



## Python lecture 4

# جلسه قبل

- دستور break
- رشته‌ها
- آرایه
- فایل
- سوکت
- Banner grabbing
- راه اندازی و استفاده مقدماتی از git

# امروز

- سازمان بندی کد و مخفی سازی جزئیات
- انتزاع
- تقسیم وظایف
- توابع
- محدوده متغیرها
- معرفی `python-nmap`
- معرفی `ftplib`

# شیوه کد نویسی

- تا کنون

- مفاهیم اولیه زبان
- دانستن اینکه چگونه فایل‌های جدا برای انجام محاسبات مختلف بسازیم
- هر فایل شامل چندین خط کد
- هر کد شامل مجموعه‌ای از دستورالعمل‌ها

- مشکلات این روش کد نویسی

- برای برنامه‌ها با ابعاد کوچک ساده است
- غیر قابل استفاده برای برنامه‌های بزرگ
- مشکل مدیریت کد

# روش درست کد نویسی

- کد بیشتر الزاماً نشان دهنده بهتر بودن کد نیست
- ارزیابی برنامه نویسان بر اساس میزان کارایی کدشان است
- استفاده از توابع
- تقسیم وظایف و استفاده از انتزاع در کد

# مثال دستگاه ویدئو پروژکتور

- ویدئو پروژکتور یک جعبه سیاه است
- شما نمی دانید چگونه کار می کند
- می دانید که یک واسط برای ورودی و یک واسط برای خروجی دارد
- هر نوع وسیله الکترونیکی که می تواند با واسط ورودی ویدئو پروژکتور کار کند را می توانید به آن متصل کنید
- این جعبه سیاه به روشی که ما نمی دانیم تصویر ورودی را بر روی دیوار انداخته و آن را بزرگنمایی می کند
- **ایده استفاده از انتزاع:** شما نیازی نیست که بدانید ویدئو پروژکتور چگونه کار می کند تا بتوانید از آن استفاده کنید

# مثال دستگاه ویدئو پروژکتور

- نمایش تصویر برای فضاهای بزرگ مثل استادیوم نیاز به چندین ویدئو پروژکتور دارد
- هر پروژکتور یک قسمت از تصویر بعنوان ورودی دریافت کرده و یک خروجی منحصر بفرد ایجاد می‌کند
- همه ویدئو پروژکتور ها با یکدیگر کار می‌کنند تا بتوانند یک تصویر بزرگ را نمایش دهند
- **ایده تقسیم وظایف:** دستگاه‌های مختلف با یکدیگر کار می‌کنند تا بتوانند به یک هدف نهایی یعنی نمایش تصاویر بزرگ برای یک استادیوم دست یابند

استفاده از این مفاهیم در برنامه نویسی



## ایجاد ساختار با استفاده از تقسیم وظایف

- در مثال ویدئو پروژکتور، استفاده از دستگاه‌های جدا از هم
- در برنامه نویسی، تقسیم کد به ماژول‌ها
  - کد های مربوط به یک کاربرد خاص در یک ماژول قرار می‌گیرند
  - برای شکستن کد به قطعه‌های کوچک استفاده می‌شود
  - با این تفکر که کد ها قابلیت استفاده مجدد دارند
  - سازماندهی کدها
- در این جلسه وظایف کدها را با **توابع** تقسیم می‌کنیم
- در جلسات بعد وظایف کدها را با استفاده از **کلاس** ها تقسیم‌بندی می‌کنیم

# مخفی سازی جزئیات با استفاده از انتزاع

- در مثال ویدئو پروژکتور، دانستن اینکه چگونه از آن استفاده کنیم کافیست، نیازی نیست بدانیم که چگونه یک ویدئو پروژکتور بسازیم
- در برنامه نویسی، یک قطعه کد را بعنوان یک **جعبه سیاه** در نظر می گیریم
  - **نمی توانیم** جزئیات آن را ببینیم
  - **نیازی نداریم** که جزئیات را ببینیم
  - **نمی خواهیم** جزئیات را ببینیم
  - **مخفی سازی مفاهیم** برنامه نویسی خسته کننده
- برای دستیابی به انتزاع از توابع استفاده می کنیم

# توابع

- نوشتن قطعه کدها قابل استفاده مجدد، که تابع نامیده می‌شوند
- توابع در برنامه اجرا نمی‌شوند مگر اینکه **فراخوانی** شوند
- خصوصیات یک تابع
  - دارای یک **نام** است
  - دارای **پارامتر ورودی** است
  - یک **بدنه** دارد
  - **می‌تواند** مقداری را باز گرداند

# چگونه می‌توان یک تابع نوشت و آن را فراخوانی نمود؟

پارامتر ورودی      نام تابع      کلمه کلیدی تابع

```
def is_even(number):
```

```
    """this code will check whether  
    the number is even or odd  
    """
```

```
    print("inside is_even function")  
    return number%2==0
```

بدنه تابع

فراخوانی تابع

```
print(is_even(5))      →    False  
print(is_even(10))    →    True
```

# بدنه تابع

```
def is_even(number):  
    """this code will check whether  
       the number is even or odd  
    """
```

اجرای تعدادی دستور

```
print("inside is_even function")
```

```
return number%2==0
```


عبارتی که باید ارزیابی شود و نتیجه  
آن بازگردانده شود

کلمه کلیدی برای بازگرداندن یک مقدار

# محدوده متغیرها


- زمانی که یک تابع فراخوانده می‌شود پارامترهای **ظاهری** به پارامترهای **واقعی** انتساب داده می‌شوند

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))  
    return x
```



```
x=3
```

```
z=f(x)
```



پارامتر واقعی

# محدوده متغیرها

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))  
    return x
```

```
x=3  
z=f(x)
```

محدوده سراسری

f چندین خط کد

x 3

z

محدوده تابع f

x 3

# محدوده متغیرها

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))  
    return x
```

```
x=3  
z=f(x)
```

محدوده سراسری

f چندین خط کد

x 3

z

محدوده تابع f

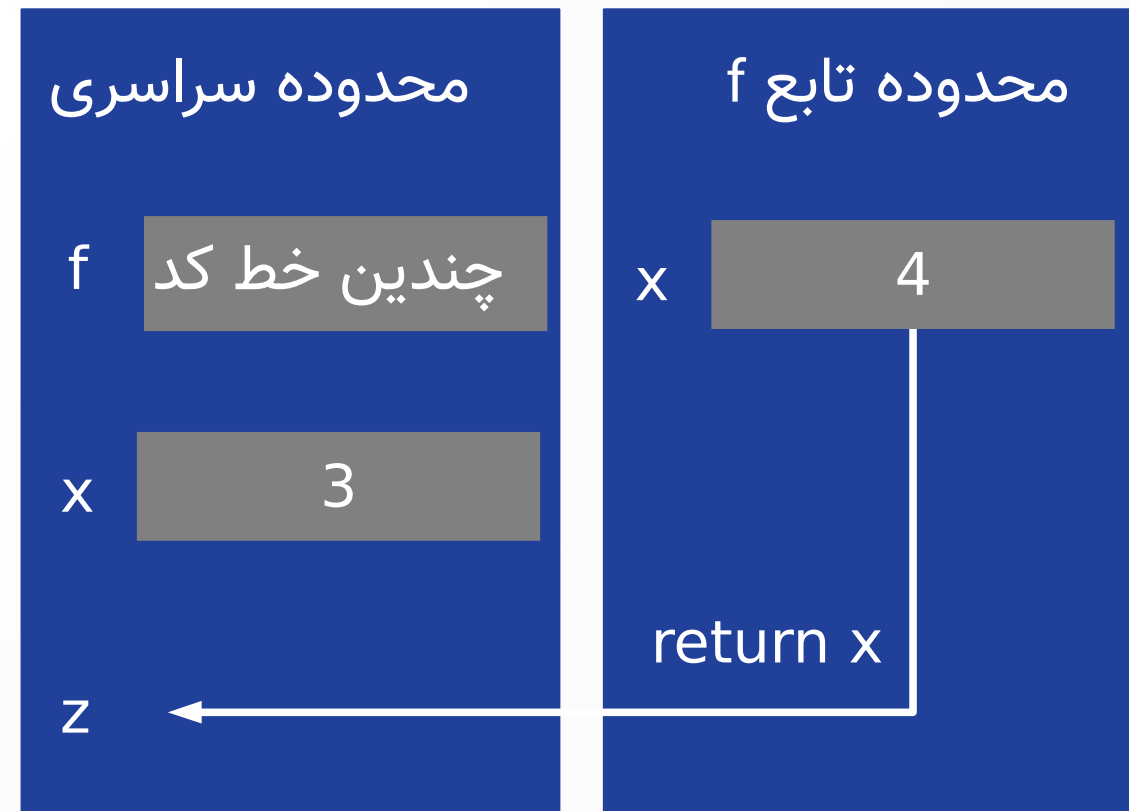
x 4



# محدوده متغیرها

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))  
    return x
```

```
x=3  
z=f(x)
```



# محدوده متغیرها

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))  
    return x
```

```
x=3  
z=f(x)
```

محدوده سراسری

f چندین خط کد

x 3

z 4

# هشدار

- در صورت عدم استفاده از دستور **return** پایتون مقدار **None** را بعنوان مقدار بازگشتی تابع بر می‌گرداند

```
def f(x):  
    x=x+1  
    print("inside f(x):{0} ".format(x))
```

```
x=3  
z=f(x)  
print(z)    → prints None
```

# تقسیم وظایف و انتزاع

- در صورت استفاده با هم قدرتمند اند
- کد ها می‌توانند چندین مرتبه استفاده شوند ولی تنها به یک بار دیباگ کردن نیاز دارند

# Nmap چیست

- مخفف network mapper
- امکان اسکن شبکه‌های بزرگ به صورت کارآمد
- امکان استفاده از انواع متدهای اسکن به سادگی (TCP SYN,UDP,TCP FIN,TCP  
(...,NULL
- وجود یک لایبرری پایتونی برای اسکن شبکه

# لینک داندود و مثال‌هایی از روش استفاده از nmap

- <https://xael.org/pages/python-nmap-en.html>
- <https://www.programcreek.com/python/example/92225/nmap.PortScanner>
- <https://nmap.org/book/man-port-scanning-techniques.html>
- <https://pypi.org/project/python-nmap/>

# روش دانلود و نصب لایبرری nmap

- `pip install python-nmap`

# روش نصب لایبرری nmap برای پایتون

The screenshot displays the PyCharm Community Edition interface. The top toolbar shows the 'Run' button (a green play icon). The main editor window is open to the file `4_0_using_nmap_port_scanner.py` within the `lecture4` project. The code in the editor is as follows:

```
22 for host_name in hosts:
23     ports = scanner(host_name)
24     for port in ports:
25         print("{0} : {1} {2}".format(host_name, port, ports.get(p
26     print("\n*****\n")
27
```

Below the editor is a terminal window. It shows the execution of the command `pip3 install python-nmap`. The output indicates that the package was successfully installed.

```
+ symm# pip3 install python-nmap
x Collecting python-nmap
Installing collected packages: python-nmap
Successfully installed python-nmap-0.6.1
symm#
symm#
symm#
```

The bottom status bar of the IDE shows the current file encoding as UTF-8 and the Git status as master.



# استفاده از لایبرری python-nmap

```
import nmap

def scanner(hostname):
    nm = nmap.PortScanner()
    nm.scan(hostname, '21-443')
    for host_names in nm.all_hosts():
        ports = nm[host_names].get('tcp')
        return ports

def get_host_names():
    f = open('host_names', 'r')
    hosts = f.read()
    hosts = hosts.rstrip('\n')
    return hosts

hosts = get_host_names()
for host_name in hosts:
    ports = scanner(host_name)
    for port in ports:
        print("{0} : {1} {2}".format(host_name, port, ports.get(port)))
print("\n*****\n")
```

# استفاده از لایبری ftplib

```
from ftplib import FTP

def check_credential(hostName,userName,password):
    ftp = FTP(hostName)
    try:
        ftp.login(userName,password)
        print("successfully connected to")
        print("{0} {1}:{2}".format(hostName,userName,password))
    except Exception :
        pass

def get_user_names():
    f = open('user_names', 'r')
    user_names = f.read()
    user_names = user_names.rspllit('\n')
    return user_names

def get_passwords():
    f = open('passwords', 'r')
    passwords = f.read()
    passwords = passwords.rspllit('\n')
    return passwords

user_names=get_user_names()
passwords=get_passwords()
hostName='127.0.0.1'
for user_name in user_names:
    for password in passwords:
        check_credential(hostName,user_name,password)
```