Python_Loops

January 15, 2025

1 Python Loops:

1.1 Presented by Abhisek Sarkar

1.1.1 as20ms091@iiserkol.ac.in

Python has **two** primitive loop commands:

- **while** loops
- **for** loops

1.1.2 While loop:

The while loop can execute a set of statements as long as a condition is true.

[3]:
```
# Print i as long as i is less than 7
i = 1 # The while loop requires relevant variables to be ready, in this example
 ↪we need to define an indexing variable, i, which we set to 1.
while i < 7:
  print(i)
  i += 1 # the loop will continue forever if we do not increment i
```

```
1
2
3
4
5
6
```

The break Statement The break statement can stop the loop even if the while condition is true:

[4]:
```
i = 1
while i < 6:
  print(i)
  if i == 3: # Exit the loop when i is 3
    break
  i += 1
```

```
1
2
3
```

The continue Statement The continue statement can stop the current iteration, and continue with the next:

[5]:
```
i = 0
while i < 6:
  i += 1
  if i == 3: #Continue to the next iteration if i is 3
    continue
  print(i)
```

```
1
2
4
5
6
```

The else Statement The else statement can run a block of code once when the condition no longer is true:

[6]:
```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

2 For Loop

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.

[9]:
```
games = ["COD", "GTA V", "Red Dead Redemption 2", "God of War", "FIFA 14"]
for x in games:
```

```
  print(x)
```

```
COD
GTA V
Red Dead Redemption 2
God of War
FIFA 14
```

Unlike `while` loop , The for loop does not require an indexing variable to set beforehand.

2.0.1 Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

[11]:
```
for x in "python": # Loop through the letters in the word "python"
  print(x)
```

```
p
y
t
h
o
n
```

2.0.2 The break Statement

With the break statement the loop can be stopped before it has looped through all the items:

[17]:
```
games = ["COD", "GTA V", "Red Dead Redemption 2", "God of War", "FIFA 14"]
for x in games:
  if x == "Red Dead Redemption 2":
    break
  print(x)
```

```
COD
GTA V
```

2.0.3 The continue Statement

The continue statement can stop the current iteration of the loop, and continue with the next:

[18]:
```
games = ["COD", "GTA V", "Red Dead Redemption 2", "God of War", "FIFA 14"]
for x in games:
  if x == "God of War":
    continue
  print(x)
```

```
COD
GTA V
```

```
Red Dead Redemption 2
FIFA 14
```

2.0.4 The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

[19]:
```
for x in [0, 1, 2]:
  pass
```

2.0.5 Else in For Loop

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

[21]:
```
# Print all numbers from 0 to 5, and print a message when the loop has ended:
for x in range(5):
  print(x)
else:
  print("The program is finished!")
```

```
0
1
2
3
4
The program is finished!
```

2.0.6 Range:

```
range(start, end, step)
```

[23]:
```
# Printing Odd numbers from 1 to 10
for x in range(1, 10, 2):
  print(x)
```

```
1
3
5
7
9
```