

Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

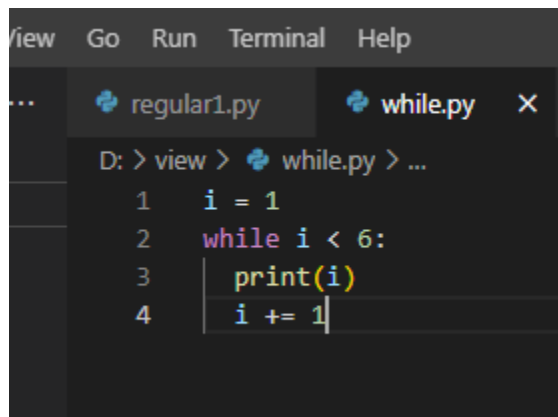
The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

Example

Print `i` as long as `i` is less than 6:

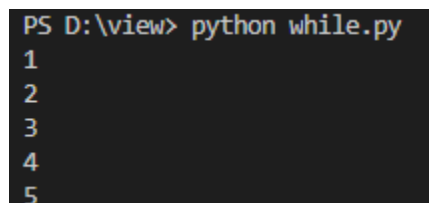
Save file name `while.loop` as shown below

A screenshot of a code editor window with a dark theme. The window title is 'view Go Run Terminal Help'. There are two tabs: 'regular1.py' and 'while.py'. The 'while.py' tab is active and shows the following code:

```
D: > view > while.py > ...
1  i = 1
2  while i < 6:
3      print(i)
4      i += 1
```

Run file in terminal for this type command `python while.py` as shown below

Output:-

A screenshot of a terminal window with a dark background. The prompt is 'PS D:\view>' and the command 'python while.py' has been executed. The output is the numbers 1 through 5, each on a new line.

```
PS D:\view> python while.py
1
2
3
4
5
```

Python For Loops

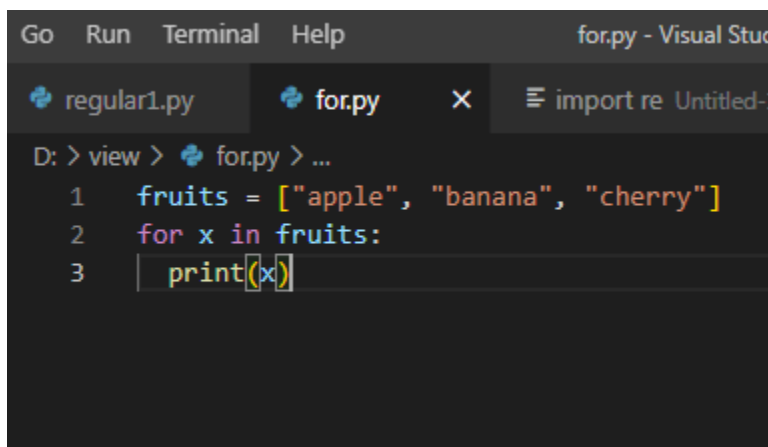
A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the **for** keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the **for** loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

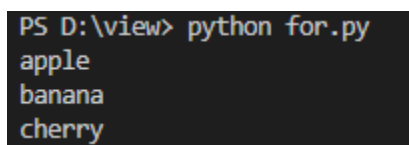
Print each fruit in a fruit list(save file name `for.py`)



```
Go Run Terminal Help for.py - Visual Stud
regular1.py for.py x import re Untitled-
D: > view > for.py > ...
1 fruits = ["apple", "banana", "cherry"]
2 for x in fruits:
3     print(x)
```

Run `for.py` file type command in terminal "`python for.py`" as shown below.

Output:-



```
PS D:\view> python for.py
apple
banana
cherry
```

The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number

Example

Using the `range()` function:

```
for x in range(6):  
    print(x)
```

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):  
    print(x)
```

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)`:

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):  
    print(x)
```

Else in For Loop

The `else` keyword in a `for` loop specifies a block of code to be executed when the loop is finished:

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

```
for x in range(6):
    print(x)
else:
    print("Finally finished!")
```

Note: The `else` block will NOT be executed if the loop is stopped by a `break` statement.

Example

Break the loop when `x` is 3, and see what happens with the `else` block:

```
for x in range(6):
    if x == 3: break
    print(x)
else:
    print("Finally finished!")
```