

[C1113-OO-EX4][TahrehGholami-401114037180030]

n = 5

while n > 0 :

 print(n)

 n = n - 1

print('Blastoff!')

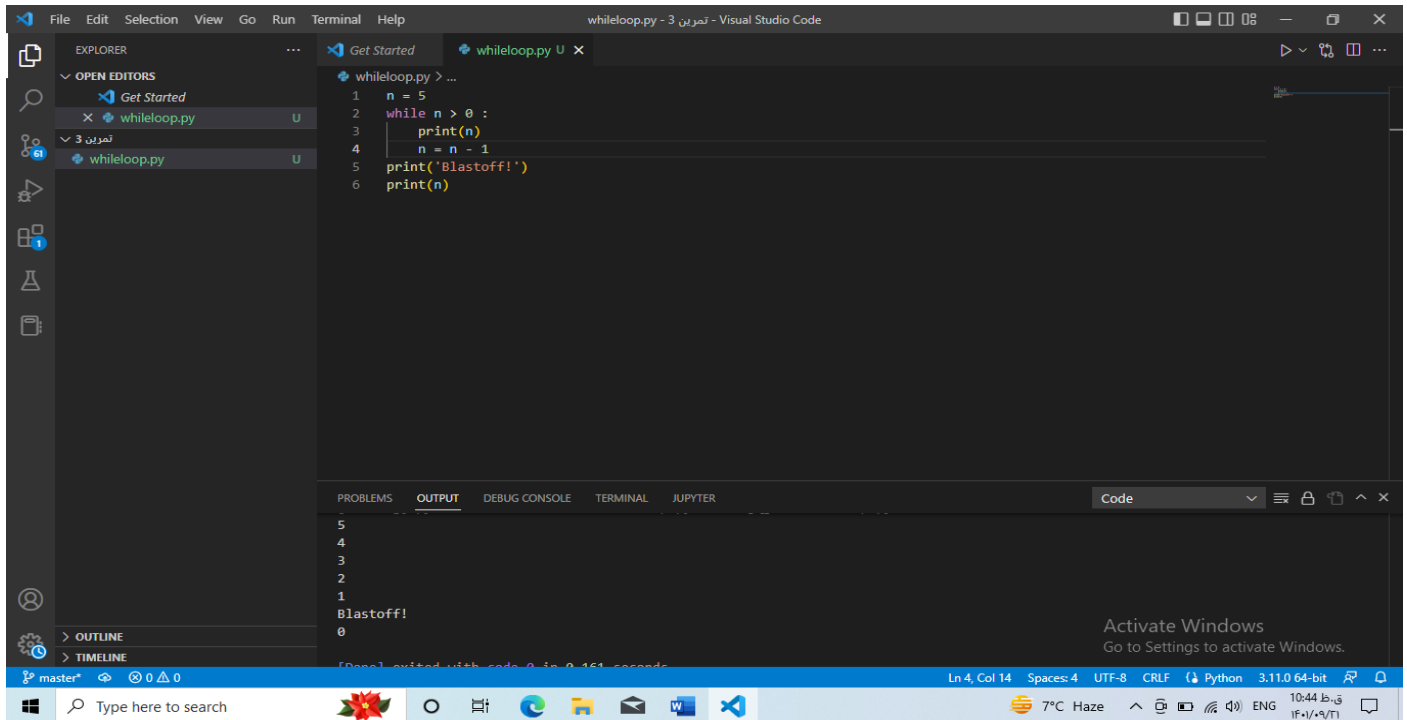
print(n)

While Loop

حلقه ها (مراحل تکراری) دارای متغیرهای تکراری هستند که هر بار از طریق یک حلقه تغییر می کنند.

اغلب این متغیرهای تکراری از طریق یک دنباله از اعداد عبور می کنند

در این کد تا زمانی که به عدد 5 برسد اعداد را از کوچک به بزرگ زیر هم پرینت می کند.



```
whileloop.py > ...
1  n = 5
2  while n > 0 :
3      print(n)
4      n = n - 1
5  print('Blastoff!')
6  print(n)
```

OUTPUT

```
5
4
3
2
1
Blastoff!
0
```

n = 5

while n > 0 :

 print('Lather')

 print('Rinse')

print('Dry off!')

Infinite loop

حلقه بینهایت در این کد حلقه پایان وجود ندارد

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'Infinite loop.py'. The main editor displays the following Python code:

```
1 n = 5
2 while n > 0 :
3     print('Lather')
4     print('Rinse')
5     print('Dry off!')
6
```

The Output panel at the bottom shows the execution results:

```
Lather
Rinse
Lather
Rinse
Lather
Rinse
Lather
Rinse
Lather
```

The status bar at the bottom indicates the file is 'Infinite loop.py' with 3 tabs, using UTF-8 encoding, CRLF line endings, and Python 3.11.0 64-bit.

Using break

while True:

line = input(' hello there ')

if line == 'done' :

break

print(line)

print('Done!')

دستور break حلقه جاری را پایان می دهد و بلافاصله بعد از حلقه به دستور می پرد

این مانند یک تست حلقه است که می تواند در هر نقطه از بدنه حلقه اتفاق بیفتد.

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'Usingbreak.py'. The main editor displays the following Python code:

```
1 while True:
2     line = input(' hello there ')
3     if line == 'done' :
4         break
5     print(line)
6 print('Done!')
```

The Output panel at the bottom shows the execution results:

```
Rinse
Lather
Rinse
Lather

[Done] exited with code=1 in 85.689 seconds

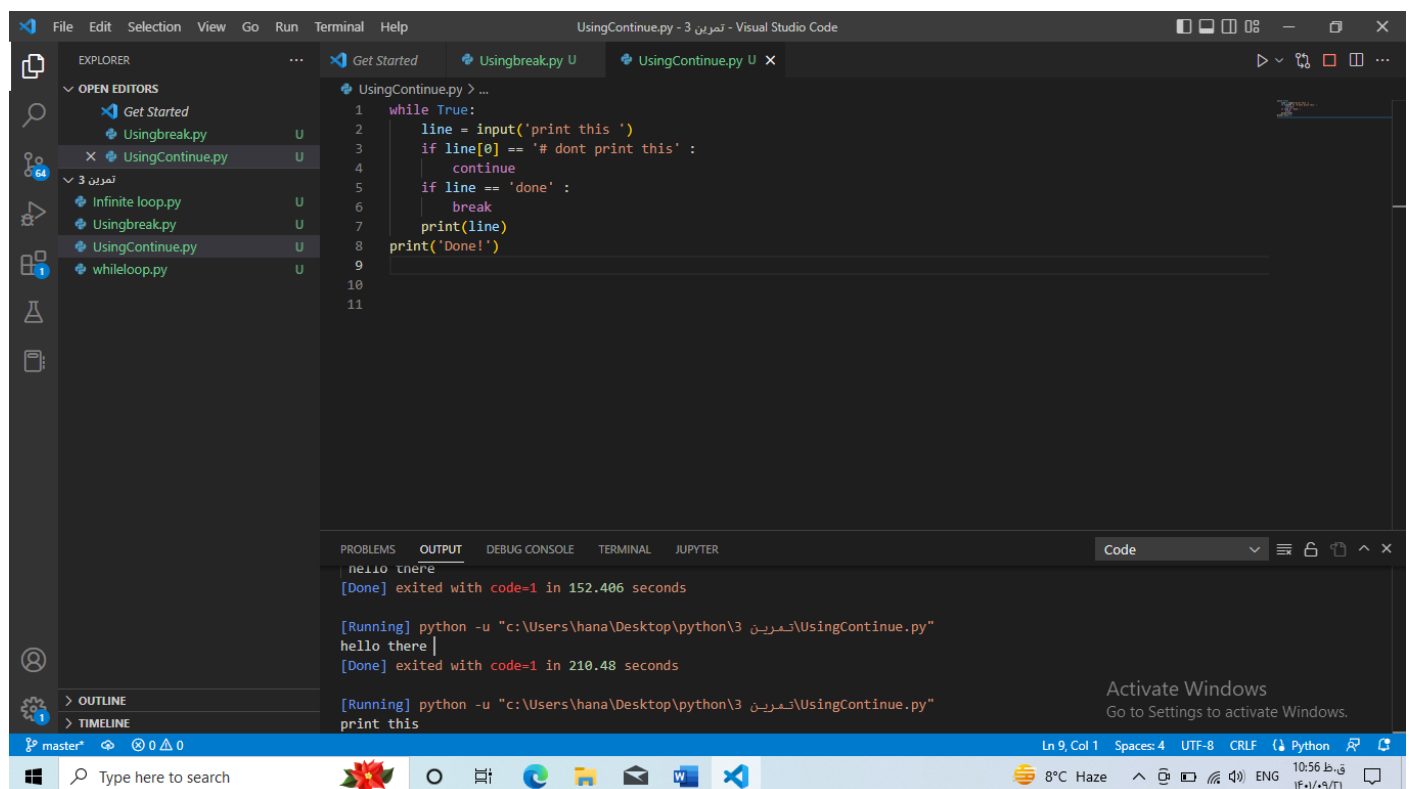
[Running] python -u "c:\Users\hana\Desktop\python\3\تمرین\Usingbreak.py"
hello there
```

The status bar at the bottom indicates the file is 'Usingbreak.py' with 3 tabs, using UTF-8 encoding, CRLF line endings, and Python 3.11.0 64-bit.

Using continue

دستور continue تکرار فعلی را پایان می دهد و به بالای حلقه می پرد و تکرار بعدی را شروع می کند.

```
while True:
    line = input('print this ')
    if line[0] == '# don't print this' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```



```
for i in [5, 4, 3, 2, 1] :
```

```
    print(i)
```

```
print('Blastoff!')
```

Definite Loop

برای حلقه های معین با اعداد پرینت تکرار می شود

```
File Edit Selection View Go Run Terminal Help
Definite loop.py - تمرین 3 - Visual Studio Code

EXPLORER
OPEN EDITORS 1 unsaved
  Get Started
  Usingbreak.py U
  UsingContinue.py U
  Definite loop.py U
تمرین 3
  Definite loop.py U
  Infinite loop.py U
  Usingbreak.py U
  UsingContinue.py U
  whileloop.py U

Definite loop.py > ...
1 for i in [5, 4, 3, 2, 1]:
2     print(i)
3     print('Blastoff!')
4

[Running] python -u "c:\Users\hana\Desktop\python\3\تمرین\Definite loop.py"
5
4
3
2
1
Blastoff!
```

Iteration variables

```
friends = ['Joseph', 'Glenn', 'Sally']
```

```
for friend in friends:
```

```
    print('Happy New Year:', friend)
```

```
print('Done!')
```

Definite Loop with Strings

برای حلقه های معین با رشته پرینت تکرار می شود.

```
File Edit Selection View Go Run Terminal Help
InterationVariables.py - تمرین 3 - Visual Studio Code

EXPLORER
OPEN EDITORS
  Get Started
  InterationVariables.py U
  Definite loop.py U
تمرین 3
  Definite loop.py U
  Infinite loop.py U
  InterationVariables.py U
  Usingbreak.py U
  UsingContinue.py U
  whileloop.py U

InterationVariables.py > ...
1 friends = ['Joseph', 'Glenn', 'Sally']
2 for friend in friends:
3     print('Happy New Year:', friend)
4     print('Done!')
5

[Running] python -u "c:\Users\hana\Desktop\python\3\تمرین\InterationVariables.py"
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!

[Done] exited with code=0 in 1.262 seconds
```

Counting in a Loop

```
zork = 0
```

```
print('Before', zork)
```

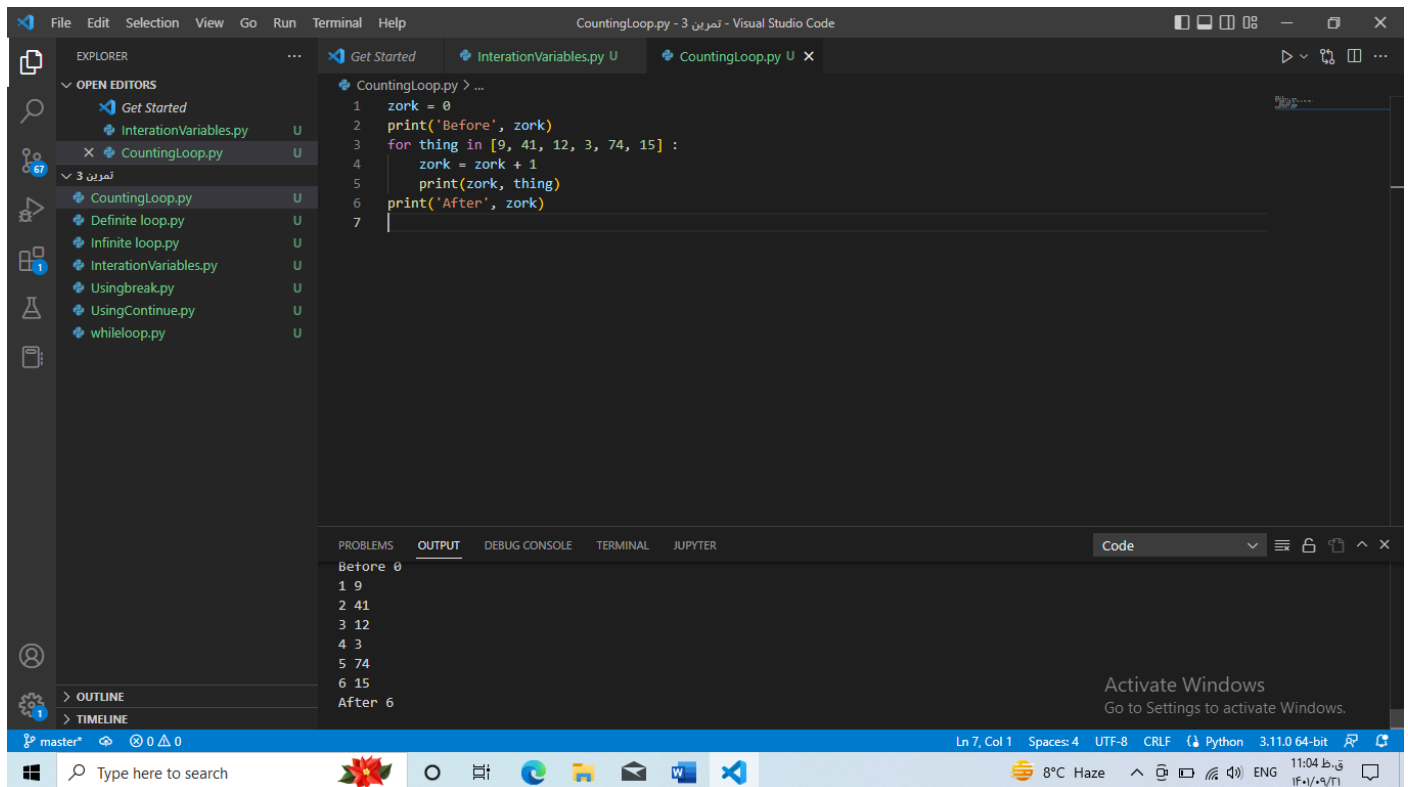
```
for thing in [9, 41, 12, 3, 74, 15] :
```

```
    zork = zork + 1
```

```
    print(zork, thing)
```

```
print('After', zork)
```

برای شمارش تعداد دفعاتی که یک حلقه را اجرا می کنیم، یک متغیر شمارنده معرفی می کنیم که از 0 شروع می شود و هر بار از طریق حلقه یک عدد به آن اضافه می کنیم.



```
1 zork = 0
2 print('Before', zork)
3 for thing in [9, 41, 12, 3, 74, 15] :
4     zork = zork + 1
5     print(zork, thing)
6 print('After', zork)
7
```

OUTPUT

```
Before 0
1 9
2 41
3 12
4 3
5 74
6 15
After 6
```

Summing in a Loop

```
zork = 0
```

```
print('Before', zork)
```

```
for thing in [9, 41, 12, 3, 74, 15] :
```

```
    zork = zork + thing
```

```
    print(zork, thing)
```

```
print('After', zork)
```

برای جمع کردن مقداری که در یک حلقه با آن مواجه می شویم، یک متغیر مجموع را معرفی می کنیم که از 0 شروع می شود و هر بار از طریق حلقه مقدار را به جمع اضافه می کنیم.

```
1 zork = 0
2 print('Before', zork)
3 for thing in [9, 41, 12, 3, 74, 15]:
4     zork = zork + thing
5     print(zork, thing)
6 print('After', zork)
7
```

Output:

```
9 9
50 41
62 12
65 3
139 74
154 15
After 154
```

```
count = 0
sum = 0
print('Before', count, sum)
for value in [9, 41, 12, 3, 74, 15]:
    count = count + 1
    sum = sum + value
    print(count, sum, value)
print('After', count, sum, sum / count)
```

Finding the Average in a Loop

یک میانگین فقط الگوهای شمارش و جمع را ترکیب می کند و وقتی حلقه انجام می شود تقسیم می شود.

```
1 count = 0
2 sum = 0
3 print('Before', count, sum)
4 for value in [9, 41, 12, 3, 74, 15]:
5     count = count + 1
6     sum = sum + value
7     print(count, sum, value)
8 print('After', count, sum, sum / count)
9
```

Output:

```
[Running] python -u "c:\Users\hana\Desktop\python\3 تمرین\AvgLoop.py"
Before 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
After 6 154 25.666666666666668
[Done] exited with code=0 in 0.236 seconds
```

```
print('Before')
```

```
for value in [9, 41, 12, 3, 74, 15] :
```

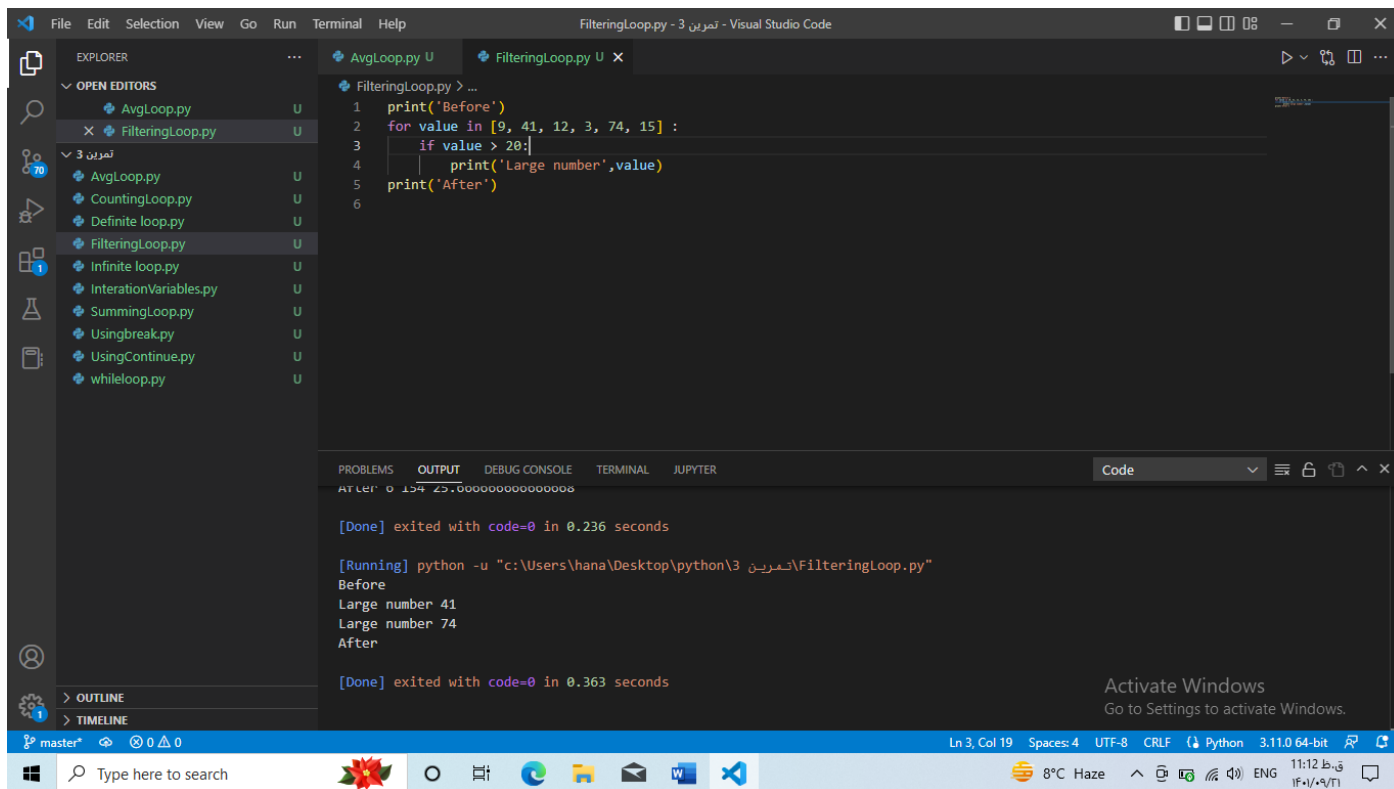
```
    if value > 20:
```

```
        print('Large number',value)
```

```
print('After')
```

Filtering in a Loop

ما از دستور if در حلقه برای گرفتن / فیلتر کردن مقادیر مورد نظر استفاده می کنیم.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left lists several Python files, including FilteringLoop.py. The main editor window displays the code for FilteringLoop.py, which is a for loop that prints 'Before', iterates over the list [9, 41, 12, 3, 74, 15], and prints 'Large number' followed by the value if it is greater than 20, and finally prints 'After'. The Output panel at the bottom shows the execution results: 'Before', 'Large number 41', 'Large number 74', and 'After'.

```
1 print('Before')
2 for value in [9, 41, 12, 3, 74, 15] :
3     if value > 20:
4         print('Large number',value)
5
6 print('After')
```

Output:

```
[Done] exited with code=0 in 0.236 seconds

[Running] python -u "c:\Users\hana\Desktop\python\3\تمرین\FilteringLoop.py"
Before
Large number 41
Large number 74
After

[Done] exited with code=0 in 0.363 seconds
```

```
smallest = None
```

```
print('Before')
```

```
for value in [9, 41, 12, 3, 74, 15] :
```

```
    if smallest is None :
```

```
        smallest = value
```

```
    elif value < smallest :
```

```
        smallest = value
```

```
    print(smallest, value)
```

```
print('After', smallest)
```

Finding the Smallest Value

پیدا کردن کوچکترین متغیر

در این دستور ابتدا برای کوچکترین مقداری در نظر نمیگیریم و از بین متغیرها به ترتیب به عنوان کوچکترین در نظر گرفته می شود و اگر در مقایسه متغیر بعدی کوچکتر بود جایگزین می شود تا در نهایت کوچکترین متغیر پیدا شود.

```
1 smallest = None
2 print('Before')
3 for value in [9, 41, 12, 3, 74, 15]:
4     if smallest is None:
5         smallest = value
6     elif value < smallest:
7         smallest = value
8     print(smallest, value)
9 print('After', smallest)
10
```

Before
9 9
9 41
9 12
3 3
3 74
3 15
After 3

[Done] exited with code=0 in 0.347 seconds

Finding the Smallest Value

`smallest_so_far = -1`

`print('Before', smallest_so_far)`

`for the_num in [9, 41, 12, 3, 74, 15]:`

`if the_num < smallest_so_far:`

`smallest_so_far = the_num`

`print(smallest_so_far, the_num)`

`print('After', smallest_so_far)`

در این دستور ابتدا برای کوچکترین مقداری در نظر می گیریم و از بین متغیر ها به ترتیب مقایسه انجام می شود تا در نهایت کوچکترین متغیر پیدا شود

```
1 smallest_so_far = -1
2 print('Before', smallest_so_far)
3 for the_num in [9, 41, 12, 3, 74, 15]:
4     if the_num < smallest_so_far:
5         smallest_so_far = the_num
6     print(smallest_so_far, the_num)
7 print('After', smallest_so_far)
8
```

Before -1
-1 9
-1 41
-1 12
-1 3
-1 74
-1 15
After -1

[Done] exited with code=0 in 0.28 seconds