Python Data Types

String

Series of characters or data stored as text

```
string = "Hello"
```

String Operations

```
# returns the string with all uppercase letters
string.upper()
```

```
# replaces any instance of the first string with the second in the
string
string.replace('H', 'C')
```

```
# returns the index of the first instance of the string inside the
subject string, otherwise -1
string.find('l')
```

```
# replaces any instance of the first string with the second in the
string
string.replace('H', 'C')
```

Integer

A whole number

```
integer = 12321
```

Float

A decimal number

```
decimal = 3.14
```

Boolean

Descrete value true or false

```
a = True b = False
```

Dictionary

Changeble collection of key-value pairs

```
dictionary = {'banana': 1, 12: 'apple', (0,0):'center'}
```

Dictionary Operations

```
# Access value using key
dictionary['banana']
```

```
# Get all keys in a dictionary as a list
dictionary.keys()
```

```
# Get all values in a dictionary as a list
dictionary.values()
```

List

Changeble collection of objects

```
collection = [1, 1, 3.12, False, "Hi"]
```

List Operations

Changeble collection of objects

```
# returns the length of a list
len(collection)
```

```
# Add multiple items to a list
collection.extend(["More", "Items"])
```

```
# Add a single item to a list
collection.append("Single")
```

```
# Delete the object of a list at a specified index
del(collection[2])
```

```
# Clone a list
clone = collection
```

```
# Concatenate two lists
collection_2 = ["a", "b", "c"]
collection_3 = collection + collection_2
```

```
# Calculate the sum of a list of ints or floats
number_collection = [1,2,3,4.5]
sum(number_collection)
```

Set

Unordered collection of unique objects

```
a = {100, 3.12, False, "Bye"}
b = {100, 3.12, "Welcome"}
```

Set Operations

```
# Convert a list to a set
set([1,1,2,3])
```

```
# Add an item to the set
a.add(4)
```

```
# Remove an item from a set
a.remove("Bye")
```

```
# Returns set a minus b
a.difference(b)
```

```
# Returns intersection of set a and b
a.intersection(b)
```

```
# Returns the union of set a and b
a.union(b)
```

```
a = {1, "a"}
b = {1, "a", "b", "c", 3.14, True}
# Returns True if a is a subset of b, false otherwise
a.issubset(b)
```

```
# Returns True if b is a superset of a, false otherwise
b.issuperset(a)
```

Tuple

Unchangeble collection of objects

```
tup = (1, 3.12, False, "Hi")
```

Comparison Operators

Comparison Operators compare operands and return a result of true or false

Equal

```
a == b
```

Less Than

```
a < b
```

Greater Than

```
a > b
```

Greater Than or Equal

```
a >= b
```

Less Than or Equal

```
a <= b
```

Not Equal

```
a != b
```

Python Operators

- +: Addition
- -: Subtraction
- *: Multiplication
- /: division
- //: Integer Division (Result rounded to neartest integer)

Conditional Operators

Conditional Operators evaluate the operands and produce a true of false result

And - returns true if both statement a and b are true, otherwise false

```
a and b
```

Or - returns true if either statement a or b are true, otherwise false

```
a or b
```

Not - returns the opposite of the statement

```
not a
```

Loops

For Loops

```
for x in range(x):
    # Executes loop x number of times
```

```
for x in iterable:
    # Executes loop for each object in an iterable like a string, tuple,
list, or set
```

While Loops

```
while statement:
    # Executes the loop while statement is true
```

Conditional Statements

```
if statement_1:
  # Execute of statement_1 is true
elif statement_2:
  # Execute if statement_1 is false and statement_2 is true
else:
  # Execute if all previous statements are false
```

Try/Except

```
try:
   # Code to try to execute
except a:
   # Code to execute if there is an error of type a
except b:
   # Code to execute if there is an error of type b
except:
   # Code to execute if there is any exception that have not been
handeled above
else:
   # Code to execute if there is nto exception
```

Error Types

- IndexError - When an index is out of range
- NameError - When a varaible name is not found
- SyntaxError - When there is an error with how the code is written
- ZeroDivisionError - When your code tries to divide by zero

Webscraping

```
# Import BeatifulSoup
from bs4 import BeautifulSoup
```

```
# Parse HTML stored as a string
soup = BeautifulSoup(html, 'html5lib')
```

```
# Returns formatted html
soup.prettify()
```

```
# Find the first instance of an html tag
soup.find(tag)
```

```
# Find all instances of an html tag
soup.find_all(tag)
```

Requests

```
# Import the requests library
import requests
```

```
# Send a get requests to the url with optional parameters
response = requests.get(url, parameters)
```

```
# Get the url of the response
response.url
# Get the status code of the response
response.status_code
# Get the headers of the request
response.request.headers
# Get the body of the requests
response.request.body
# Get the headers of the response
response.headers
# Get the content of the response in text
response.text
# Get the content of the response in json
response.json
```

```
# Send a post requests to the url with optional parameters
requests.post(url, parameters)
```

Working with Files

Reading a File

```
# Opens a file in read mode
file = open(file_name, "r")
# Returns the file name
file.name
# Returns the mode the file was opened in
file.mode
```

```
# Reads the contents of a file
file.read()
```

```
# Reads a certain number of characters of a file
file.read(characters)
```

```
# Read a single line of a file
file.readline()
```

```
# Read all the lines of a file and stores it in a list
file.readlines()
```

```
# Closes a file
file.close()
```

Writing to a File

```
# Opens a file in write mode
file = open(file_name, "w")
```

```
# Writes content to a file
file.write(content)
```

```
# Adds content to the end of a file
file.append(content)
```

Range

Produce an iterable sequence from 0 to y-1

```
range(y)
```

Produce an interable sequence from x to y-1

```
range(x, y)
```