Cheatsheets / **Learn Python 3**

codecademy

Loops

`break` Keyword

In a loop, the `break` keyword exits the loop immediately, regardless of the iteration number. Once `break` executes, the program will continue executing from the first line after the loop.

In this example, the output would be:

- 0
- 254
- 2
- Negative number detected!

```
numbers = [0, 254, 2, -1, 3]

for num in numbers:
  if (num < 0):
    print("Negative number detected!")
    break
  print(num)

# 0
# 254
# 2
# Negative number detected!
```

Python List Comprehension

Python list comprehensions provide a concise way for creating lists. It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses: [EXPRESSION for ITEM in LIST <if CONDITIONAL>] .

The expressions can be anything - any kind of object can go into a list.

A list comprehension always returns a list.

```
# List comprehension for the squares of
all even numbers between 0 and 9
result = [x**2 for x in range(10) if x % 2
== 0]

print(result)
# [0, 4, 16, 36, 64]
```

Python For Loop

A Python `for` loop can be used to iterate over a list of items and perform a set of actions on each item. The syntax of a `for` loop consists of assigning a temporary value to a variable on each successive iteration.
When writing a `for` loop, remember to properly indent each action, otherwise an `IndentationError` will result.

```
for <temporary variable> in <list variable>:
  <action statement>
  <action statement>

#each num in nums will be printed below
nums = [1,2,3,4,5]
for num in nums:
  print(num)
```

The Python `continue` Keyword

In Python, the `continue` keyword is used inside a loop to skip the remaining code inside the loop code block and begin the next loop iteration.

```
big_number_list = [1, 2, -1, 4, -5, 5, 2, -9]

# Print only positive numbers:
for i in big_number_list:
  if i < 0:
    continue
  print(i)
```

Python Loops with `range()`.

In Python, a `for` loop can be used to perform an action a specific number of times in a row.
The `range()` function can be used to create a list that can be used to specify the number of iterations in a `for` loop.

```
# Print the numbers 0, 1, 2:
for i in range(3):
  print(i)

# Print "WARNING" 3 times:
for i in range(3):
  print("WARNING")
```

Infinite Loop

An infinite loop is a loop that never terminates. Infinite loops result when the conditions of the loop prevent it from terminating. This could be due to a typo in the conditional statement within the loop or incorrect logic. To interrupt a Python program that is running forever, press the Ctrl and C keys together on your keyboard.

Python `while` Loops

In Python, a `while` loop will repeatedly execute a code block as long as a condition evaluates to `True`.
The condition of a `while` loop is always checked first before the block of code runs. If the condition is not met initially, then the code block will never run.

```
# This loop will only run 1 time
hungry = True
while hungry:
  print("Time to eat!")
  hungry = False

# This loop will run 5 times
i = 1
while i < 6:
  print(i)
  i = i + 1
```

Python Nested Loops

In Python, loops can be *nested* inside other loops. Nested loops can be used to access items of lists which are inside other lists. The item selected from the outer loop can be used as the list for the inner loop to iterate over.

```
groups = [["Jobs", "Gates"], ["Newton",
"Euclid"], ["Einstein", "Feynman"]]

# This outer loop will iterate over each
list in the groups list
for group in groups:
  # This inner loop will go through each
name in each list
  for name in group:
    print(name)
```

Print Share