## Python Basics

| | |
|---|---|
| Single Line Comments | #This is a comment on a single line |
| Doc String: These appear right after a function definition | Def foo:<br>‘’’this is a docstring, this will usually contain information about what is within the function’’’ |
| Arithmetic Operations:<br>+ is addition<br>- is subtraction<br>* is multiplic-ation<br>/ is division<br>% is modulus (remainder)<br>** is exponenti-ation | Addition: result = 1 + 3<br>Subtraction: result = 1 - 3<br>Multiplication: result = 1 * 3<br>Division: result = 1 / 3<br>Modulus: result = 1 % 3<br>Exponentiation: result = 1 ** 3 |
| Plus-Equals Operator += | counter = 0<br>counter += 10<br>This is Equivalent to:<br>counter = 0<br>counter = counter + 10<br>**Note: This also works with strings |

## Python Basics (cont)

| | |
|---|---|
| Variables: Unlike in many other programming languages, python does not require you to declare a type before assigning a value to the variable | user_name = ‘Rando-mgirlll13’<br>id_number = 2345<br>user_v-erified = True<br><br>user_float = 14<br>user_float = 14.55 |
| String Concatenation | first = ‘good’<br>second = ‘morning’<br>sentence = first + last + ‘!’ |
| Function print(): This outputs information to the user in the format of text | print(‘Hello World!’)<br>print(100 +300)<br>print(14.5-55556) |

## Loops

| | |
|---|---|
| Break Keyword | In a loop, the break keyword will escape the loop<br>#Example<br>nums = [0, 2, -3, 5, 7]<br>for num in nums:<br>if (num < 0):<br>break<br>this will only run to -3 before breaking out of the loop |

## Loops (cont)

| | |
|---|---|
| List Compre-hension | This is a concise way of creating lists.<br>Syntax: list_name = [Expression for Item in List <if Confiditonal>]<br>The expressions can be anything.<br>A list comprehension will ALWAYS return a list |
| For Loops | #Example<br>nums = [1,2,3,4,5]<br>for num in nums:<br>print(num) |
| Continue Keyword | The continue Keyword is inside a loop to skip the remaining code within the loop and begin the next loop iteration |
| Loops with range() function | Using the range() function, we can have a for loop that performs an action a specific number of times<br>#Example<br>for i in range(3):<br>print(i) #Prints 0, 1, 2 |
| While Loops | A while loop will repeatedly execute a code block as long as the condition is True<br>hungry = True<br>while hungry:<br>print(‘I’m Hungry)<br>hungry = False |



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## Modules

| | |
|---|---|
| Importing Python Modules | The keyword import can be used to import python modules.<br>#Example<br>import module<br>module.function() |
| Module importing from file | To import from a file, provided it is in the same folder as the current file you are writing, you can import it as follows<br>import filename |
| Aliasing with 'as' Keyword | The 'as' keyword can give an alias to a python module or function<br>#example<br>from matplotlib import pyplot as plt<br>plt.plot(x,y) |
| Random Module | The random module offers methods that simulate non-deterministic behavior in selecting numbers from a range |

## Files

| | |
|---|---|
| Python File Object | A python file object is created with the open() function. You can associate the file object with a variable using the with and as keywords<br>with open('somefile.txt') as file_object: |

## Files (cont)

| | |
|---|---|
| Python Read Method | After having opened a file with open(), call the .read() method to return the entire file contents as a Python string. |
| Python Readline Method | If you only want to read one line, use Readline() on the file object. This will extract one single line of text at a time |
| Python Readlines Method | Instead of getting a single string of text, readlines will return a list of strings representing individual lines in the file |
| Python Write to File | By default, all files opened are only for reading. To write to a file, you must open the file with a 'w' argument, then you can use the .write() method to write the file.<br>**Note If the file already exists, all prior content will be overwritten.<br><br>Example<br>with open('text.txt', 'w') as text:<br>text.write('This is example text') |

## Files (cont)

| | |
|---|---|
| Python Append to File | Since writing to an existing file will overwrite it, to keep the original contents, we can write to a file using append instead. To appead we pass it an 'a' argument in place of a 'w' |
| Class csv.DictWriter | the csv module implements classes to read and write data in CSV format.<br>This has a class DictWrite which operates like a normal writer but will map a dictionary onto output rows. The keys of the dictionary are column names while values are actual data.<br><br>csv.DictWriter constructor takes two arguments. first is the open file handler that CSV is written to. second is 'fieldnames', this is a list of field names that the CSV is going to handle. |

## Control Flow Operations

| | |
|---|---|
| Else If Statements | #elif Statement<br>pet_type = 'fish'<br>if pet_type == 'dog':<br>print("You have a dog.")<br>elif pet_type == 'fish':<br>print("You have a fish.")<br>else:<br>print("Not Sure!") |



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## Control Flow Operations (cont)

| | |
|---|---|
| Or Operator | True or True #Evaluates to True<br>True or False #Evaluates to True<br>False or False #Evaluates to False<br>1 < 2 or 3 < 1 #Evaluates to True |
| Equal Operator | ==<br>Used to compare two values, variables or expressions to determine if they are the same.<br>if they are the same, it returns True. Otherwise, it will return False |
| Not Equals Operator | !=<br>This is used to compare two values, variables or expressions to see if they are not the same.<br>If they are not the same, it returns True. Otherwise, it will return False |
| Comparison Operators | < #Less than<br>> #Greater than<br><= #less than or equal to<br>>= #Greater than or equal to |
| And Operator | True and True #Evaluates to True<br>True and False #Evaluates to False<br>False and False #Evaluates to False<br>1 == 1 and 1 < 2 #Evaluates to True |

## Control Flow Operations (cont)

| | |
|---|---|
| Not Operator | not True #Evaluates to False<br>not False #Evaluates to True<br>not 1 > 2 #Evaluates to True |

## Functions

| | |
|---|---|
| Functions | If a task will need to be performed multiple times, it is good practice to have it done within a function.<br>In python these are defined with the 'def' keyword and then the name of your function<br>#Example<br>def my_function(x): |
| Function Parameters | Some functions require input to provide data to their code. These are known as parameters. A function can have no parameters, one parameter, or multiple parameters<br>#Examples<br>def zero_function():<br>def one_function(number):<br>def three_function(age, height, weight): |
| Calling Functions | To call a function you simply have the name of the function and the arguments it needs<br>#Example<br>zero_function()<br>one_function(44) |

## Functions (cont)

| | |
|---|---|
| Variable Scope | When it comes to the scope of variables, those not within a function are typically global variables, those within a function are local variables to that function and can be utilized in that function only.<br>If you want to get a value back from a function you can use the special keyword return |

## Dictionaries

| | |
|---|---|
| Syntax of Dictionaries in Python | example_dictionary = {"elem1": 1, "elem2": 2} |
| Dictionary Value Types | In Python, the 'Value' type can be anything, the 'key' type must be a mutable data type |
| Accessing and Writing data in a dictionary | Values can be accessed by placing the key within square brackets next to the dictionary name<br>print(example_dictionary["elem1"]<br>To write a new value to a key it is the same syntax as accessing but with an = sign and what you'd like the new value to be<br>example_dictionary["elem1"] = 3 |



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## Dictionaries (cont)

| | |
|---|---|
| Merging Dictionaries with .update() | If two dictionaries need to be combined you can use the .update() function<br>dict1 = {1 : 'one'}<br>dict2 = {2 : "two"}<br>dict1.update(dict2) {{n1}} //dict1 is now {1 : 'one", 2 : 'two"} |
| Dictionary key-value Methods | if you want to look the keys, values, or both of the dictionary, there are methods<br>.keys() will return a list of the keys<br>.values() will return a list of the values<br>.items will return a list of tuples containing the key -value pairs |
| Dictionary get() Method | The get() method will return the value of a key if it exists otherwise it will return None if no default value is given for the key |
| Dictionary .pop() Method | .pop() will remove a key from a dictionary and return that keys value |

## Classes

| | |
|---|---|
| Instantiate Python Class | class Example:<br>""This is an empty class""<br>pass |
| Python Class Variable | Class variables are defined locally within the class and outside of all methods. They have the same value for every instance of the class<br>they can be access with instance.variable or class_name.variable syntax |
| Python repr Method | The Python __repr__() method is used to tell Python what the string representation of the class should be. It only has one parameter, self, and it returns a string |
| Python Class Methods | In Python, methods are functions that are defined as part of a class. Common pract is that the first argument of any method that is part of a class is the actual object calling the method. This argument is usually called self |

## Classes (cont)

| | |
|---|---|
| Python init Method | In Python, the __init__ method is used to initalize a newly created object. It will be called every time the class is instantiated<br>class Animal:<br>def __init__ (self, voice):<br>self.voice = voice<br>cat = Animal('Meow') |
| Python type() function | the type() function will return the data type of the argument that was passed to it |
| Python dir() function | In Python, the dir() function, with no arguments, returns a list of all the attributes in current scope<br>With an object as argument, dir() will try tor return all valid object attributes |
| _main_ in Python | In Python, __main__ is an identified used to reference the current file context |

## Lists

| | |
|---|---|
| List Syntax | primes = [1,2,3,5,7,11]<br>empty_list = [] |

By **Randomgirlll13**

cheatography.com/randomgirlll13/

Published 7th June, 2023.
Last updated 7th June, 2023.


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## Lists (cont)

| | |
|---|---|
| Adding Lists Together | items = [‘cake’, ‘cookie’, ‘pie’]<br>total_items = items + [‘tart’, ‘cheesecake’]<br>print(total_items) #Result: [‘cake’, ‘cookie’, ‘pie’, ‘tart’, ‘cheesecake’] |
| Lists: Data Types | Lists can contain multiple types of data types within one list |
| List Method .append() | numbers = [11, 333, 44]<br>numbers.append(22)<br>print(numbers)<br>#Result: [11, 333, 44, 22] |
| List Indexing | In Python, Lists start at zero for the first index spot<br>#Example<br>names = [‘Lauren’, ‘Maria’, ‘Bailey’]<br>‘Maria’ is in the first index spot and ‘Lauren’ is in the zero index spot |
| Negative List Indexing | In Python, You can also access list elements using negative indices.<br>#Example<br>names = [‘Kim’, ‘Ashley’, ‘Hailey’, ‘Ginny’]<br>names[-1] # ‘Ginny’<br>names [-4] # ‘Kim’ |

## Lists (cont)

| | |
|---|---|
| List Method .remove() | This will remove the first occurrence of an element from a list in python |
| List Method .count() | This will return the number of times a certain element shows up in the list |
| Determ-ining List Length | The len() function can be used to determine the number of items found in a list<br>#Example<br>sack = [2, 4, 5, 6]<br>size = len(sack)<br>print(sack) # 4 |
| List Method .sort() | This Method will sort the contents of the list in either ascending order (numerical lists), or alphabetical order (string lists) |
| List Slicing | This allows for only a portion of the list to be returned<br>#example<br>tools = [‘hammer’, ‘ruler’, ‘pen’]<br>tools_slice = tools[1:3] # [‘ruler’, ‘pen’]<br>**Note: The original list will remain unaltered |
| Sorted() Function | This will take a list as the functions argument and will return a new sorted list without altering the original list |

## Lists (cont)

| | |
|---|---|
| List Method .insert() | This allows us to add an element to a specific index into the list |
| List Method .pop() | This allows us to remove an element from the list and also return it |

## Strings

| | |
|---|---|
| Escaping Characters | Backslashes (\) are used to escape characters in Python Strings |
| In Syntax | The in syntax is used to determine if a letter or substring exists within a string. This will return True or False<br>#Example<br>sentence = “Creating this has been a lot of work”<br>print(“work” in sentence)<br>#True |
| Indexing and Slicing Strings | Using the same notation as lists, you can index strings<br>You can also get a substring from a string using slicing, the notation is string_name[sta-rt:end] |
| Iterate Strings | To iterate through a string, utilize the for … in notation<br>#Example<br>str = “hello”<br>for c in str:<br>print(c) |



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

| Strings (cont) | |
|---|---|
| String len() function | This function can be used to determine the length of a string among other objects |
| String Concat-enation | To combine two strings, simply use the + operator |
| String Immuta-bility | In python, strings are considered immutable, meaning once it has been defined, it cannot be changed |
| String .format() | This replaces empty braces ({}) placeholders in a string with the arguments passed.<br>If keywords are specified within the placeholders, they are replaced with the corresponding named arguments<br>#Example<br>msg1 = 'Mary had {} glasses of water and John had {} glasses of juice.'<br>msg1.format(3, 2) |
| String Method .lower() | This will convert a string to all lowercase letters |
| String Method .strip() | This will remove characters from the beginning and end of a string. You can specify what characters to remove |
| String Method .title() | This will return a string in title case |

| Strings (cont) | |
|---|---|
| String Method .split() | This will split a string into a list of items based on arguments. If no arguments are passed, it uses white space, otherwise it will split based on whatever the argument passed is |
| String Method .find() | This will return the index of the first occurence of the string argument passed. If nothing is found, it will return -1 |
| String Method .replace() | This will replace the first occurrence of the first string argument with the second string argument |
| String Method .upper() | This will make the string all uppercase |
| String Method .join() | This concatenation a list of strings together with the desired delimiter |

By **Randomgirlll13**

cheatography.com/randomgirlll13/

Published 7th June, 2023.
Last updated 7th June, 2023.


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com