

# **Python Cheat Sheet: The Basics**

## **Python Data Types**

#### String

Series of characters or data stored as text

my\_string = "Hello"

#### **String Operations**

# returns the string with all uppercase letters
my\_string.upper()

# returns the length of a string
len(my\_string)

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

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

#### Integer

A whole number

 $my_integer = 12321$ 

#### Float

A decimal number

 $my_decimal = 3.14$ 

## Boolean

Discrete value true or false

a = True b = False

### Dictionary

Changeable collection of key-value pairs

my\_dictionary = {'banana': 1, 12: 'laptop', (0,0):'center'}

## **Dictionary Operations**

# Access value using key
my\_dictionary['banana']

# Get all keys in a dictionary as a list
my\_dictionary.keys()

# Get all values in a dictionary as a list
my\_dictionary.values()

# Tuple

Unchangeable collection of objects

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

### List

Changeable collection of objects

my\_collection = [1, 1, 3.12, False, "Hi"]

#### **List Operations**

# returns the length of a list
len(my\_collection)

# Add multiple items to a list
my\_collection.extend(["More", "Items"])

# Add a single item to a list
my\_collection.append("Single")

# Delete the object of a list at a specified index
del(my\_collection[2])

# Clone a list
clone = my\_collection[:]

# Concatenate two lists
my\_collection\_2 = ["a", "b", "c"]
my\_collection\_3 = my\_collection + my\_collection\_2

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

# Check if an item is in a list, returns Boolean
item in my\_collection
# Check if an item is not in a list, returns Boolean
item not in my\_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
my\_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)

 $\begin{tabular}{ll} \# \ Returns \ intersection \ of \ set \ a \ and \ b \\ a.intersection(b) \end{tabular}$ 

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

# Returns True if a is a subset of b, false otherwise a.issubset(b)

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

#### Indexing

Accessing data from a string, list, or tuple using an element number

my\_string[element\_number]
my\_collection[element\_number]
my\_tup[element\_number]

#### Slicing

Accessing a subset of data from a string, list, or tuple using element numbers from start to stop -1

my\_string[start:stop]
my\_collection[start:stop]
my\_tup[start:stop]

## **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 the nearest 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



# **Python Cheat Sheet: The Basics**

#### 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 has not been handled
else:
    # Code to execute if there is no exception
```

# **Error Types**

- IndexError When an index is out of range
- NameError When a variable 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

## Range

Produce an iterable sequence from 0 to stop-1

range(stop)

Produce an interable sequence from start to stop-1 incrementing by step

range(start, stop, step)

### Webscraping

```
# Import BeautifulSoup
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)

#### **Functions**

```
# Create a function
def function_name(optional_parameter_1, optional_prameter_2):
    # code to execute
    return optional_output

# Calling a function
output = function_name(parameter_1, parameter_2)
```

## 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
```

### Writing to a File

file.readlines()

# Closes a file
file.close()

```
# 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)
```

# **Objects and Classes**

```
# Creating a class
class class_name:
    def __init__(self. optional_parameter_1, optional_parameter_2):
        self.attribute_1 = optional_parameter_1
        self.attribute_2 = optional_parameter_2

def method_name(self, optional_parameter_1):
    # Code to execute
    return optional_output

# Create an instance of a class
object = class_name(parameter_1, parameter_2)

# Calling an object method
object.method_name(parameter_3)
```