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.

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

Example Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Explanation

```
fruits = ["apple", "banana", "cherry"]

# Create a list called fruits and assign it three values: "apple", "banana", and "cherry"
# A list is a type of data that can store multiple items in a specific order, separated by commas

for x in fruits:

# Start a loop that will iterate over each item in the fruits list
# The word `for` means that we are going to loop for each element in the collection
# The variable `x` is used to store the current element in each iteration
# The word `in` means that we are looping over the elements in the fruits list
# The : sign means that we are ending the loop header and starting the block of statements that w
# A block of statements is a group of instructions that are indented under the same level, using
# The indentation tells the computer which statements belong to the loop and which ones do not

  print(x)

  # Print the value of x on the screen
  # The word print is a function, which is a special kind of instruction that can do something fo
  # A function has a name, followed by parentheses, and sometimes has some arguments inside the p
  # An argument is a piece of data that we give to the function to use
  # In this case, the argument is x, which means that we are giving the value of x to the print f
```

```
# End the loop and the program
# When the loop has gone through all the elements in the fruits list, it stops and the program mo
# In this case, there is no next statement, so the program ends
```

Try it yourself below in the code block

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "apple":
        continue
    print(x)
```

```
banana
cherry
```

Looping Through a String

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

Example

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

Try it Yourself below»

```
for x in "bananananananananananana":
    if x == "n":
        continue
    print(x)
```

```
b
a
a
a
a
a
a
a
a
a
a
a
```

The break Statement

With the break statement we can stop the loop before it has looped through all the items:

Example 1 of break

Exit the loop when `x` is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Try it Yourself below »

```
#positioning of break creates different outputs
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

Example 2 of break

Exit the loop when `x` is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Try it Yourself below »

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

```
apple
```

The continue Statement

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana: A continue statement is a way to skip the rest of the current iteration of a loop and move on to the next one.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Try it Yourself below »

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

```
#here the positioning of continue doesnt print, since the continue condition happens
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    continue
```

```
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.

```
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.

`range()` function takes the following three parameters.

```
range(start, end, steps)
```

Try it Yourself below »

```
for x in range(1,101):
    if x%3 == 0:
        print("fizz")
        continue
    elif x%5 == 0:
        print("buzz")
        continue
    elif x%3 == 0 and x%5 == 0:
        print("fizzbuzz")
        continue
    print(x)
```

```
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizz
16
17
fizz
19
buzz
fizz
22
23
fizz
buzz
26
fizz
28
29
fizz
31
32
fizz
34
```

```
buzz
fizz
37
38
fizz
buzz
41
fizz
43
44
fizz
46
47
fizz
49
buzz
fizz
52
53
fizz
buzz
56
fizz
```

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):

```
# Using the start parameter
for x in range(2, 6):
  print(x)
```

Try it Yourself below »

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

```
2
3
4
5
```

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)
```

Try it Yourself below »

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

```
2
5
8
11
14
17
20
23
26
29
```

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!")
```

Try it Yourself below »

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

```
0
1
2
3
4
5
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!")
```

Try it Yourself below »

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

```
0
1
2
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

```
Print each adjective for every fruit:

adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
 for y in fruits:
   print(x, y)
```

Try it Yourself below »

```
#Print each adjective for every fruit:
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
```

```

for x in adj:
 for y in fruits:
   print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

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.

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

```
# without the pass statement
for x in [0, 1, 2]:
```

```
  File "<ipython-input-3-5aa29cfa8ab9>", line 1
    for x in [0, 1, 2]:
                       ^
SyntaxError: incomplete input
```

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

Practice the following Exercises

1. Exercise 1
2. Exercise 2
3. Exercise 3
4. Exercise 4

Challenge Activity for `for` loop

Watch the accompanying video if you have to and complete the activity below.

For

Unlike a while loop, which is a conditional loop, a for loop is set to run a specific number of times. For example, you may want code to run to represent each month of a year. For that, you would want to create a loop that runs 12 times. The amount of times a loop will run is specified within a range.

The one caveat with a for loop within Python is that the last number in the range does not run. Thus, if you want a loop to run 12 times, 13 needs to be the last number in the range.

Purpose:

Upon completing this project, you will be able to create a for loop that will create a monthly increase sales goal to display.

Instructions for Completion:

1. Below the variables, replace the comment, # add for loop and logic, with a for loop that does the following: a. Uses a variable named monthlyGoal and set the range to be large enough for all 12 months. b. Within the loop, a variable named monthlySalesGoal is defined and set to the initialSalesGoal plus the total of the monthlyGoal multiplied by the multiplier. c. Within the loop, a message with the text **Your sales goal for month** and the monthly goal plus the word is and the monthlySalesGoal is displayed.
2. Run the code. You should see the following:

Your sales goal for month 1 is 20100

Your sales goal for month 2 is 20200

Your sales goal for month 3 is 20300

Your sales goal for month 4 is 20400

Your sales goal for month 5 is 20500

Your sales goal for month 6 is 20600

Your sales goal for month 7 is 20700

Your sales goal for month 8 is 20800

Your sales goal for month 9 is 20900

Your sales goal for month 10 is 21000

Your sales goal for month 11 is 21100

Your sales goal for month 12 is 21200

Good luck with your goals.

3. Close the output window.

```
initialSalesGoal = 20000
multiplier = 100

# add for loop and logic

```

Answer

Follow Along exercises

for loop

```
initialSalesGoal = 20000
multiplier = 100

# add for loop and logic

```

_22c_Break_student

```
capitalGuess = input("What is the capital of Latvia? ")
numberOfGuesses = 1

while capitalGuess != "Riga":
    numberOfGuesses = numberOfGuesses + 1
    if numberOfGuesses > 3:
        print("You guessed incorrectly three times. Game over.")
    capitalGuess = input("Guess again. ")


print("You guessed it. Riga is the capital of Latvia. It took you " + str(numberOfGue
```

22d_Continue

```
initialSalesGoal = 20000
multiplier = 100

for monthlyGoal in range(1,12):
    if monthlyGoal == 6:
        #add continue and print statement

    monthlySalesGoal = initialSalesGoal + (monthlyGoal * multiplier)
```

```

    print("Your sales goal for month " + str(monthlyGoal) + " is " + str(monthlySale

print("Good luck with your goals.")
```

22e-pass

```
annualSales = 200000
if annualSales >= 500000:
    print("Gold Customer")
elif annualSales >= 300000:
    print("Silver Customer")
elif annualSales >= 100000:
    print("Bronze Customer")
print("Thank you for your business")
```

_22f_Nested_and_Conditional_loops_student

```
initialSalesGoal = 20000
multiplier = 100
offMonth = True

for monthlyGoal in range(1,13):
    if monthlyGoal == 6:
        print("No goal for month 6")
        continue
    monthlySalesGoal = initialSalesGoal + (monthlyGoal * multiplier)

    print("Your sales goal for month " + str(monthlyGoal) + " is " + str(monthlySales

print("Good luck with your goals.")
```