Libraries

- Umuman olganda, **Library** - bu siz yoki boshqalar tomonidan yozilgan code, siz o'z dasturingizda keyinchalik foydalanishingiz mumkin.
- **Python** sizga "modullar" orqali funktsiyalar yoki xususiyatlarni boshqalar bilan almashish imkonini beradi.

Random

`random` - bu o'z codingizga import qilishingiz mumkin bo'lgan Python bilan birga keladigan library.
Xo'sh, modulni o'z dasturingizga qanday yuklaysiz? Dasturingizda `import` so'zidan foydalanishingiz mumkin.
Random moduli ichida `random.choice(seq)` deb nomlangan funksiya mavjud.
Random siz import qilayotgan modul. Ushbu modul ichida `choice` funktsiyasi mavjud.
Bu funktsiya o'z ichiga ketma-ketlikni yoki listni oladi.

```
import random

coin = random.choice(["heads", "tails"])
print(coin)
```

Buyerda biz `choice` funksiyasiga list berdik. Ikkita elementni topshirganingiz uchun, Python 50% ehtimolligi bilan shulardan bitasini qaytaradi.

Biz kodimizni yaxshilashimiz mumkin. `from` sozi orqalik biz aynan qaysi funksiya import qilinishini korsatishimiz mumkun. Oldingi usulda biz butun random modulini ozimizga import qilgan edik. Ammo, agar biz modulning faqat kichik qismini import qilmoqchi bo'lsak-chi? Kodingizni quyidagicha o'zgartiring:

```
from random import choice

coin = choice(["heads", "tails"])
print(coin)
```

Etibor bering biz faqat `random` modulidan faqat `choice` funsiyasini import qildik. Endi biz `random.choice` codinig ishlatishimiz shart emas. Biz faqat `choice` ishlatsak bas. `choice` bizning dasturimizga togri dan togri yuklangan. Bu tizim resurslarini tejaydi va bizning kodimizni tezroq ishlashiga yordam beradi!

Endi quyidagi code ga etibor bering `random.randint(a,b)`. Bu funksiya a va b orsadiga tahminiy bita son qaytaradi. Codingizni quyidagi tarzda o'zgartiring

```
import Random

number = random.randint(1, 10)
print(number)
```

Etibor bering codimiz 1 dan 10 gacha bolgan tahminiy bita son qaytardi.

Bizga berilgan royxatni elementlarni aralashtirish uchun biz `random.shuffle(x)` orqali foydalansak boladi.

```
import random

cards = ["jack", "queen", "king"]
random.shuffle(cards)
for card in cards:
    print(card)
```

`random.shuffle` kartalarni joyida aralashtirib yuboradi. Boshqa funktsiyalardan farqli o'laroq, u qiymatni qaytmaydi. Buning o'rniga, u kartalar ro'yxatini oladi va ularni shu ro'yxat ichida aralashtiradi. Kod ishlashini ko'rish uchun kodingizni bir necha marta ishga tushiring.

Endi bizda tasodifiy ma'lumotni yaratish uchun yuqoridagi uchta usul mavjud.

Python-ning hujjatlarida **random** haqida ko'proq ma'lumot olishingiz mumkin.

Statistics

Python tayyor **statistics** library si bilan birga keladi. Ushbu moduldan qanday foydalanishimiz mumkin?

```
import statistics

print(statistics.mean([100, 90]))
```

Etibor bering biz boshqa `statistics` library sini import qildik. `mean` funktsiya qiymatlar ro'yxatini oladi. Bu qiymatlarning o'rtacha qiymatini print etadi.

Python-ning hujjatlarida **statistics** haqida ko'proq ma'lumot olishingiz mumkin.

Command-line Arguments

Hozirgacha biz yaratgan dastur doirasida barcha qiymatlarni dastur ichida taqdim etib kelmoqdamiz. Agar biz command line dan ma'lumot olishni istasak nima bo'ladi? Misol uchun, terminalda python average.py yozish o'rniga, Agar biz python average.py 100 90 ni yozishni xohlasak nima bo'ladi? va 100 dan 90 gacha bo'lgan o'rtacha ko'rsatkichni olish mumkinmi?

`sys` - bu command line orqali argumentlarni qabul qilish imkonini beruvchi modul.

`argv` - bu `sys` modulidagi funksiya bo'lib, foydalanuvchi command line da nima yozganligi haqida bilib olish imkonini beradi. Quyidagi codda `sys.argv` qanday ishlatilganiga e'tibor bering.

```
import sys

print("hello, my name is", sys.argv[1])
```

E'tibor bering, dastur foydalanuvchi command line da nima yozganini ko'rib chiqadi. Hozirda terminal ga python name.py David yozsangiz, “salom mening ismim David” habarini korasiz.

E'tibor bering, `sys.argv[1]` Devid saqlanadigan joy. Nima sababdan? Oldingi darslarda listlar 0-elementdan boshlanishini eslab qolishingiz mumkin. Sizningcha, `sys.argv[0]` da hozir nima saqlanmoqda? Agar siz name.py-ni taxmin qilgan bo'lsangiz, to'g'ri bo'lardingiz! Bizning dasturimizda kichik muammo bor. Agar foydalanuvchi terminal da ismni yozmasa nima bo'ladi?

O'zingiz sinab ko'ring. Terminal oynasiga python name.py yozing. `list index out of range` kompilyator tomonidan taqdim etiladi. Buning sababi shundaki, `sys.argv[1]` da hech narsa yo'q, chunki hech narsa yozilmagan! Dasturimizni ushbu turdagi xatolardan qanday himoya qilishimiz mumkin:

```
import sys

try:
    print("hello, my name is", sys.argv[1])
except IndexError:
    print("Kam Argument berildi")
```

E'tibor bering, agar foydalanuvchi nom yozishni unutib qo'ysa, dasturni qanday ishlashi haqida foydali maslahat so'raladi. Biroq, foydalanuvchi to'g'ri qiymatlarni kiritishini ta'minlash uchun ko'proq himoyalangan bo'lishimiz mumkinmi?

Bizning dasturimizni quyidagicha yaxshilash mumkin:

```
import sys

if len(sys.argv) < 2:
    print("Kam Argument berildi")
elif len(sys.argv) > 2:
    print("Kop Argument berildi")
else:
    print("hello, my name is", sys.argv[1])
```

E'tibor bering, agar siz kodingizni sinab ko'rsangiz, foydalanuvchiga yanada aniqroq maslahatlar berib, ushbu istisnolar qanday hal qilinishini ko'rasiz. Agar foydalanuvchi juda ko'p yoki juda oz argumentlarni yozsa ham, foydalanuvchiga muammoni qanday hal qilish bo'yicha aniq ko'rsatmalar beriladi.

Hozir bizning kodimiz mantiqan to'g'ri. Biroq, xatolarimizni tekshirishni kodimizning qolgan qismidan alohida saqlashda juda yaxshi foydasi bor. Xatolar bilan ishlashni qanday ajratishimiz mumkin? Kodingizni quyidagicha o'zgartiring:

```
import sys

if len(sys.argv) < 2:
    sys.exit("Kam Argument berildi")
elif len(sys.argv) > 2:
    sys.exit("Kop Argument berildi")

print("hello, my name is", sys.argv[1])
```

Agar foydalanuvchi tomonidan xatolik yuz bergan bo'lsa, dasturdan chiqishga imkon beruvchi `exit` deb funksiyasidan qanday foydalanayotganimizga e'tibor bering. Dastur hech qachon oxirgi kod satrini bajarmasligiga va xatolikka yo'l qo'ymasligiga amin

bo'lishimiz mumkin. Shuning uchun, `sys.argv` foydalanuvchilarga terminaldan ma'lumotlarni kiritish usulini taqdim etadi. `sys.exit` xatolik yuzaga kelsa, dasturdan chiqishi mumkin bo'lgan vositani taqdim etadi.

Siz ko'proq ma'lumotni Python-ning **sys** hujjatlarida bilib olishingiz mumkin.

Slice

`slice` - bu bizga ro'yxatni olish va kompilyatorga ro'yxatning boshi va oxirini ko'rib chiqishni istagan joyni aytish imkonini beruvchi buyruq. Masalan, kodingizni quyidagicha o'zgartiring:

```
import sys

if len(sys.argv) < 2:
    sys.exit("Too few arguments")

for arg in sys.argv:
    print("hello, my name is", arg)
```

E'tibor bering, agar siz terminal oynasiga python name.py Devid Karter Rongxin yozsangiz, kompilyator nafaqat nomlarning mo'ljallangan chiqishini chiqaradi, balki salom, mening ismim name.py ni ham. Qanday qilib biz kompilyator name.py saqlanayotgan ro'yxatning birinchi elementiga e'tibor bermasligiga ishonch hosil qilishimiz mumkin?

Listni boshqa joyda boshlash uchun bizning kodimizda `slice` ishlatilishi mumkin! Kodingizni quyidagicha o'zgartiring:

```
import sys

if len(sys.argv) < 2:
    sys.exit("Too few arguments")

for arg in sys.argv[1:]:
    print("hello, my name is", arg)
```

E'tibor bering, ro'yxatni 0 dan boshlash o'rniga, biz kompilyatorga 1 dan boshlashni va 1: argumenti yordamida oxirigacha borishni aytish uchun kvadrat qavslardan foydalanamiz. Ushbu kodni ishga tushirganda, biz nisbatan oddiy sintaksis yordamida kodimizni yaxshilashimiz mumkinligini ko'rasiz.

Packages

Python-ning shu qadar mashhur bo'lishining sabablaridan biri shundaki, u funktsional imkoniyatlarni qo'shadigan ko'plab kuchli uchinchi tomon library lar mavjud. Uchinchi tomon library larni biz package deb nomlaymiz.

PyPI hozirda mavjud bo'lgan barcha uchinchi tomon paketlarining ombori yoki katalogidir.

cowsay - judaham qiziqarli package.

Python tizimida package larni tezda o'rnatish imkonini beruvchi pip deb nomlangan package kar menejeriga ega.

Terminal oynasida siz pip install cowsay yozish orqali cowsay paketini o'rnatishingiz mumkin. Endi siz ushbu package ni kodingizda ishlatishingiz mumkin.

Terminal oynasida code say.py kodini kiriting. Quyidagi kodni kiriting:

```
import cowsay
import sys

if len(sys.argv) == 2:
    cowsay.cow("hello, " + sys.argv[1])
```

E'tibor bering, dastur avval foydalanuvchi buyruq satrida kamida ikkita argument kiritganligini tekshiradi. Keyin, cowsay foydalanuvchi bilan gaplashishi kerak. python say.py David yozing va habarni ko'rasiz.

Kodingizni qo'shimcha ravishda o'zgartiring:

```
import cowsay
import sys

if len(sys.argv) == 2:
    cowsay.trex("hello, " + sys.argv[1])
```

API

API yoki "Application Programming Interface" sizga boshqalarning kodiga ulanish imkonini beradi.

requests - bu sizning dasturingizga veb-brauzer kabi harakat qilish imkonini beruvchi paackage.

Terminalingizda pip install requests kiriting. Keyin code itunes.py kodini kiriting.

Ma'lum bo'lishicha, Apple iTunes-ning o'z API-si bor, siz o'z dasturlaringiz orqali kirishingiz mumkin. Internet-brauzeringizda https://itunes.apple.com/search?entity=song&limit=1&term=shohruhhon saytiga tashrif buyurishingiz mumkin va tekst fayli yuklab olinadi.

Bu url dokumentatsiya orqali olingan. Etibor bering bu yerda biz term=shohruhhon orqali qoshiqchiga aloqador entity=song yani qoshiq qidiryapmiz va limit=1 orqali 1 qoshiq qidirish ga chegara ornatyapmiz. Yuklab olingan ushbu tekst fayliga qarab, siz Python-da biz avval dasturlagan formatga o'xshash formatni topishingiz mumkin.

Yuklab olingan matn faylidagi format JSON deb ataladi, bu matnga asoslangan format bo'lib, ilovalar o'rtasida matnga asoslangan ma'lumotlarni almashish uchun ishlatiladi. Tom ma'noda, Apple o'z Python dasturimizda sharhlashimiz mumkin bo'lgan JSON faylini taqdim etadi.

Terminal oynasida code itunes.py kodini kiriting. Kodni quyidagicha yozing:

```
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()

response = requests.get("https://itunes.apple.com/search?entity=song&limit=1&term=" +
sys.argv[1])
print(response.json())
```

response da sifatida requests.get ning qaytarilgan qiymati qanday saqlanishiga e'tibor bering. python itunes.py shohruhhon ishga tushiring, Apple tomonidan qaytarilgan JSON faylini ko'rasiz. Biroq, JSON javobi Python tomonidan dictionary ga aylantiriladi. Ma'lum bo'lishicha, Python JSON library si ga ega bo'lib, u bizga olingan ma'lumotlarni sharhlashda yordam beradi.

```
import json
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()

response = requests.get("https://itunes.apple.com/search?entity=song&limit=1&term=" +
sys.argv[1])
print(json.dumps(response.json(), indent=2))
```

E'tibor bering, json.dumps o'qilishi uchun indentdan foydalanadi. Va oqishga qulay format da print qib beradi. Endi e'tibor bering, natijada results deb nomlangan dictinoary ni ko'rasiz. Results deb nomlangan ushbu dictionary ichida ko'plab kalitlar mavjud. Dictionarydagi trackName qiymatiga qarang. Resutls da qanday trek nomini ko'rdiz?

Qanday qilib biz shunchaki o'sha qoshiq nomini chiqarishimiz mumkin? Kodingizni quyidagicha o'zgartiring:

```
import json
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()

response = requests.get("https://itunes.apple.com/search?entity=song&limit=50&term=" +
sys.argv[1])

o = response.json()
for result in o["results"]:
    print(result["trackName"])
```

response.json() natijasini qanday qabul qilayotganimizga va uni o ozgaruvchida saqlashimizga e'tibor bering (kichik harfda bo'lgani kabi). Keyin, biz o dagi natijalarni takrorlaymiz va har bir trek nomini chop etamiz. Natijalar sonini qanday qilib 50 ga oshirganimizga ham e'tibor bering. Dasturingizni ishga tushiring. Natijalarni ko'ring.

Siz kutubxona hujjatlari orqali request lar haqida ko'proq bilib olishingiz mumkin.

JSON haqida ko'proq ma'lumotni Python-ning JSON hujjatlarida olishingiz mumkin.

Libraries

Siz Python dasturchisi sifatida o'z kutubxonangizni yaratish qobiliyatiga egasiz!

Kod qismlarini qayta-qayta ishlatishni yoki hatto ularni boshqalar bilan baham ko'rishni xohlashingiz mumkin bo'lgan vaziyatlarni tasavvur qiling!

Biz ushbu kursda hozirgacha "salom" deyish uchun juda ko'p kod yozdik. Keling, "salom" va "xayr" deyishimizga imkon beradigan package yarataylik. Terminal oynasida code saids.py kodini kiriting, quyidagi kodni kiriting:

```
def hello(name):
    print(f"hello, {name}")


def goodbye(name):
    print(f"goodbye, {name}")
```

E'tibor bering, ushbu kod o'z-o'zidan foydalanuvchi uchun hech narsa qilmaydi. Biroq, agar dasturchi ushbu package ni o'z dasturiga import qilsa, yuqoridagi funktsiyalar tomonidan yaratilgan qobiliyatlar ularning kodida amalga oshirilishi mumkin.

Keling, biz yaratgan ushbu package dan foydalangan holda kodni qanday amalga oshirishimiz mumkinligini ko'rib chiqaylik. Terminal oynasida code say.py kodini kiriting, quyidagilarni kiriting:

```
import sys

from saying import goodbye

if len(sys.argv) == 2:
    goodbye(sys.argv[1])
```

E'tibor bering, ushbu kod saids dan xayrlashish qobiliyatini import qiladi. Agar foydalanuvchi terminalda kamida ikkita argument kiritgan bo'lsa, u terminalga kiritilgan qator bilan birga "xayr" deb aytadi.

Summint Up

Libraries Python imkoniyatlarini kengaytiradi. Ba'zi library lar Python ozida bor va ularni faqat import qilish kerak. Boshqalar pip yordamida o'rnatilishi kerak bo'lgan uchinchi tomon package lar. Siz o'zingiz yoki boshqalar foydalanishi uchun o'zingizning packagelaringizni yaratishingiz mumkin! Ushbu ma'ruzada siz ...

- Libraries
- Random
- Statistics
- Command-Line Arguments
- Slice
- Packages
- APIs
- Making Your Own Libraries