حلقه ها

- به طور معمول در پایتون هر خط کد اجرا شده و پس از آن خط کد بعدی اجرا می شود و به همین ترتیب تمامی خطوط کد اجرا می شود
- گاهی نیاز است تا یک دستور (خط کد) چندین بار اجرا شود یا به عبارت دیگر تا زمانی که شرایط مشخصی حاکم است، دستور مذکور اجرا شود. در این گونه مواقع از حلقه ها (Loops) استفاده می شود.

نوع حلقه	توصیف
while loop	تا زمانی که شرطی صادق (True) باشد، دستور و یا مجموعه ای از دستورات را اجرا می کند و همواره قبل از اجرا، صادق بودن دستور را بررسی می کند
for loop	توالی از دستورات را به طور مکرر اجرا می کند
nested loops	مجموعه ای از حلقه های تو در تو

حلقه ها
while

```
1 number = 0
2 while number <=5:
3   print(number * 2)
4   number += 1
5 print("End of Loop!")
```

```
0
2
4
6
8
10
End of Loop!
```

حلقه ها
for

- عبارت های for این توانایی را دارند که بر روی توالی تکرار شوند که این توالی می تواند لیست یا رشته باشد
- منظور از iterating variable متغیری است که حین اجرای حلقه تعریف شده و به اجرای دستورات داخل حلقه کمک می کند.
- منظور از sequence توالی است که iterating variable بر روی آن تکرار می شود
- در صورتی که می خواهیم بر روی توالی از اعداد عمل تکرار را انجام دهیم می توان از تابع داخلی ()range استفاده نمود
  - این تابع یک تکرار شونده ایجاد می کند که می توان از آن برای به دست آوردن توالی از اعداد استفاده کرد

Layout of for loops

```
for iteration_var in sequence:
  statements(s)
```

```
1 for number in [0,1,2,3,4,5]:
2   x = number * 2
3   print(x)
```

```
0
2
4
6
8
10
```

حلقه ها
for

Syntax of range() function

```
range(start, stop, step)
```

[start,stop)

```
1 print("First Form: ",range(10, 20, 1))
2 print("Second Form: ",list(range(10, 20, 1)))
```

```
First Form:  range(10, 20)
Second Form:  [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

- مثالی از تابع داخلی ()range
- روش جایگزینی که وجود دارد این است که از اندیس ها برای تکرار کردن استفاده کرد که در اینصورت از دستور دیگری به نام ()len همراه با دستور range استفاده می کنیم
- یادآوری: از دستور ()len برای به دست آوردن طول یا تعداد عناصر استفاده می شود

```
1 Grocery = ["Apple","Pineapple","Strawberry"]
2 for fruit in Grocery:
3   print(fruit)
4   print("You have to buy: ", fruit)
```

```
Apple
You have to buy:  Apple
Pineapple
You have to buy:  Pineapple
Strawberry
You have to buy:  Strawberry
```

```
1 Grocery = ["Apple","Pineapple","Strawberry"]
2 for i in range(len(Grocery)):
3   print(i)
4   print("You have to buy: ", Grocery[i])
```

```
0
You have to buy:  Apple
1
You have to buy:  Pineapple
2
You have to buy:  Strawberry
```

حلقه های تو در تو
Nested loops

Layout of Nested loops

Nested For loops

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

Nested While loops

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

```
for i in range(1,4):
    for j in range(1,4):
        k = i*j
        print (k)
    print()
```

```
1
2
3

2
4
6

3
6
9
```

```
i = 1
j = 5
while i < 4:
    while j < 8:
        print(i, ",", j)
        j = j + 1
        i = i + 1
```

```
1 , 5
2 , 6
3 , 7
```

عبارات کنترلی حلقه ها
LOOP CONTROL STATEMENTS

این عبارات امکان کنترل توالی اجرایی حلقه ها را فراهم می سازند

عبارت	عملکرد
break	عملکرد حلقه را متوقف کرده و عبارتی که بلافاصله بعد از حلقه قرار دارد را اجرا می کند
continue	ادامه اجرای حلقه را متوقف کرده و به ابتدای حلقه می رود
pass	زمانی که به دستوری نیاز است تا syntax کد کامل شود ولی دستور خاصی مدنظر نیست!