

## More Loops

---

مروری داشته باشیم بر حلقه های تو در تو و مسائل پیچیده جلسه قبل :


```
test2.py > ...
Click here to ask Blackbox to help you code faster
1  user_input_list = []
2
3  for i in range(100):
4      if i == 10:
5          break
6
7      user_input = input("enter your value ...")
8
9      user_input_list.append(user_input)
10
11 print(user_input_list)
```

در این سوال تایین کردیم که حلقه تا 100 تا ادامه پیدا کند و در آن مشخص کردیم که اگر شمارشگر به 10 تا رسید حلقه شکسته شود و اگر اینطوری نباشه یعنی شمارشگر زیر 10 تا باشه از کاربر میگیره ورودی رو و به لیست اضافه میکنه.

حواستون باشه که برای بار 10 امین باز دیگه حلقه اجرا نمیشه چون کد های پایتونی از بالا به پایین میان پس برای 10 امین بار اجرا نمیشه مگر اینکه اون بریک رو پایین بقیه کد ها بزاریم تا بعد از اجرا شدن شون اجرا بشه

---


```
test2.py > ...
Click here to ask Blackbox to help you code fast
1 a = ['a' , 'b' , 'c' , 'd']
2 b = ['e' , 'f' , 'c' , 'd']
3
4 for i in a :
5     for j in b :
6         if i == j:
7             print(i)
```



در این سوال از حلقه تو در تو استفاده شده . همونطور که گفتم بهتون اول یک ایتm از a وارد حلقه اولی میشود و به ازای هر آیتm از a یکبار کل حلقه دومی اجرا میشود و این ترکیب به صورت دکاریتی است یعنی اول a میاد و با a b c d e بررسی میشود و حلقه اولی بدون جلو میرود یعنی b و این دوباره به اعزایش کل حلقه دوم اجرا میشود تا آخر

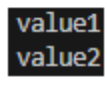

حلقه در دیکشنری ها :

```
test2.py > ...
Click here to ask Blackbox to help you c
1 my_dic = {
2     "key1" : "value1",
3     "key2" : "value2"
4 }
5
6 for key in my_dic.keys():
7     print(key)
```



اینجا ما اومدیم روی کلید های یک دیکشنری حلقه زدیم همونطور که میدونید تابع keys میاد و کلید های یک دیکشنری رو برمیگردونه

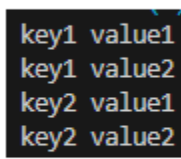

```
test2.py > ...  
Click here to ask Blackbox to help you code faster  
1 my_dic = {  
2     "key1" : "value1",  
3     "key2" : "value2"  
4 }  
5  
6 for value in my_dic.values():  
7     print(value)
```



این جا هم اومدیم همین کار رو برای مقادیر اون دیکشنری کردیم .

حالا ما میخاییم کاری کنیم که روی یک دیکشنری فقط با یه حلقه بیاییم جفت این مقادیر رو به صورت دکارتی بگیریم :

```
test2.py > ...  
Click here to ask Blackbox to help you code faster  
1 my_dic = {  
2     "key1" : "value1",  
3     "key2" : "value2"  
4 }  
5  
6 for key in my_dic.keys():  
7     for value in my_dic.values():  
8         print(key , value)
```




به اعضای هر عضو از کلید ها یک بار حلقه روی کل مقادیر value اعمال میشه

حالا نمیخاییم دکارتی بشه :

گفتیم که توی حلقه ها ما یه متغیر با اسم دلخواه داریم . اگر بیاییم 2 یا چند تا متغیر بزاریم مثل این میمونه که ما چند تا حلقه زدیم :

```
test2.py > ...
Click here to ask Blackbox to help you code faster
1 my_dic = {
2     "key1" : "value1",
3     "key2" : "value2"
4 }
5
6 for key , value in my_dic.items():
7     print(key , value)
```




```
key1 value1
key2 value2
```

متد items گفتیم میاد و کلید ها و مقادیر رو به صورت جفت در یک لیست میاره . حالا میاییم 2 تا متغیر تعریف میکنیم و داخل اون لیست های که items به ما داده هم حلقه میزنیم .

برای راحتی اینجارو دقت کنید :

```
test2.py > ...
Click here to ask Blackbox to help
1 my_dic = {
2     "key1" : "value1",
3     "key2" : "value2"
4 }
5
6 print(my_dic.items())
```



```
dict_items([('key1', 'value1'), ('key2', 'value2')])
```

متد items میاد و به صورت جفت کلید ها و مقادیر رو براتون میاره و ما با تعریف کردن 2 تا متغیر در حلقه یکبار در خود لیست ها و یکبار در ایتم هاشون حلقه میزنیم

## While Loop

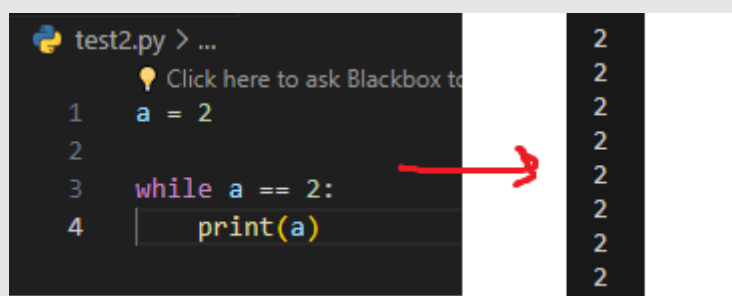
---

حلقه های while فرقتشون با for اینکه for تا دامنه ای که بهش میگفتی اجرا میشد ولی حلقه while تا جایی ادامه پیدا میکنه که شرط آن حلقه درست باشد

While condition :

Expression

این سینتکس حلقه وایل است :



```
test2.py > ...  
Click here to ask Blackbox to  
1 a = 2  
2  
3 while a == 2:  
4     print(a)
```

2  
2  
2  
2  
2  
2  
2  
2  
2

حواستون باشه که اگر شرط حلقه همیشه درست باشد حلقه بی نهایت میشود و این خوب نیست مگر اینکه کنترل شده باشد

---

یک نمونه از حلقه بی نهایت کنترل شده :

```
test2.py > ...
Click here to ask Blackbox to help you code faster
1 user_input = input('enter a thing : ')
2
3 while user_input != "exit":
4     print(user_input)
5
6     user_input = input("enter again ...")

enter a thing : erfana
erfana
enter again ...mama
mama
enter again ...exit
PS C:\Users\hamid\Desktop\New folder (2)\test>
```

اینجا مشخص کردیم که تا وقتی چیزی که کاربر تایین کرده exit نیست حلقه رو اجرا کنه و همونطور که میبینید من تا وقتی چیزی رو که exit نباشه بزنم حلقه ادامه پیدا میکنه

یا مثلا همین کد رو با دستور break میتونیم بنویسیم :

```
test2.py > ...
Click here to ask Blackbox to help you code faster
1 user_input = input('enter a thing : ')
2
3 while True:
4     print(user_input)
5
6     user_input = input("enter again ...")
7
8     if user_input == 'exit':
9         break

enter a thing : erfana
erfana
enter again ...mamada
mamada
enter again ...exit
```

اینجا اومدیم گفتیم که حلقه تا بی نهایت ادامه پیدا کنه و اگر اینپوت ما exit بود حلقه رو بکشنه

پیاده سازی سیستم پسورد :

```
test2.py > ...  
Click here to ask Blackbox to help you code faster  
1 pass1 = input("enter your password")  
2  
3 while True:  
4  
5  
6     pass2 = input("enter your password again to login")  
7  
8     if pass1 == pass2:  
9         break  
10    else :  
11        print('you have entered wrong password')  
12  
13 print("you are loggedin")
```

در این سیستم یک حلقه بی نهایت هست و تا موقعی که break نشه ادامه پیدا میکنه اینجا اگر پسور 1 با 2 برابر باشه break میشه و از حلقه خارج میشه و اگر نباشه پیام میده که رمز اشتباست و دوباره امتحان کن و حلقه دوباره از اول اجرا میشه

حالا همین رو میشه به صورت دیگه ای هم نوشت :

```
test2.py > ...  
Click here to ask Blackbox to help you code faster  
1 pass1 = input("enter your password")  
2 pass2 = input("enter your password again to login")  
3  
4 while pass1 != pass2:  
5     print('you have entered wrong password ...')  
6     pass2 = input('enter your password again ...')  
7  
8  
9  
10  
11  
12 print("you are loggedin")
```

اینجا اگر پسورت 1 با 2 برابر نباشه کد داخل حلقه میوفته

---

حالا همین رو میخاییم با منطق برعکس اجرا کنیم :



```
test2.py > ...
  Click here to ask Blackbox to help you code faster
1 pass1 = None
2
3 while True :
4     pass1 = input('enter your password ...')
5
6     pass2 = input('enter your password again to login ')
7
8     if pass1 == pass2:
9         break
10    else :
11        print("you have entered your password wrong...")
12
13 print("you are loggedin")
```

حالا زمانی که پسورد درست باشه از حلقه خارج میشه

-----

یک مثال دیگه :

```
test2.py > ...
  Click here to ask Blackbox to help you code faster
1  play_status = True
2
3  while play_status == True :
4      print('you are plagin now')
5
6      play_again = input("do you wanna play agian : ")
7
8      if play_again == 'n' or play_again == 'no':
9          play_status = False
10
11 print("thanks for playing")
```

```
test2.py > ...
  Click here to ask Blackbox to help you code faster
1  my_dic = {
2      'key1' : 'value'
3  }
4
5  while len(my_dic.keys()) < 6:
6      print("enter a new key and value pairs")
7
8      a = input("enter a key")
9      b = input("enter a value")
10
11     my_dic.update({a:b})
12
13 print(my_dic)
```

در این کد اول حلقه چک میکند تا تعداد کلید ها زیر 6 تا باشد و از کاربر کلید و مقدار رو میگیره و دیکشنری رو اپدیت میکنه تا موقعی که به 6 تا برسه بعد هم در آخر چاپ میکنه خود دیکشنری اپدیت شده رو

چرا {'a':'b'} نداشتیم ؟ چون اگر اینو بزاری میاد یه کلید به مقدار 'a' و مقدار 'b' میزاره  
ما اینجا میخاییم بیاد متغیر a رو بزاره برای کلید و متغیر b رو بزاره برای value

این 2 تا کد رو مقایسه کنید تا بهتر بفهمید :

```
test2.py > ...
Click here to ask Blackbox to help you code faster
1 my_dic = {
2     'key1' : 'value'
3 }
4
5 a = 'erfan'
6 b = 'mamad'
7
8 my_dic.update({"a" : "b"})
9
10 print(my_dic)
```

```
test2.py > ...
Click here to ask Blackbox to help you code faster
1 my_dic = {
2     'key1' : 'value'
3 }
4
5 a = 'erfan'
6 b = 'mamad'
7
8 my_dic.update({a : b})
9
10 print(my_dic)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hamid\Desktop\New folder (2)\test> /test2.py
PS C:\Users\hamid\Desktop\New folder (2)\test> /test2.py
{'key1': 'value', 'a': 'b'}
```

```
PS C:\Users\hamid\Desktop\New folder (2)\test> /test2.py
{'key1': 'value', 'erfan': 'mamad'}
```

```
PS C:\Users\hamid\Desktop\New folder (2)\test> /test2.py
```

خروجی هارو چک کنید اول

توی یکیش متغیر رو دادیم یعنی راستی تا متغیر بشینه توی دیکشنری ولی توی  
راستی اومدیم 2 تا رشته رو دادیم و رشته میشینه توی دیکشنری