi.
- **next(x)-**iteratorning keyingi elementini qaytaradi.
- **oct(x)-** butun sonni sakkizlik sanoq sistemasiga o`tkazadi.
- **open(file, mode='r', buffering=None, encoding=None, errors=None, newline=None, closefd=True)-** faylni ochadi va kerakli oqimni qaytaradi.
- **ord(x)-** belgi kodi.
- **pow(x, y[,r])-**(x**y)%r.
- **reversed(object)-**yoyilgan obyektning iteratori.
- **print([object,…],*,sep=" ", end='/n', file=sys.stdout)-** ma'lumotlarni ekranga chop etish.
- **round(X,[N])-** verguldan keyin N- belgilargacha to`g`rilash.
- **setattr(obekt, nom, qiymat)-** obyekt atributini belgilash.
- **sorted(iterable[, key][, reverse])-** tartiblangan ro`yxat.
- **staticmethod(function)-** funksiya uchun statistik metod.
- **sum(iter, start=0)-**ketmaketlik elementlarini yig`indisi.
- **type(object)-** obyekt tipini qaytaradi.
- **type(name, bases, dict)-** name sinfidagi yangi ekzemplarni qaytaradi.
- **vars([object])-** obyekt atributlarining ro`yxati. Jimlik holatida- local nomlar lug`ati.

Biz hozir yuqorida Python dasturida kiritilgan funksiyalar bilan tanishdik. Ammo lekin siz o'zingizning funksiyalaringizni ham yaratishingiz mumkin. Ushbu funksiyalar foydalanuvchi tomonidan belgilangan funksiyalar deb ataladi.

Funksiya koddagi bir blok hisoblanadi. U faqat chaqirilgandagina ishlaydi. Ya'ni qandaydir funksiya tuzilgan, ammo uni hali ishlashiga buyruq bermasak kodimiz ishga tushganda bu funksiya bajarilmaydi.

Funksiyaga ma'lumotlar uzatishimiz mumkin va bu ma'lumotlar funksiya uchun parametrlar hisoblanadi. Funksiya bizga ma'lumotlarni natija sifatida qaytarishi mumkin.

Funksiyalarni hosil qilish

Funksiyalar **def** kalit so'zi bilan hosil qilinadi. **def** so`zidan so`ng **funksiya nomi** va qavs ichida **formal parametrlar ro`yxati** ko`rsatiladi. Funksiya tanasini hosil qiluvchi instruksiyalar keyingi qatordan boshlab bo`sh joy(отступь) bilan yoziladi. Quyidagi kodimiz ishga tushsa, bizga hech qanday natija bermaydi. Chunki biz faqat funksiya hosil qilgan bo'lamiz:

```
def my_func():
    print("Funksiya ishga tushdi")
```

Funksiyani chaqirish

Avval aytganimizdek funksiya faqat chaqirilgandagina ishlaydi. Uni chaqirish uchun funksiyaning nomi qavslar bilan yozamiz. Yuqoridagi kodimiz natija berishi uchun o'sha funksiyani chaqiramiz va funksiya ishga tushadi:

```
def my_func():
    print("Funksiya ishga tushdi")

my_func()
```

```
Funksiya ishga tushdi
```

Argumentlar

Funksiyada ma'lumotlar argumentlar orqali uzatiladi. Argumentlar funksiya hosil qilayotganda funksiya nomidan so'ng qavslar ichiga kiritiladi. Argumentlar bir emas bir nechta bo'lishi mumkin. Bunday holatda ularni vergul bilan ajratib yoziladi.

Quyidagi misolimizda bizda ism degan argument bor. Funksiya hosil qilinganda argumentni qayerda qo'llash kerakligini ko'rsatamiz. Funksiyani chaqirayotganda esa o'sha argument o'rnida qanday qiymat bo'lishi kerakligini ko'rsatamiz:

```
def my_func(ism):
    print(ism + " Hamidov")

my_func("Mahmud")
my_func("Shahzod")
my_func("Odil")
```

```
Mahmud Hamidov
Shahzod Hamidov
Odil Hamidov
```

Funksiya tuzilayotganda nechta argument bilan tuzilsa, chaqirilayotganda ham shuncha argument bilan chaqirilishi kerak. Aks holda xatolik yuz beradi.

Masalan, quyidagi misolimizda ikkita – ism va familiya argumentli funksiya tuzamiz va uni shu ikkita argument orqali chaqiramiz:

```
def my_func(ism, familiya):
    print(ism + " " + familiya)

my_func("Abbosbek", "Ibragimov")
```

```
Abbosbek Ibragimov
```

*args

Bir argument orqali bir nechta qiymatlarda foydalanmoqchi bo'lsak, funksiya tuzilayotgan vaqtda argument nomi oldidan ***** belgisi qo'yiladi. Bu usul bilan ko'proq qiymatlar to'plamiga ega bo'lamiz va bir argumentni bir nechta qiymatlar bilan ishlatishimiz mumkin.

```
def mevalar(*meva):
    print(meva[0] + "," + meva[2])

mevalar("anjir", "gilos", "uzum")
```

```
anjir,uzum
```

Qiymat qaytarish

Funksiyalar vazifasiga ko'ra ikki turga bo'linadi. Bular qiymat qaytaradigan va qiymat qaytarmaydigan funksiyalar. Biz yuqorida hosil qiligan funksiyalarimiz bu qiymat qaytarmaydigan funksiyalar hisoblanadi. Endi esa qiymat qaytaruvchi funksiyalar hosil qilish bilan tanishamiz. Qiymat qaytaruvchi funksiyalar hosil qilish uchun **return** so'zidan foydalanamiz.Masalan,

istalgan sonning kvadratini chiqaruvchi funksiya tuzsak:

```
def kvadrat(x):
    return x*x

print(kvadrat(5))
```

```
25
```

Bunda yuqoridagi dasturga e'tibor bersangiz funksiya **return** kalit so'zi orqali x argumentning ikkinchi darajasini ya'ni kvadratini qaytarmoqda va bu shuning uchun ham biz yaratgan fuksiya qiymat qaytaruvchi funksiyaga misol bo'la oladi.

LAMBDA FUNKSIYA

Lambda funksiyasi kichik anonim funksiya hisoblanadi. Unda istalgancha argument qatnashishi mumkin va barchasi bir ifodada yoziladi. Hozir kiritilgan sonni 10 ga oshiradigan **lambda** funksiya hosil qilamiz:

```
x = lambda a: a + 10
print(x(2))
```

```
12
```

Endi ikki va uch argumetli lambda funksiyalarini tuzami. Avvalgisi ikki sonning o'zaro ko'paytmasini, keyingisi esa barcha sonlar yig'indisini topadi.

```
x = lambda a, b : a*b
print(x(5,6))

y = lambda a, b, c : a+b+c
print(y(7,9,5))
```

```
30
21
```

Nega lambda funksiya ishlatamiz ?

Lambda fuksiyalarni funksiya ichida boshqa bir anonim funksiya sifatida ishlatish qulay. Masalan, bir argumentli funksiya bor va uning argumenti noma'lum bir songa ko'payadi. Shu funksiyani lambda funksiya yordamida istalgan sonni ikkilantiradigan va uchlantiradiga funksiyaga aylantiramiz.

```
def myfunc(n):
    return lambda a: a*n

ikkilantir = myfunc(2)
uchlantir = myfunc(3)

print(ikkilantir(5))
print(uchlantir(5))
```

```
10
15
```

PYTHONDA MODULLARDAN FOYDALANISH

Modul – bu biz yozgan kodimizning fayl ko'rinishi. Bitta katta dasturimiz ko'pgina modullardan tashkil topishi mumkin. Pythonda modul hosil qilish uchun yozga kodimizning **.py** fayl kengaytmasi bilan saqlashimiz kerak bo'ladi.

Masalan, quyidagi salomlash funksiyasi yozilgan kodimizni **salom.py** nomi bilan saqlaymiz va shu nomli modul hosil bo'ladi:

```
def salomlashish (ism):
    print("Salom"+ ism)
```

Moduldan foydalanish

Endi biz katta dastur tuzishni boshlar ekanmiz, biz barcha kodni bir modulga yozib ishlatishimiz noqulay bo'ladi. Shuning uchun umumiy dasturni modullarga bo'lib ishlatganimiz ma'qul. Bu ancha qulay bo'ladi. Chunki biror moduldagi funksiya yoki ma'lumot kerak bo'lsa uni qayta-qayta yozib o'tirmasdan o'sha modulning o'zidan olib ishlatishimiz mumkin. Masalan, yuqorida funksiyatuzib uni **salom.py** moduliga saqlab qo'ydik. Endi biz yangi modul ochib unda **salom.py** modulidagi funksiyani ishlatamiz. Buning uchun **import** kalit so'zi bilan **salom.py** modulini chiqaramiz. So'ngra undagi **salomlash()** funksiyasini olib ishlatamiz.

E'tibor bering, bu yerda shunchaki funksiya nomini yozib ishlatayapmiz. Funksiyaning o'zi esa biz chaqirgan modulda tuzilgan:

```
import salom

salom.salomlash("Abbosbek")
```

```
Salom Abbosbek
```

Modulda o'zgaruvchilar

Modullar nafaqat funksiya, balki o'zgaruvchilarni ham o'z ichiga olishi mumkin. Shu sababli bir moduldagi ma'lumotdan boshqa modullarda ham foydalanish mumkin. Masalan, **avto** nomli dictionary o'zgaruvchisini **mashina.py** moduliga saqlaymiz:

```
avto = {
    "brend": "Audi",
    "model": "R8",
    "rang": "kumush",
    "yil" : 2018
}
```

Endi yangi modul ochamiz va unda mashinaning modelini ekranga chiqarishi buyuramiz:

```
import mashina

a = mashina.avto["model"]
print(a)
```

```
R8
```

Modulni nomlash

Biz murojaat qilgan modulning nomi uzunroq bo'lsa, uni kodimizda keyinchalik bu uchun nom bilan ishlatishimiz biroz noqulay bo'ladi. Ammo bizda uni kod ichida o'zimiz uchun qulay nom bilan ishlatish imkoniyati bor. Buning uchun modulni chaqirayotgan vaqtda uni pass kalit so'zi bilan o'zimizga qulay qilib nomlab olamiz. Natijada, kod ichida uni shu qulay nom bilan ishlatishimiz mumkin bo'ladi.

Masalan, avvalroq biz saqlagan **mashina.py** moduliga murojaat qilamiz va uni o'zimiz uchun m sifatida belgilaymiz. So'ngra uni shu nom bilan ishlatamiz:

```
import mashina  as m

a = m.avto["model"]
print(a)
```

```
R8
```

dir() funksiyasi

dir() maxsus funksiyasi istalgan modulga tegishli barcha funksiya yoki o'zgaruvchilar ro'yxatini chiqarib beradi. Xoh u maxsus modul bo'lsin, xoh o'zimiz tuzgan, barchasi uchun amal qiladi.

Masalan, Pythonda matamatik hisob-kitoblar uchun **math** moduli mavjud. Undagi barcha funksiyalar ro'yxati kerak bo'lsa, uni quyidagicha ekranga chiqaramiz:

```
import math

x = dir(math)
print(x)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copy
sign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm
1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd'
, 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldex
p', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi
', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', '
tanh', 'tau', 'trunc']
```

Keraklisini olish

Tasavvur qiling biz tuzgan biror modulda ko’p funksiya yoki o’zgaruvchilar bor. Bizga ulardan faqat bittasi kerak. Qolganlari shart emas. Bunday vaziyatda butun boshli modulni olmasdan, shunchaki undagi kerakli funksiyani olishimiz mumkin.

Tushunish uchun bir modul tuzamiz, uning nomi **suhbat.py** bo’lsin. Unda ikkita funksiya tuzamiz. Bular **salom()** va **xayr()** funksiyalari bo’ladi:

```
def salom(ism)
    print("Salom "+ism)

def xayr(ism):
    print("Xayr "+ism)
```

Endi yangi modul ochamizda, **suhbat.py** modulidagi faqat **xayr()** funksiyasini olib ishlatamiz:

```
from suhbat import xayr

xayr ("Abbosbek")
```

```
Xayr Abbosbek
```

E’tirbor bering: ilgari biror modulni chaqirganimizdan keyin undagi funksiyalar oldidan shu modul nomini qo’yib, so’ng nuqta (.) va kerakli funksiyani yozar edik. Agar o’sha moduldan ayna bir funksiyaning o’zini chaqirsak, shunchaki funksiya nomi yozilib ishlatiladi.

Masalan: **suhbat.xary(“**Madaminbek**”)** emas, shunchaki **xayr(“Madaminbek”)** tarzida yoziladi.

PYTHONDA MAXSUS MODULAR

Pythonda ayrim narsalarga mo'ljallangan tayyor, maxsus modulla bor. Ularning har birining o'z vazifasi bor va biz o'zimizga kerak o'rinda ularga murojaat qilib ishlatamiz. Bunday modular ro'yxati, ularning vazifalari va ularni qo'llash haqida yana qo'shimcha adabiyotlardan olishingiz mumkin.

Standart kutubxonalar

Python tili standart kutubxonasining modullarini shartli ravishda mavzular bo`yicha quyidagiguruhlarga ajratish mumkin:

- Bajarish davri servislari. Modular: sys, atexit, copy, traceback, math, cmath, random, time, calendar, datetime, sets, array, struct, intertools, locale, gettext.
- Siklni qayta ishlashni qo`llab-quvvatlovchi. Modullar: pdb, hotshot, profile, unittest, pydoc. Paketlar: docutils, distutils.
- OS (fayllar, protseslar) bilan ishlash. Modullar: os, os.path, getopt, glob, popen2, shutil, select, signal, stat, tempfile.
- Matnlarni qayta ishlovchi. Modullar: string, re, StringIO, codecs, difflib, mmap, sgmllib, htmllib, htmlentitydefs. Paket: xml.
- Ko`p oqimli hisoblashlar. Modullar: threading, thread, Queque.
- Ma'lumotlarni saqlash. Arxivlash. Modullar: pickle, shelve, anydbm, gdbm, gzip, zlib, zipfile, bz2, csv, tarfile.
- Platformaga tobe modullar. UNIX uchun: commands, pwd, grp, fcntl, resource, termios, readline, rlcompleter. Windows uchun: msvcrt, _winreg, winsound.
- Tarmoqni qo`llab-quvvatlash. Internet protokollari. Modullar: cgi, Cookie, urllib, urlparse, htttplib, smtplib, poplib, telnetlib, socket, asyncore. Serverlarga misollar: SocketServer, BaseHTTPServer, xmlrpclib, asynchat.
- Internetni qo`llab-quvvatlash. Ma'lumotlar formatlari. Modullar: quopri, uu, base64, binhex, binascii, rfc822, mimetools, MimeWriter, multifile, mailbox. Paket: email.
- Python uchun. Modullar: parser, symbol, token, keyword, incpect, tokenize, pyclbr, py_compile, compileall, dis, compiler.
- Grafik interfeys. Modul: Tkinter.

Ko`pincha modullar o`zida bir yoki bir nechta sinflarni saqlaydilar. Bu sinflar yordamida kerakli tipdagi obyekt yaratiladi, lekin gap moduldagi nomlar haqida emas, aksincha shu obtekt atributi haqida boradi. Bir nechta modullar faqat erkin obyetlar ustida ishlash uchun umumiy bo`lgan funksiyalardan iborat bo`ladilar.

Platform moduli

Hozir biz **platform** modulini ishlatamiz. Bu modul bilan biz qaysi operatsion sistemada ishlayotganimizni bilish mumkin. Masalan, deylik biz **Windows** operatsion tizimidagi kompyuterni ishlatmoqdamiz. Demak kodni ishga tushirsak, ekranga **Windows** yozuvi chiqib keladi.

```
import platform

x = platform.system()
print(x)
```

```
Windows
```

Pickle moduli

Pythonning **pickle** moduli yordamida har qanday obyektni faylga saqlash va keyinchalik fayldan o'qib olish mumkin. Bunday imkoniyat ob'yektlarni uzoq muddat saqlashda qo'l keladi.

```
import pickle
# obyektni saqlash fayli

shoplistfile = 'shoplist.data'

# xaridlar ro'yxati

shoplist = ['olma', 'mango', 'sabzi']

# faylga yozish

f = open(shoplistfile, 'wb')

pickle.dump(shoplist, f) # obyektni faylga yozamiz

f.close()

del shoplist # shoplist o'zgaruvchisini o'chirib tashlaymiz

# fayldan o'qish

f = open(shoplistfile, 'rb')

storedlist = pickle.load(f) # ob'yektni fayldan yuklab olish

print(storedlist)
```

Natija:

```
['olma', 'mango', 'sabzi']
```

Bu misolda obyektni faylga yozish uchun birinchi galda faylni binar yozish **(“wb”)** rejimida ochilyapti, so’ng pickle modulining dump funksiyasi chaqirilyapti. Bu jarayon “konservatsiya” (“pickling”) deyiladi. Shundan so’ng obyektni fayldan o’qib olish uchun pickle modulining **load** funksiyasidan foydalanilyapti.

Sys moduli

Sys moduli Python interpretatorida dasturni bajaruvchi muhitdir. Quyida bu modulni eng ko`p qo`llaniladigan obyektlari keltilgan:

- **Exit([c])-** dasturdan chiqish. Tugatishning raqamli kodini yuborish mumkin: agarda dasturni tugatish muvafaqqiyatli amalga oshsa 0 ni yuboradi, aksincha bo`lsa ya’ni xatolik yuz bersa boshqa raqamlarni yuboradi.
- **Argv-** buyruqlar qatori argumentlari ro`yxati. Oddiy holatda sys.argv[0] buyruqlar qatoriga ishga tushirilgan dastur nomini va boshqa parametrlar yuboriladi.
- **Platform-** interpretator ishlaydigan platforma.
- **Stdin, stdout, stderr-** standart kiritish, chiqarish, xalolarni chiqarish. Ochiq faylli obyektlar.
- **Version-** interpretator versiyasi.
- **Serecursionlimit(limit)-** rekursiv chaqirishlarni maksimal kiritish darajasini o`rnatadi.
- **Exc_info()-**kiritish-chiqarish istisnosi haqida ma’lumot.

Copy moduli

Bu modul obyektlarni nusxalashga mo`ljallangan funksiyalarga ega. Boshida Pyhtonda sal sarosimaga solish uchun “paradoks” ni ko`rib chiqish tavsiya etiladi.

```
list1 = [0, 0, 0]
list = [list1] * 3
print(list)
list[0][1] = 1
print (list)
```

Va biz kutmagan natija paydo bo`ladi:

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 1, 0], [0, 1, 0], [0, 1, 0]]
```

Gap shundaki bu yerda lst ro`yxati shu ro`yxatnining izohiga ega. Agarda rostdan ham ro`yxatni ko`paytirmoqchi bo`lsak, copy modulidagi **copy()** funksiyasini qo`llash kerak.

```
from copy import copy
lst1 = [0, 0, 0]
lst = [copy(lst1) for i in range(3)]
print (lst)
lst[0][1] = 1
print (lst)
```

Endi kutilgan natija paydo bo`ladi:

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 1, 0], [0, 0, 0], [0, 0, 0]]
```

Copy modulida yuqori aniqalikda nusxalash uchun **deepcopy()** funksiyasi bor bu funksiya yordamida obektlar butun imkoniyati bilan rekursiv nusxalanadi.

Random moduli

Bu modul har xil taqsimotlar uchun tasodifiy raqamlarni generatsiya qiladi. Eng ko`p qo`llaniladigan funksiyalari:

- **Random()**-[0.0, 1.0) yarim ochiq diapozondagi tasodifiy sonlarni generatsiya qiladi.
- **Choice(s)**- s ketma- ketlikdan tasodifiy elementni tanlab oladi.
- **Shuffle(s)**- s o`zgaruvchan ketma-ketlik elementlarini joyiga joylashtiradi.
- **Randrange([start], stop, [step])**- renge(start, stop, step) diapozondagi tasodifiy butun raqamni chiqaradi. Choice(range(start, stop, step)) ga analogik holatda.
- **Normalvariate(mu, sigma)**- normal holatda taqsimlangan ketma-ketlikdan raqamni chiqaradi. Bu yerda mu- o`rtacha, sigma-o`rta kvadratli (sigma>0) sonlar.

Boshqa funksiyalar va uning parametrlarini hujjatlashdan aniqlab olish mumkin. Modulda qandaydir holatga tasodifiy raqamlar generatorini joylashtirishga imkon beruvchi **seed(n)** funksiyasi ham mavjud. Masalan: agarda bitta tasodifiy raqamlar ketma-ketligidan ko`p marta foydalanishga ehtiyoj sezilsa.

Os moduli

Os moduli-har xil operatsion sistemalarning o`ziga xos xususiyatlari bilan ishlovchi kategoriyadagi asosiy modul hisoblanadi. Bu modul funksiyalari ko`plab operatsion sistemalarda ishlaydilar. Kataloglarni bo`luvchi os moduli va u bilan bog`liq bo`lgan ifodalar konstanta ko`rinishida berilgan.

Konstanta	**Vazifasi**
Os.curdir	Joriy katalog
Os.pardir	Bosh katalog
Os.sep	Yo`lning elementlarini taqsimlovchi
Os.altsep	Boshqa yo`lning elementlarini taqsimlovchi
Os.pathsep	Yo`llar ro`yxatidagi yo`llarni taqsimlovchi
Os.defpath	Yashirin yo`llar ro`yxati
Os.linesep	Satrni yakunlovchi belgi

Kataloglarni bo`luvchi os moduli ifodalari konstanta ko`rinishida

Pythondagi dastur operatsion tizimda alohida jarayon ko`rinishida ishlaydi. Os modulining funksiyalari protsesda, muhitda bajariladigan turli xildagi ahamiyatga ega bo`lgan kirishlarga ruxsat etadilar. Os modulining eng muhim ruxsat etuvchi obyektlaridan biri deb environ o`rab oluvchi muhiti o`zgaruvchilarning lug`ati hisoblanadi. Masalan o`rab oluvchi muhit o`zgaruvchilar yordamida web server CGI-ssenariyga bir qancha parametrlarni o`tkazadi.

Quyidagi misolda PATH o`rab oluvchi muhiti o`zgaruvchini olish mumkin:

```
import os
PATH=os.environ['PATH']
```

Funksiyalarning katta qismi fayllar va kataloglar bilan ishlashga mo`ljallangan. Quyida **UNIX** va **Windows** OT lar uchun ruxsat etilgan funksiyalar taqdim etilgan:

- **Access(path, flags)-** path nomli fayl yoki catalog ruxsat etish(доступь) ni tekshiradi. Buyurma qilishga rucsatning tartibi flags raqami bilan belgilanadi. U esa yaratilgan kombinatsiyalar os.F_OK (fayl mavjud), os.R_OK (fayldan o`qish mumkin), os.W_OK (faylga yozish mumkin) va os.X_OK (fayllarni bajarishni, katalogni ko`rib chiqish mumkin) bayroqlari bilan belgilash mumkin.
- **Chdir(path)-** path ni joriy ishchi katalog qiladi.
- **Getcwd()-** joriy ishchi catalog.
- **Chmod(path, mode)-** mode ga path bo`lgan ruxsat etish rejimini belgilaydi. Ruxsat etish tartibi bayroqlarni kombinatsiya qilib belgilashi mumkin. Bu ishda chmod() harakatda bo`lgan tartibni to`ldirmaydi, uni yangidan belgilamaydi, uni yangidan belgilaydi.
- **Listdir(dir)-** dir katalogidagi fayllar ro`yxatini qaytaradi. Ro`yxatga maxsus belgilar “.” va “..” kirmaydi.
- **Mkdir(path [, mode])-** path katalogini tuzadi. Jimlik holatida mode tartibi 0777 ga teng bo`ladi, bu degani S_IRWXU|S_IRWXG|S_IRWXO agarda stat moduli konstantalari bilan foydalansak.

```
import os
os.mkdir('C:\Users\Guljakhon\Desktop\Новая папка\katalog\dir2')
#ko`rsatilgan manzilda dir2 nomli yangi katalog yaratadi.
```

```
import os
os.mkdir('./dir2')
#joriy manzilda dir2 nomli yangi catalog yaratadi.
```

- **Makedirs(path [,mode])-** hamma kataloglarni yaratuvchi, agarda ular mavjud bo`lmasalar mkdir() analogi oxirgi katalog mavjud bo`lgandan so`ng mustasnoni ishga tushiradi.
- **Remove(path), unlink(path)-** path katalogini yo`qotadi. Kataloglarni yo`qotish uchun rmdir() va removedirs() dan foydalanadi.
- **Rmdir(path)-** path nomli bo`sh katalogni yo`qotadi.
- **Removedirs(path)-** birinchi bo`sh bo`lgan kataloggacha pathni yo`q qiladi. Agarda yo`lda eng oxirgi kiritilgan katalog osti bo`sh bo`lmasa OSError mustasnosini ishga tushiradi.
- **Rename(src, dst)-** src fayli yoki katallogini dst deb qayta nomlaydi.
- **Renames(src, dst)-** rename() analogi dst yo`li uchun kerakli kataloglarni yaratadi va src yo`lining bo`sh kataloglarini yo`qotadi.
- **Stat(path)-** path haqidagi malumotni o`nta elementlik kortej shaklida qaytaradi. Kortej elementlariga kirish uchun stat moduli konstantalaridan foydalanish mumkin. Masalan stat.ST_MTIME (faylning oxirgi modifikatsiyasi vaqti).
- **Utime(path, times)-** oxirgi modifikatsiya (mtime) va faylga kirishga ruxsat(atime) larini belgilaydi. Boshqa holatlarda times ikki elementli kortej (atime, mtime) sifatida ko`rib chiqiladi. Qaysidir faylni atime va mtime ni olish uchun stat() va stat modulining konstantalarini barobar ishga tushirib olish mumkin.
- Os moduli protsesslar bilan ishlash uchun quyidagi funksiyalarni taqdim etadi (ular ham UNIX hamda windowsda ishlaydilar).
- **System(cmd)-** alohida oynada cmd buyruqlar satrini bajaradi. U C tilining system kutubxonasi chqirig`iga analogik bo`ladi. Qaytarilgan qiymat foydalanadigan platformadan tobe bo`ladi.
- **Times()-** beshta elementdan iborat bo`lgan kortejni qaytaradi. U ish jarayoni vaqtini lahzalarda ko`rsatadi, qo`shimcha protsesslar vaqtini, qo`shimcha protsesslarning axborot tizimlari vaqtini, va o`tgan zamonda qotib qolgan vaqtni ko`rsatadi (masalan tizim ishga tushgan paytdan).
- **Getloadavg()-** coo, uchta qiymatlik kortejni qaytaradi.

SANA VA VAQT (DATETIME MODULI)

Sana yoki vaqt bilan ishlash uchun **datetime** modulini ishga solamiz. Bu modul bilan yildan tortib millisekundlargacha ma’lumot olish mumkin.

Masalan, quyidagi kodni ishga tushirsak, joriy vaqtni ko’rsatadi. Bunda to’liq holda, ya’ni yil, oy, kun, soat, minut, sekund, millisekundlar ko’rinadi:

```
import datetime as dt

x = dt.datetime.now()
print(x)
```

```
2020-11-14 14:00:56.561937
```

O’lchov turlari

datetime modulida juda ko’p funksiyalar bor. Ular orqali vaqtni nafaqat to’liq holda, balki faqatgina bizga kerakli holda ham aniqlashimiz mumkin. Masalan, bizga faqat joriy yil yoki bugungi hafta kuni kerak. Bularning alohida funksiyalari bor. Quyidagi kodimizda dastlab faqat joriy yilni so’ngra bugungi hafta kunini ekranga chiqaramiz:

```
import datetime as dt

x = dt.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

```
2020
Saturday
```

Vaqtni belgilash

Biz o’zimiz sana yoki vaqtni belgilashimiz mumkin. Buning uchun kerakli sonlarni ko’rsatsak kifoya. Masalan, hozir sanani 28-noyabr 2020-yil deb belgilaymiz:

```
import datetime as dt

x = dt.datetime(2020, 11, 28)

print(x)
```

```
2020-11-28 00:00:00
```

strtime() funksiyasi

strtime() funksiyasi vaqtga oid turli ma’lumotlarni turli formatlarda bizga qaytaradi. Ushbu fuksiyani maxsus format kodlari bilan qo’llash kerak. Masalan, joriy oyning nomini ko’rsatish kodi **%B**. Demak hozirgi oy nomini ekranga chiqarish quyidagicha bo’ladi.

```
import datetime as dtx

= dt.datetime.now()

print(x.strftime("%B"))
```

```
November
```

Quyida maxsus format kodlariga misollar keltiramiz. Ularni xuddi yuqoridagi kod singari sinab ko’rsangiz tushunish yanada osonroq bo’ladi:

- **%a** – hafta kuni (qisqa)
- **%A** – hafta kuni (to’liq)
- **%w** – hafta kuni (raqam shaklida)
- **%d** – oyning sanasi
- **%b** – oy nomi (qisqa)
- **%B** – oy nomi (to’liq)
- **%m** – oy (raqam ko’rinishida)
- **%y** – yil (qisqa)
- **%Y** – yil (to’liq)
- **%H** – soat (00-23)
- **%I** – soat (00-12)
- **%p** – kun vaqti (AM/PM)
- **%M** – minut (00-59)
- **%S** – sekund (00-59)
- **%j** – yildagi kun raqami (001-366)
- **%U** – yildagi hafta raqami, Yakshanba birinchi kun sifatida (00-53)
- **%W** – yildagi hafta raqami, Dushanba birinchi kun sifatida (00-53)
- **%c** – mahalliy sana va vaqt
- **%x** – mahalliy sana
- **%X** – mahalliy vaqt

PYTHONDA MATEMATIKA

Amallar bajarilish ketma-ketligi

2 + 3 * 4 ifodada qaysi amal birinchi bajariladi: qo'shishmi yoki ko'paytirish?

Matematika fanida ko'paytirish birinchi bajarilishi ko'rsatilgan. Demak, ko'paytirishoperatori qo'shish operatoriga qaraganda katta prioritetga(muhimlik darajasiga) ega. Quyidagi jadvalda Python operatorlari prioriteti ko'rsatilgan. Bunda yuqoridan pastga qarab Python operatorlari prioriteti oshib boradi. Bu shuni anglatadiki, ixtiyoriy ifodada Python oldin eng quyidagi operatorlarni hisoblaydi va keyin esa yuqoridagilarini. Amaliyotda esa amallarni qavslar bilan aniq ajratish tavsiya etiladi. Bu dastur kodini oson o'qishga yordam beradi.

Operator	Izoh
Lambda	lambda ifoda
Or	Mantiqiy 'yoki'
And	Mantiqiy 'va'
Not x	Mantiqiy 'emas'
in, not in	Tegishlilikni tekshirish
is, is not	Bir xillikni tekshirish
<, <=, >, >=, !=, ==	Taqqoslash
|	'yoki' bit operatori
^	'shartlik yoki' bit operatori
&	'va' bit operatori
<<, >>	Surilishlar
+, –	Qo'shish va ayirish
***, /, //, %**	Ko'paytirish, bo'lish, qoldiqsiz bo'lish va qoldiqlik bo'lish
+x, -x	Musbat va manfiy
~x	'emas' bit operatori
******	Darajaga ko'tarish
x.attribute	Atributga link
x[index]	Indeks bo'yicha murojat

x[index1:index2]	Kesib olish
f(argumentlar …)	Funksiyani chaqirish
(ifoda, …)	Kortej (Связка или кортеж)
[ifoda, …]	Ro'yxat (Список)
{kalit:qiymat, …}	Lug'at (Словарь)

Bu jadvalda bir xil prioritetga ega bo'lgan operatorlar bir qatorda joylashgan. Misol uchun '+' va '-'.

Hisoblash tartibini o'zgartirish

Ifodalarni o'qishni osonlashtirish uchun qavslarni ishlatish mumkin. Misol uchun, **2 + (3 * 4)** ni tushunish oson operatorlar prioriteni bilish lozim bo'lgan **2 + 3 * 4** ifodadan ko'ra. Qavslarni o'ylab ishlatish kerak. Ortiqcha qavslarni ishlatishdan saqlaning. Misol uchun: **(2 + (3 * 4)).**
Qavslarni ishlatishni ya'na bir afzalligi hisoblash tartibini o'zgartirish imkonini beradi. Misol uchun, qo'shish amalini ko'paytirish amalidan birinchi bajarish kerak bo'lsa, quyidagicha yozish mumkin: **(2 + 3) * 4.**

Pythonda matematik hisob-kitoblar uchun o'zining maxsus funksiyalariga ega. Bu funksiyalar tayyor holda bo'lib, kerakli natijalarni tezda chiqarib beradi.

min() funksiyasi berilgan sonlar ichida eng kichigini, **max()** funksiyasi esa eng kattasini aniqlaydi.

```
x = min(3, 8, 11)
y = max(3, 8, 11)

print(x)
print(y)
```

```
3
11
```

abs() funksiyasi sonning absolyut qiymatini aniqlaydi.

pow(x,y) funksiyasi **x** ni **y** darajaga ko'taradi.

```
x = abs(-5)
y = pow(5, 3)

print(x)
print(y)
```

```
5
25
```

Math va cmath moduli

Yuqoridagi ko'rganlarimiz Pythondagi ichki funksiyalar edi. Ularni to'g'rida-to'g'ri ishlatish mumkin. Ammo boshqa bir guruh matematik funksiyalar borki ular **math** moduliga mansub. Shuning uchun ularni ishlatishdan avval math moduliga murojaat qilamiz. Masalan, biror sonning kvadrat ildizini hisoblamoqchimiz. Buning uchun maxsus **sqrt()** funksiyasi mavjud. Uning ishlatilishi uchun **math** moduliga murojaat qilamiz:

```
import math

x = math.sqrt(64)
print(x)
```

```
8.0
```

Math va cmath modullarida haqiqiy va compleksli argumentlar uchun matematik funksiyalar to`plangan. Bu C tilida foydalaniladigan funksiyalar. Quyida math modulining funksiyalari keltirilgan. Qayerda z harfi bilan argumentga belgilash kiritilgan bo`lsa, u cmath modulidagi analogik funksiya ham shunday belgilanishini bildiradi.

- **acos(z)-**arkkosinus z.
- **asin(z)-** arksinus z.
- **atan(z)-** arktangens z.
- **atan2(y, x)-** atan(y/x).
- **ceil(x)-** x ga teng yoki katta eng kichik butun son.
- **cos(z)-** kosinus z.
- **cosh(x)-** giperbolik x kosinusi.
- **e-** e konstantasi.
- **exp(z)-** eksponenta (bu degani e**z)
- **fabs(x)-**x absolute raqami.
- **floor(x)-** xga teng yoki kichik eng katta butun son
- **fmod(x,y)-** x ni y ga bo`lgandagi qoldiq qism.
- **frexp(x)-** mantisa va tartibni (m, i) juftligi kabi qaytaradi, m- o`zgaruvchan nuqtali son, i esa- x=m*2**i ga teng butun son bo`ladi. Agarda 0-(0,0) qaytarsa boshqa paytda 0.5<=abs(m)<1.0 bo`ladi.
- **factorial(x)-** x ning faktoriali. N!=1*2*3*…*n
- **hypot(x,y)-** sqrt(x*x+y*y)
- **ldexp(m,i)-** m*(2**i).
- **log(z)-** natural logarifm z.
- **log10(z)-** o`nlik logarifm z.

- **log2(z)**-logarifm ikki asosga ko`ra z.
- **modf(x)-** (y,q) juftlikda x ning butun va kasr qismini qaytaradi.
- **pi-**pi konstantasi.
- **pow(x,y)-** x**y.
- **sin(z)-** z ning sinusi.
- **sinh(z)-** z ning giperbolik sinusi.
- **sqrt(z)-** z ning kvadrat ildizi.
- **tan(z)-** z ning tangensi.
- **tanh(z)-** z ning giperbolik tangensi.
- **trunc(x)-** x haqiqiy sonning butun qismini qaytaradi.
- **degrees(x)-**x ni radiandan gradusga o`tkazish.
- **radians(x)-** x ni gradusdan radianga o`tkazish.

math.ceil() funksiyasi eng yaqin yuqori butun songacha yaxlitlaydi. **math.floor()** funksiyasi esa eng yaqin pastki butun songacha yaxlitlaydi. Quyidagi misolimizda birinchi funksiyaning natijasi 2, keyingisi esa 1 bo'ladi:

```
import math

x = math.ceil(1.4) y
= math.floor(1.4)

print(x)
print(y)
```

```
2
1
```

Matematikadagi **PI** sonining qiymati pythondagi konstantalar ro'yxatida bor. Undan bemalol foydalanish mumkin:

```
import math

x = math.pi
print(x)
```

```
3.141592653589793
```

Bu darsimizda Pythonda matematika bo'limi bilan tanishdik. **math** va **cmath** modulining funksiyalari juda ko'p, ularning bacrhasiga doir misollarni ko'rib chiqa olmaymiz. Yana ham ko'proq funksiyalar haqida bilish uchun qo'shimcha adabiyotlarga murojaat qilishni maslahatberamiz.

PYTHONDA SANOQ SISTEMASINING ISHLATILISHI

Maktab kursidagi informatika faninidan bizga ma'lumki, sonlar nafaqat o`nlik sanoq sistemasida balki boshqa sanoq sistemalarida ham bo`lishi mumkin. Masalan: kompyuter ikkilik sanoq sistemasidan foydalanadi ya'ni 19-soni ikkilik sanoq sistemasida (kompyuterda) 10011 ko`rinishida ifodalanadi. Bundan tashqari sonlarni bir sanoq sistemasidan ikkinchi sanoq sistemasiga o`tkazish kerak. Python bu uchun bir qancha funksiyalarni taqdim etadi:

- **int([object],[sanoq sistemasi asosi])-** butun sonni berilgan sanoq sistemasidan o`nliksanoq sistemasiga o`tkazadi.
- **bin(x)-** butun sonni ikkilik sanoq sistemasiga o`tkazadi
- **hex(x)-** butun sonni o`n oltilik sanoq sistemasiga o`tkazadi
- **oct(x)-** butun sonni sakkizlik sanoq sistemasiga o`tkazadi.

```
>>> bin(19)
'0b10011'
>>> oct(19)
'0o23'
>>> hex(19)
'0x13'
>>> int('10011',2)
19
>>> int('0b10011',2)
19
>>> a=int('19')# satrni songa o`tkazadi
>>> b=int(19.5)# haqiqiy sonni butun qismini qaytaradi
>>> print(a,b)
19 19
```