برنامه نویسی کامپیوتر ـ زمستان ۹۷
تدریس توسط: حسین جوهری
دانشگاه خواجه نصیرالدین طوسی

خلاصه هفته هشتم

مفاهیم: **تابع** **function**

در این هفته با ساختار تابع در پایتون آشنا می‌شویم. چند مسئله ساده را با استفاده از ساختار تابع پیاده‌سازی می‌کنیم و در انتها یک برنامه برای ثبت نام و ورود کاربران می‌نویسیم.

۱ تابع در پایتون

تابع مانند آنچه در ریاضیات دبیرستانی با آشنا هستیم، یک یا چند ورودی دریافت می‌کند و خروجی مورد نظر را تولید می‌کند. برای مثال تابع $f(x) = x^2$ یک عدد را دریافت کرده و توان 2 آن را تولید می‌کند. تابع $f(x, y) = x + y$ دو ورودی x و y را دریافت کرده و حاصلجمع آنها را تولید می‌کند. امکان تعریف توابع تقریبا در همه زبانهای برنامه‌نویسی وجود دارد و در پایتون نیز گنجانده شده است. امکان تعریف توابع مزایای قابل توجهی را به برنامه‌نویسی اضافه می‌کند و کاربرد فراوانی دارد.

در پایتون تابع با استفاده از کلمه کلیدی def تعریف می‌شود. سپس نام تابع آورده می‌شود و در ادامه در داخل پرانتز لیست ورودیهای مورد نیاز تابع آورده می‌شود. به هر ورودی مورد نیاز تابع یک پارامتر گفته می‌شود. در خط پایینتر و بعد از ایجاد فاصله مناسب بدنه تابع آورده می‌شود که مجموعه از دستورات است که معمولا پردازشی را روی ورودیهای تابع انجام می‌دهد. بعد از انجام محاسبات لازم، نتیجه تابع با استفاده از دستور return گزارش می‌شود و یا اصطلاحا برگردانده می‌شود. به این مقدار برگشتی تابع نیز گفته می‌شود.

```
def function_name( list of parameters ):
      statement 1
      statement 2
      statement 3
      ...
      return  ...
```

مثال: تابع square یک عدد را گرفته و مربع آن را برمی‌گرداند.

```
def square(n):
    return n*n
```

مثال: تابع max دو عدد a و b را گرفته و بزرگتر آنها را برمی‌گرداند.

```
def max(a,b):
    if (a >= b):
       return a
    else:
      return b
```

مثال: تابع Max دو عدد a و b را گرفته و بزرگتر آنها را برمی‌گرداند.

```
def Max(a,b):
    if (a >= b):
       return a
    else:
      return b
```

در اینجا یک شکل استفاده از تابع Max آورده شده است. مقدار برگشتی تابع در متغیر c قرار می‌گیرد.

```
y = 12
z = 30
c = Max(y,z)
print(c)   # prints 30
```

یک تابع می‌تواند مقدار برگشتی نداشته باشد.
مثال: تابع greet عدد n را دریافت کرده و به تعداد n بار رشته hello را چاپ می‌کند.

```
def greet(n):
    for i in range(n):
       print("hello")
```

با فراخوانی $\text{greet}(14)$ به تعداد 14 بار hello چاپ می‌شود.
چون تابع greet مقدار برگشتی ندارد در کد زیر متغیر x مقدار None را خواهد گرفت که به معنی هیچی (پوچ) است. دقت کنید که None با صفر فرق دارد.

```
x = greet(12)

print(x)  # prints None
```

یک تابع می‌تواند پارامتر نداشته باشد.
مثال: تابع farewell به تعداد 3 بار رشته goodbye را چاپ می‌کند.

```
def farewell():
    for i in range(3):
       print("goodbye")
```

مثال: تابع isPrime عدد n را دریافت کرده و در صورتی که n اول باشد مقدار True و در غیر این

صورت مقدار False را برمی‌گرداند.

```
def isPrime(n):

    if (n==1):
      return False

    for i in range(2,n):
        if (n % i == 0):
            return False

    return True
```

مثال: تابع nonZeroItems لیست a را به عنوان پارامتر ورودی می‌گیرد و لیستی را برمی‌گرداند که حاوی عناصر غیر صفر لیست a است.

```
def nonZeroItems(a):

    b = []

    for i in range(len(a)):
        if (a[i] != 0):
            b.append(a[i])

    return b
```

۱.۱ حوزه دید متغیرها

متغیرهایی که در بدنه تابع ایجاد می‌شوند (در سمت چپ انتصاب قرار می‌گیرند) تنها در خود تابع قابل دسترسی هستند (حوزه دیدشان تنها داخل تابع است) اصطلاحا این متغیرها، متغیرهای محلی local variables برای تابع هستند. به مثال زیر دقت کنید.

```
def fun():
    x = 12
    return 1

fun()
print(x)  # Error
```

در اینجا دستور print با خطا مواجه می‌شود چون متغیر x یک متغیر محلی برای تابع fun است و در خارج آن قابل دسترسی نیست.

دستور print در کد زیر مقدار 12 را چاپ می‌کند و نه مقدار 20 را.

```
x = 12

def fun():
    x =  20
    return 1

fun()
print(x)  # Prints 12
```

متغیر x داخل تابع چون در سمت چپ یک انتصات قرار گرفته است ($x = 20$) تابع آن را به عنوان یک متغیر جدید در نظر می‌گیرد و مختص خود تابع است. با وجود هم اسم بودن با متغیر x خارج تابع فرق دارد.

به کد زیر دقت کنید.

```
x = [12]

def fun():
    x.append(20)
    return 1

fun()
print(x)  # Prints [12,20]
```

در مثال بالا، عمل append روی متغیر x خارج از تابع اثر می‌گذارد. در اینجا چون متغیر جدیدی ایجاد نشده است (عمل انتصابی صورت نگرفته است) متغیر x همان متغیر بیرون تابع است.

در مثال زیر چون x طرف چپ انتصاب است یک متغیر جدید به نام x ایجاد می‌شود که فقط مختص داخل تابع است. به همین دلیل عمل اضافه کردن 20 به انتهای لیست x در داخل تابع روی متغیر بیرون تابع تاثیری نمی‌گذارد.

```
x = [12]

def fun():
    x = x + [20]
    return 1

fun()
print(x)  # Prints [12]
```

نکته مهم: بطور کلی عمل انتصاب در پایتون مثل ایجاد یک متغیر جدید است.

۲ یک برنامه کاربردی با استفاده از توابع: مدیریت کاربران

در اینجا می‌خواهیم یک برنامه کاربردی ساده برای مدیریت کاربران بنویسیم. این برنامه دو امکان ساده دارد: کاربر می‌تواند با انتخاب اسم کاربری جدید و کلمه عبور ثبت نام کند (عمل sign up) و یا با داشتن اسم کابری و کلمه عبور به سیسم وارد شود (عمل login).

برنامه وقتی اجرا می‌شود از کاربر می‌خواهد برای عمل sign up عدد 1 را وارد کند، برای login کردن عدد 2 را وارد کند و برای خروج از برنامه عدد 3 را وارد کند. این اصطلاحا منوی اصلی برنامه است. بعد از انتخاب و خاتمه کار دوباره منوی اصلی ظاهر می‌شود. این امکان را با استفاده از یک حلقه while پیاده‌سازی کرده‌ایم. کارهای مربوط به اعمال sign up و login در توابع مجزا گنجانده شده است.

```
n = int(input("For signup enter 1, For login enter 2, For exit enter 3:"))

while(n==1 or n==2):
     if (n == 1):
         signup()
     if (n == 2):
         login()
     n = int(input("For signup enter 1, For login enter 2, For exit enter 3"))

print("Goodbye.")
```

برای ذخیره‌سازی اسم کاربران و کلمه‌های عبور از دو لیست userNames و passWords استفاده کرده‌ایم.

```
userNames=[]
passWords=[]
```

کد زیر تعریف تابع signup را نشان می‌دهد. وقتی کاربری می‌خواهد ثبت نام کند، یک اسم کابری جدید را وارد می‌کند. در صورتی که قبلا اسم کابری داده شده در لیست userNames باشد پیامی چاپ می‌شود مبنی بر اینکه قبلا اسم داده شده استفاده شده است در غیر این صورت کلمه عبور درخواست می‌شود. کلمه عبور باید ویژگی‌های خاصی داشته باشد. چک کردن این ویژگیها در تابع isGoodPass انجام می‌شود. در صورتی که تابع isGoodPass کلمه عبور داده شده را تایید کرد مقدار True را برمی‌گرداند در غیر اینصورت پیامی مبنی بر نامناسب بودن کلمه عبور چاپ می‌شود. در صورت تایید اسم کابری و کلمه عبور، داده‌های مربوطه به لیستهای userNames و passWords اضافه می‌شوند.

```
def signup():
    print("Sign up ...")
    username = input("Please enter a user name:")
    if (username in userNames):
        print("user name is already in use. please try again.")
    else:
        password = input("Please enter a password:")
        if(isGoodPass(password)):
            userNames.append(username)
            passWords.append(password)
        else:
            print("The password is unacceptable. Please try again.")
```

کد زیر تابع isGoodPass را نشان می‌دهد. اگر طول کلمه داده شده از ۸ کمتر باشد مقدار False برگردانده می‌شود. علاوه بر این کلمه داده شده باید حاوی حرف بزرگ، حرف کوچک و عدد نیز باشد.

```
def isGoodPass(x):
    hasDigit = False
    hasUpper = False
    hasLower = False
    if(len(x)<8):
        return False
    for i in range(len(x)):
        if(x[i].isdigit()):
            hasDigit = True
        if (x[i].isupper()):
            hasUpper = True
        if (x[i].islower()):
            hasLower = True

    if(hasDigit and hasUpper and hasLower):
        return True
    else:
        return False
```

در کد زیر تابع login تعریف شده است. کاربر یک اسم کاربری وارد می‌کند. در صورت وجود برنامه محل اسم کاربری داده شده را در لیست userNames پیدا می‌کند. محل اسم مربوطه در متغیر index قرار می‌گیرد. در صورت عدم وجود مقدار index برابر با -1 خواهد بود. در صورت وجود اسم کاربری، کلمه عبور داده شده با آنچه در لیست passWords ذخیره شده مقایسه می‌شود و پیام مناسب چاپ می‌شود.

```
def login():
    print("Login ... ")
    username = input("Please enter a user name:")
    password = input("Please enter a password:")
    index = -1
    for i in range(len(userNames)):
        if(userNames[i]==username):
            index = i
            break

    if(index == -1):
        print("User name does not exist.")
    else:
        if(passWords[index]!=password):
            print("Wrong password.")
        else:
            print("Login Successful.")
```