**List comprehensions**

Python list comprehensions provide a short and easy way to create lists.

The list comprehensions contains an expression, a **for** statement, and an optional **if** statement.

*Exercise:*

|  |
| --- |
| *Create a list:*  **myList1 = ['Michael', 'Neil', 'Kevin', 'David', 'Anthony', 'Ferdi']**  *Create a list comprehension, before we do this that’s firstly create an empty list:*  **myList2 = []**  *Next place an expression inside the empty list:*  **myList2 = [i for i in myList1]**  *The above list comprehension runs a* ***for*** *loop which goes through each item in* ***myList1****. The* ***i*** *before the* ***for*** *loop populates* ***myList2****.*  **Question**  *Why not just state* ***myList2 = myList1****?*  *To answer the above question consider the following:*  **myList2 = [i for i in myList1 if len(i) > 6]**  *The above statement only allows items which satisfy the* ***if*** *statement to be added into* ***myList2****.* |

*Exercise:*

|  |
| --- |
| *Let’s create a new list of integers (****myList3****), and a list comprehension (****myList4****) that multiplies the integers in* ***myList3*** *by the value 2.*  **myList3 = [1,2,3,4,5]**    **myList4 = [i\*2 for i in myList3]**  **print(myList4)**  *Why did I need to create* ***myList3****? Let’s try producing the list comprehension in one statement.*  **myList4 = [i\*2 for i in range(6)]**  **print(myList4)**  *Let’s add an* ***if*** *statement to* ***myList4*** *list comprehension.*  **myList4 = [i for i in range(11) if i%2 == 0]**  **print(myList4)** |

Exercise:

|  |
| --- |
| *Create a list comprehension which contains only odd numbers between 0 – 10, the result should return* ***[1, 3, 5, 7, 9]****.*  ***Result:***  ***Option 1:* print([x for x in range (1,10,2)])**  ***Option 2:* print([x\*2+1 for x in range(5)])**  ***Option 3:* print ([i for i in range(10) if i%2 == 1])** |

*Determine the output of the following statements:*

|  |  |
| --- | --- |
| **myList = []**  **for x in range(10):**  **myList.append(x\*\*2)**  **print(myList)**  ***Result:***  **[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]** | **myList = [x\*\*2 for x in range(10)]**  **print(myList)**  ***Result:***  **[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]** |

*Determine the output of the following statements:*

|  |
| --- |
| **myList = [x\*\*2 for x in range(1,20)if x%2 ==1]**  **print(myList)**  ***Result:***  **[1, 9, 25, 49, 81, 121, 169, 225, 289, 361]** |

*A closer look at the above list comprehension:*

|  |  |  |
| --- | --- | --- |
| ***Step 3*** | ***Step 1*** | ***Step 2*** |
| **[x\*\*2** | **for x in range(1,20)** | **if x%2 ==1]** |
| *Perform the calculation* ***x\*\*2, x*** *being the value passed from step 2.*  *The result of the calculation will be added to the* ***myList*** *list.* | *Generates a sequence of values from* ***1*** *to* ***20*** *(excluding* ***20****), incrementing by* ***1****.*  ***X*** *holds the value of each increment. This value is passed to step 2, one increment at a time.* | *Test the value assigned to* ***x*** *passed from step 1.*  *If the value meets the condition (an odd number), it is passed to step 3* |

*Determine the output of the following:*

|  |  |
| --- | --- |
| **a = "We are all enjoying Python programming".split()**  **print(a)** | ***Result:***  **['We', 'are', 'all', 'enjoy', 'Python', 'programming']** |
| **b = [x for x in a if x.count('y')==0]**  **print(b)** | ***Result:***  **['We', 'are', 'all', 'programming']** |
| **c = [x for x in a if x[0] == 'a']**  **print(c)** | ***Result:***  **['are', 'all']** |
| **d = [x.upper() for x in a if x[0].isupper()]**  **print(d)** | ***Result:***  **['WE', 'PYTHON']** |

**The Python Main Block**

As discussed in week one, Python is an interpreted language because Python programs are executed by an interpreter. Python is also considered a scripting language. It operates as a script of sequential statements executed top to bottom. A collection of statements can be placed within a block (function) which can be used over and over again.

It is vital all programs have a commencement point. In Python, programmers use the main block as a commencement point to execute their programs.

*The Python main block:*

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

The main block is placed at the bottom of the program. Why is this so? To answer this question that’s have a look at a basic program:

|  |  |
| --- | --- |
| **def main():**  **print('Hello, World!')** | ***The main function*** |
| **if \_\_name\_\_ == "\_\_main\_\_":**  **main()** | ***The main block*** |

When a Python script is executed the interpreter reads the script from top to bottom searching for the first non- indented non-defined function. It searches for the main block to run the program. As evident in the example above, the main block calls the **main()** function within the program.

If the main block was above the main function, the program will return the error (**NameError: name 'main' is not defined**) when run.

*Try it:*

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**def main():**

**print('Hello, World!') *Result:* NameError: name 'main' is not defined**

Why did we get this error message? Before answering this question, consider the above statement “*when a Python script is executed the interpreter reads the script from top to bottom*”.

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Back to our basic program:*

|  |  |
| --- | --- |
| **def main():**  **print('Hello, World!')** | ***The main function*** |
| **if \_\_name\_\_ == "\_\_main\_\_":**  **main()** | ***The main block*** |

Although the main bock calls the function **main()**function, there is no relationship between the string **\_\_main\_\_** assigned to the variable **\_\_name\_\_** and the function **main()**. You can create your own function name instead of **main()**, but be aware programmers worldwide understand the function **main()** as the entry point to a program.

A module (program) can either run on its own **\_\_name\_\_ = '\_\_main\_\_'**, or can be called by another module.

Consider the modules below:

**# a.py**

**def main():**

**print("A is running on its own.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**else:**

**print("A is imported") *Result:* A is running on its own.**

**# b.py**

**import a**

**def main():**

**print("B is running on its own.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**else:**

**print("B is imported") *Result:* A is imported**

***Result:* B is running on its own.**

**Working with Files**

**Write to a File**

**# Open the myText.txt text file in the existing directory, if the myText.txt file does not exist create it in the existing directory.**

**# The 'w' mode opens the file for writing.**

**myFile = open('myText.txt', 'w')**

**# The write() file method writes a string to an open file.**

**myFile.write( "Michael enjoys Python programming.\nWe all enjoy Python programming.")**

**# The close() file method closes the file.**

**myFile.close()**

**Read from a File**

**# Open the existing myText.txt file in the existing directory.**

**# The 'r' mode opens a file for reading**

**myFile = open("myText.txt", "r")**

**#The read() file method read a string from an open file.**

**myString = myFile.read()**

**print (myString)**

**myFile.close()**

*Read a given number of bytes from a file:*

**myFile = open("myText.txt", "r")**

**# Read the first 7 bytes from the file.**

**myString = myFile.read(7)**

**print (myString)**

**myFile.close()**

***Result:***

**Michael**

**Append to an Existing File**

**# Open the existing myText.txt file in the existing directory.**

**# The 'a' mode opens the file for appending (adding).**

**myFile = open("myText.txt", "a")**

**myFile.write("\nHello World, we are programming in Python.")**

**myFile.close()**

File modes can be any of the following:

* **'r'**for reading.
* **'r+'**for reading and writing.
* **'w'**for writing.
* **'a'** for appending.
* On Windows there is also a **'b'** option for binary reading and writing (**'rb'**, or **'wb'**).

**The readline() File Method**

**myFile = open("myText.txt", "r")**

**#The readline() file method reads a file in a line at a time.**

**myString = myFile.readline() # This reads the first line.**

**print (myString)**

**myFile.close()**

*How do you read the second line?*

**myFile = open("myText.txt", "r")**

**#The readline() method read a string from an open file.**

**myString = myFile.readline() # Read the first line.**

**myString = myFile.readline() # Read the second line.**

**print (myString)**

**myFile.close()**

*Read all lines using a loop:*

**myFile = open("myText.txt", "r")**

**for myString in myFile:**

**print (myString)**

**myFile.close()**

***Result:***

**Michael enjoys Python programming.**

**We all enjoy Python programming.**

**Hello World, we are programming in Python.**

**The readlines() File Method**

|  |
| --- |
| **myFile = open("myText.txt", "r")**  **# file.readlines() reads the file line by line and places it in a list**  **myString = myFile.readlines()**  **print (myString)**  **myFile.close()**  ***Result:***  **['Michael enjoys Python programming.\n', 'We all enjoy Python programming.\n', 'Hello World, we are programming in Python.']**  **myFile = open('myText.txt','r')**  **for myString in myFile.read().split('\n'):**  **print (myString)**  **myFile.close()**  ***Result:***  **Michael enjoys Python programming.**  **We all enjoy Python programming.**  **Hello World, we are programming in Python.** |

**EXTRA**

|  |  |
| --- | --- |
| **#Count total lines in a file**  **myFile = open("myText.txt", "r")**  **totalLines = 0**  **for line in myFile:**  **totalLines += 1**  **print ( (totalLines))**  or  **myFile = open("myText.txt", "r")**  **lines = myFile.readlines()**  **print (len(lines))**  **myFile.close()** |  |

**Rename a File**

**# import the os module**

**import os**

**# Rename a file from myText.txt to myText2.txt**

**os.rename( "myText.txt", "YourText.txt" )**

**Remove a File**

**import os**

**# Delete file myText2.txt**

**os.remove("myText2.txt")**

**Create a Directory**

**import os**

**# Create a directory "python" in the current directory.**

**os.mkdir("python")**