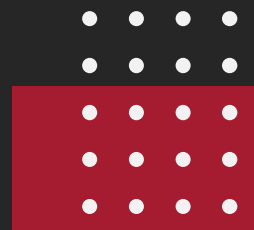
CS50x Iran

Lecture 4

Week note



پاییز و زمستان ۱۴۰۴

1 فهرست

- کتابخانه‌ها بخشی از کد هستند که توسط خودتان یا دیگران نوشته شده‌اند که می‌توانید از آن ها در برنامه تان استفاده کنید.
- پایتون به شما این امکان را می‌دهد تا قابلیت‌ها یا توابع خود را به عنوان " ماژول‌ها " با دیگران به اشتراک بگذارید.
- اگر کدی را از یک پروژه قدیمی کپی و پیست کنید، احتمالاً می‌توانید یک ماژول یا کتابخانه ایجاد کنید که می‌توانید آن را به پروژه جدید خود بیاورید.

- در پایتون کتابخانه‌ای به نام random[1] وجود دارد که می‌توانید از آن در پروژه‌های خود استفاده کنید.
- برای یک برنامه‌نویس استفاده از کدهای آماده که از قبل نوشته شده‌اند کار را آسان‌تر می‌کند.
- حال سوال اینجاست که چگونه یک ماژول را در کد خود بارگزاری کنیم؟ برای اینکار می‌توانید از واژه‌ی import استفاده کنید.
- داخل ماژول random ، یک تابع پیش فرض به نام random.choice(seq) وجود دارد. random ماژولی است که آن را در کد خود بارگزاری کرده اید و choice تابع موجود در آن ماژول است. این تابع به عنوان ورودی sequence که یک لیست است را دریافت می‌کند.

1.رندوم کتابخانه ای است که به شما امکان تولید اعداد تصادفی می دهد

- در پنجره‌ی ترمینال خود code generate.py را وارد کنید و در IDE خود کد زیر را بنوسید:

```
import random

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

توجه کنید که لیستی که به عنوان ورودی به تابع choice داده شده دارای براکت، کوتیشن و کاما است. با توجه به اینکه دو مورد را به عنوان ورودی به تابع داده‌اید، پایتون محاسبات لازم را انجام داده و یک شانس ۵۰ درصدی برای سکه انداختن به دو طرف (شیر و خط) در نظر گرفته است. با اجرای کد، متوجه خواهید شد که این کد به درستی عمل می‌کند!

- ما می‌توانیم کد خود را بهبود بخشیم. با استفاده از کلمه کلیدی from، می‌توانیم با دقت بسیار زیادی مشخص کنیم که چه چیزی را می‌خواهیم import کنیم. قبل از این، کد ما تمام محتویات کتابخانه random را وارد می‌کرد ولی فقط از یک تابع آن استفاده می‌کرد؛ حال، اگر بخواهیم فقط بخش کوچکی از یک ماژول را import کنیم، چه می‌شود؟ کد خود را به صورت زیر تغییر دهید :

```
from random import choice

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

توجه کنید که حالا ما می‌توانیم فقط تابع choice را از ماژول random ، import کنیم. از این به بعد ، دیگر نیازی به random.choice نخواهیم داشت. ما اکنون می‌توانیم فقط تابع choice را به تنهایی در برنامه خود استفاده کنیم. با این‌کار ما در مصرف منابع سیستم بهینه عمل کردیم و کد ما سریع‌تر اجرا می شود.

- به عنوان مثال، تابع random.randint(a, b) را در نظر بگیرید. این تابع یک عدد تصادفی بین a و b ایجاد می‌کند. کد خود را به شکل زیر تغییر دهید:

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

- باید توجه داشت که کد ما عددی بین ۱ تا ۱۰ انتخاب می‌کند.

- ما می‌توانیم تابع random.shuffle(x) را در برنامه خود معرفی کنیم که یک لیست را به ترتیبی تصادفی می‌چیند.

```
import random

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

نکته‌ی قابل توجه این است که تابع shuffle مقداری را به عنوان خروجی بر نمی‌گرداند و تغییرات را روی همان لیست ایجاد کرده و ترتیب عناصر را جابه‌جا می‌کند. برای تفهیم بهتر این موضوع، کد را چندین بار اجرا کنید.

- ما اکنون این سه راه را برای تولید اطلاعات تصادفی ارائه دادیم؛
- می‌توانید برای یادگیری بیشتر به مستندات پایتون سر بزنید.

- پایتون دارای یک کتابخانه‌ی داخلی به نام statistics است؛ اما چگونه باید از آن استفاده کرد؟
- avarge تابعی کارآمد و پر استفاده از این کتابخانه است. در پنجره‌ی ترمینال خود، code average.py را وارد کنید. در IDE خود کد زیر را وارد کنید:

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

شایان توجه است که ما کتابخانه‌ای متفاوت به‌نام statistics را import کردیم و تابع mean لیستی از مقادیر مختلف را ورودی می‌گیرد؛ در اخر میانگین این مقادیر را در ترمینال چاپ می‌کند. برای اجرا عبارت python average.py را تایپ کنید.

- این کتابخانه امکانات بیشتر نیز دارد ، در نظر داشته باشد که از آن‌ها استفاده کنید؛ برای یادگیری بیشتر میتواند به اسناد پایتونی کتابخانه‌ی آمار مراجعه کنید.

- تا حالا تمام مقادیر را داخل برنامه خود تعیین کرده‌ایم. اما اگر بخواهیم ورودی را از خط فرمان بگیریم چیکار باید بکنیم؟ به عنوان مثال، به جای تایپ python average.py در ترمینال، اگر بخواهیم python average.py 100 90 را تایپ کرده و میانگین ۱۰۰ و ۹۰ را بگیریم چه کاری باید انجام بدهیم؟

- برای اینکار می‌توانید از ماژولی به نام sys استفاده کنیم ، زیرا این امکان را به ما می‌دهد تا مقادیر آرگومان‌ها را از خط فرمان بگیریم.

- تابع argv در ماژول sys به ما اجازه می‌دهد تا درباره‌ی ورودی که کاربر در خط فرمان وارد کرده ، مطلع شویم. به طریقه‌ی استفاده sys.argv در کد زیر توجه کنید. در پنجره ترمینال، code name.py را تایپ کنید. در IDE خود کد زیر را وارد کنید:

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

کد ما وابسته به دستور تایپ شده‌ی کاربر است. در حال حاضر اگر شما در خط فرمان **python name.py David** را وارد کنید برنامه برای شما Hello, my name is David را چاپ خواهد کرد. مشخص است که sys.argv[1] جایی است که David در آن ذخیره شده؛ اما چرا؟ همان‌طور که قبلا اشاره کردیم اولین عنصر هر لیست در اندیس صفرم آن لیست قرار دارد؛ با توجه به این موضوع به نظر شما چه مقداری در sys.argv[0] ذخیره شده است؟ اگر حدس شما name.py باشد، درست گفتید!

- ایرادی که می‌توان از برنامه‌ی فعلی ما گرفت این است که اگر کاربر مقداری را در دستور وارد نکند و برنامه را اجرا کند؛ با ارور list index out of range روبه‌رو خواهد شد. دلیل این مورد هم وجود نداشتن مقدار در sys.argv[1] است. برای جلوگیری از این اتفاق می‌توانیم این‌گونه عمل کنیم:

```
import sys

try:
    print("hello, my name is",
          sys.argv[1])
except IndexError:
    print("Too few arguments")
```

حال کاربر با پیامی معنادار روبه‌رو خواهد شد که او را در استفاده از برنامه راهنمایی کرده و به او یادآوری می‌کند که در دستور نام دلخواه را نیز وارد کند.

ما می‌توانیم کمی بیشتر احتیاط کنیم و کد را به شیوه‌ای بهتر پیاده‌سازی کنیم، داریم:

```
import sys

if len(sys.argv) < 2:
    print("Too few arguments")
elif len(sys.argv) > 2:
    print("Too many arguments")
else:
    print("hello, my name is",
          sys.argv[1])
```

توجه کنید که اگر کد خود را تست کنید، مشاهده خواهید کرد که این Exception ها چگونه کنترل می‌شوند و به کاربر راهنمایی دقیق‌تری داده می‌شود. حتی اگر تعداد ورودی‌های کاربر زیاد یا کم باشد، به او دستورالعمل‌های واضحی در مورد چگونگی رفع مشکل ارائه می‌شود.

- در حال حاضر، کد ما از نظر منطقی درست است. با این حال، نکته‌ی خوبی در مورد جدا نگه داشتن بررسی ارور از بقیه‌ی کد ما، وجود دارد. چگونه می‌توانیم مدیریت ارور را از یکدیگر جدا کنیم؟ کد خود را به صورت زیر تغییر دهید:

```
import sys

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

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

قابل توجه است که ما از تابعی پیش فرض در کتابخانه‌ی sys به‌نام exit استفاده کردیم؛ که به ما اجازه می دهد در صورت ارائه خطایی توسط کاربر از برنامه خارج شویم. اکنون می‌توانیم مطمئن باشیم که برنامه آخرین خط از کد ما هرگز اجرا نمی‌کند و خطایی ایجاد نمی‌کند. بنابراین sys.argv این امکان را فراهم می‌کند تا کاربر ورودی را در خط فرمان به برنامه معرفی کند و sys.exit زمانی که برنامه‌ی ما به خطا بخورد آن را می‌بندد.

- **برای یادگیری بیشتر می‌توانید مستندات پایتون در رابطه با کتابخانه sys را بخوانید.**

- Slice یک تابع است که به ما اجازه می‌دهد تا با داشتن یک لیست به کامپایلر بگوییم دقیقا با کدام قسمت از آن می‌خواهیم کار کنیم. می‌توانیم با تعیین ابتدا و انتها به کامپایلر این موضوع را ارجاع دهیم، برای مثال کد زیر را در نظر بگیرید:

```
import sys

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

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

توجه داشته باشید اگه ما برنامه را با دستور python name.py David Carter Rongxin اجرا کنیم ، کامپایلر علاوه بر چاپ کردن اسامی "hello, my name is name.py" را نیز چاپ می‌کند. چگونه می‌توانیم مطمئن شویم که کامپایلر عنصر اولیه لیست ورودی یا name.py را نادیده می گیرد؟

- با استفاده از تابع Slice در کدمان می‌توانیم لیست را از جایی متفاوت شروع کنیم! کد را به صورت زیر اصلاح می‌کنیم:

```
import sys

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

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

توجه داشته باشید که با این رویکرد، کامپایلر دیگر عنصر اندیس صفرم لیست را در نظر نمی‌گیرد و از اندیس یکم شروع می‌کند؛ همچنین با [1:] ما مشخص می‌کنیم که پس شروع از اندیس یکم تا انتهای لیست حرکت کند.

- از دیگر دلایل شهرت پایتون کتابخانه‌های قدرتمند و متعدد third-party[1] پایتون هستند که باعث افزایش کارآمدی کد می‌شوند. ما این کتابخانه‌های third-party که به صورت یک فولدر پیاده‌سازی شده‌اند را پکیج می‌نامیم.
- PyPI یک ریپازیتوری برای تمام پکیج‌های third-party است که در حال حاضر در دسترس هستند.
- Cowsay یک پکیج شناخته‌شده است که اجازه می‌دهد یک «گاو» با کاربر صحبت کند.
- پایتون ابزار مدیریت پکیجی به‌نام pip را در اختیار شما قرار می‌دهد، که به شما اجازه نصب سریع پکیج‌ها را در سیستم می‌دهد.
- در پنجره‌ی ترمینال، میتوانید cowsay را با دستور pip install cowsay نصب کنید و بعد از اتمام نصب شما می‌توانید از این پکیج استفاده کنید.

1.کتابخانه هایی که توسط برنامه نویسان سرتاسر جهان نوشته شده‌اند .

- در پنجره‌ی ترمینال خود code say.py را تایپ کنید. در ادامه داریم:

```
import cowsay
import sys

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

توجه داشته باشید که برنامه ابتدا بررسی می کند کاربر حداقل دو آرگومان را در خط فرمان وارد کرده باشد. سپس گاو باید با کاربر صحبت کند. Python say.py David را تایپ کنید و سلام گفتن گاو به “David” را مشاهده خواهید کرد.

سپس کد خود را این‌گونه تغییر دهید:

```
import cowsay
import sys

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

در این مرحله یک t-rex در حال گفتن "hello" است.

- حال شما نحوه نصب پکیج‌های third-party را می‌دانید.
- برای یادگیری بیشتر در مورد پکیج cowsay میتوانید به cowsay مراجعه کنید.
- برای جست‌وجوی پکیج‌های بیشتر میتوانید به PyPI سر بزنید.

- API یا Application Program Interface به شما این امکان را می‌دهد تا به برنامه دیگران متصل شوید.
- requests پکیجی است که با آن کد شما مانند یک مرورگر وب رفتار می‌کند و به شما این امکان را میدهد تا بتوانید با ای‌پی‌آی ها کار کنید .
- در ترمینال، pip install requests را تایپ کنید. حالا code itunes.py را تایپ کنید.
- گفتنی است که Apple iTunes دارای ای‌پی‌آی خود بوده و شما می‌توانید در برنامه تان به آن دسترسی داشته باشید. در مرورگر اینترنت خود، به آدرس https://itunes.apple.com/search?entity=song&limit=1&term=weezer مراجعه کنید ، یک فایل متنی دانلود خواهد شد. دیوید این URL را با مطالعه مستندات API شرکت اپل ساخته است. توجه کنید که این Query به دنبال یک آهنگ، با محدودیت یک نتیجه، مرتبط با عبارت weezer است. با نگاه کردن به این فایل متنی که دانلود می‌شود، شما ممکن است فرمت آن را مشابه کدی که قبلا در پایتون نوشتیم ببینید.

- در فایلی که دانلود شده است ، فرمت فایل متنی JSON نام دارد . یک فرمت مبتنی بر متن است که برای تبادل داده‌های متنی بین برنامه‌ها استفاده می‌شود. به طور حرفه‌ای، اپل یک فایل JSON را ارائه می‌دهد که ما می‌توانیم آن را در برنامه پایتون خود تفسیر کنیم.
- در پنجره‌ی ترمینال خود، code itunes.py را تایپ کنید:

```
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())
```

توجه کنید که مقدار بازگشتی تابع requests.get در متغیر response ذخیره می‌شود. دیوید، با خواندن مستندات API اپل، می‌داند که آنچه بازگشت داده می‌شود، یک فایل JSON است. با اجرای دستور python itunes.py weezer ، فایل JSON بازگشت داده شده توسط اپل را مشاهده خواهید کرد. با این حال، پاسخ JSON توسط پایتون به دیکشنری تبدیل می‌شود.

نگاه کردن به خروجی ممکن است خسته‌کننده باشد!

- پایتون یک کتابخانه JSON داخلی دارد که می‌تواند به ما در تفسیر داده‌های دریافتی کمک کند. کد خود را به صورت زیر اصلاح کنید:

```
import json
import requests
import sys

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

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

توجه کنید که json.dumps طوری پیاده‌سازی شده است که از فاصله(indent) برای خواناتر کردن خروجی استفاده می‌کند. با اجرای دستور python itunes.py weezer ، شما همان فایل JSON را خواهید دید با این تفاوت که این بار خروجی خیلی خواناتر است.

حال توجه کنید که یک دیکشنری به نام results در خروجی وجود دارد، درون این دیکشنری results بسیاری از کلیدها وجود دارند .

به مقدار trackName در خروجی نگاه کنید؛ نام آهنگی که در نتیجه خروجی خود می‌بینید چیست؟

...

- چگونه می‌توانیم به سادگی نام همان آهنگ را خروجی بگیریم؟
- برای اینکار، کد خود را به صورت زیر تغییر دهید:

```
import json
import requests
import sys

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

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

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

توجه داشته باشید که چگونه resption.json() را می‌گیریم و آن را در o (مانند حروف کوچک) ذخیره می‌کنیم.سپس، نتایج را در o تکرار می کنیم و هر trackName را چاپ می‌کنیم. همچنین دقت کنید که چه طور تعداد محدود نتایج را به ۵۰ افزایش داده‌ایم.

برنامه خود را اجرا کنید تا نتایج را ببینید.

- برای یادگیری بیشتر درمورد کتابخانه requests میتوانید به مستندات این کتابخانه سر بزنید.
- مستندات پایتون را برای یادگیری بیشتر در مورد JSON دنبال کنید و کتابخانه‌ی خود را بسازید.

- حالا دیگه شما به عنوان یک برنامه نویس پایتون توانایی ایجاد کتابخانه خود را دارید!
- موقعیت‌هایی را تصور کنید که ممکن است بخواهید بارها و بارها از کد دوباره استفاده کنید یا حتی آن‌ها را با دیگران به اشتراک بگذارید!
- ما تاکنون در این دوره کدهای زیادی برای گفتن "hello" نوشته‌ایم. بیایید پکیجی ایجاد کنیم که به ما امکان می‌دهد «hello» و «goodbye» بگوییم. در پنجره ترمینال خود، کد code sayings.py را تایپ کنید و در ادیتور کد نوشته شده زیر را وارد کنید:

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


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

توجه داشته باشید که این کد به خودی خود کاری برای کاربر انجام نمی‌دهد. با این حال، اگر برنامه‌نویسی این پکیج را در برنامه خود وارد کند، توانایی‌های ایجاد شده توسط توابع بالا را می‌تواند در کد خود پیاده‌سازی کند.

- بیایید باهم ببینیم چگونه می‌توانیم کد را با استفاده از پکیجی که ایجاد کردیم پیاده‌سازی کنیم. در پنجره ترمینال، کد code say.py را تایپ کنید. در این فایل جدید در ویرایشگر متن خود، عبارت زیر را وارد کنید:

```
import sys

from saying import goodbye

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

توجه کنید که این کد توانایی های goodbye را از پکیج saying، import می کند. اگر کاربر حداقل دو آرگومان را در خط فرمان وارد کند، به همراه رشته‌ای که در خط فرمان وارد شده است، عبارت "goodbye" را به نمایش در می‌آورد.

کتابخانه‌ها توانایی‌های پایتون را گسترش می‌دهند. برخی از کتابخانه‌ها به‌طور پیش‌فرض در پایتون گنجانده شده‌اند و به‌سادگی می توانند وارد شوند. اما دسته‌ی دیگر از کتابخانه‌ها، کتابخانه‌ها third-party هستند که باید با استفاده از ابزار pip نصب شوند.

همچنین شما می‌توانید پکیج های خود را برای استفاده خودتان یا دیگران بسازید!

حال بیاید یک بار دیگر به سرفصل موضوعاتی که در این جلسه یاد گرفته‌اید را مرور کنیم :

- ✓ کتابخانه‌ها
- ✓ رندوم
- ✓ آمار
- ✓ آرگومان‌های خط فرمان
- ✓ اسلایس
- ✓ پکیج‌ها
- ✓ ای‌پی‌آی ها
- ✓ ساختن کتابخانه شخصی

CS50x Iran

Harvard's Computer Science 50x Iran