**Python**

**String**

* Text

**Bool**

* True or False

**Int**

* **Integer**

**Float**

* Decimal

**None**

* 0/none return value

**Def main()**

* Define a function
* Our function is main()
  + You would define your variables and then write the code for the function
* Main() will run the code in the file

**Type casting**

* Converting one value type to another type
* String 🡪 int
* Int 🡪 string
  + Def cast()
    - tmp\_string = 5
    - Result = 5 + int(tmp\_string)
    - Print(Result)
* Should print 10 as tmp\_string has been changed to an integer of 5
  + Print(type(tmp\_string)
    - should print ‘int’ confirming the type cast

**Type()**

* Will show the type of variable it is i.e., int, string, bool, float etc.

**Operators**

* /, \*, +, -, >= (more than), <= (less than)
* Tmp = 5
* Tmp = tmp + 1
  + Will print 11
* “Tmp += 1” is the same as “tmp = tmp + 1”

**If/Else, Elif**

* Logic flow for setting conditions
* Else: used when the first if condition isn’t matched
* Elif: introduces another if statement after the first one isn’t matched

**For/While Loops**

* A way to repeat certain sections of code for a certain amount of time or iterations

**Dictionary**

* A key and value pair
  + ‘name’ : ‘Tyler’
  + ‘Age’ : ‘23’
* Uses {}

**Lists**

* A list of items
* Uses[]

**Tuples**

* An immutable list
* Cannot be changed once its created
* Uses()

**Modules**

* Handle specific tasks like web requests
* Must be imported

**Class**

* Allows you to build a structure/blueprint for an object you want to use
* Uses initialiser or constcutor (\_\_init\_\_)

|  |  |
| --- | --- |
| **Main.py - Def, Variables, If/Else** | |
| def main():  sample\_string = 'RedTeam Nation'  sample\_int = 4444  sample\_float = 3.14  sample\_bool = True  sample\_none = None  print('sample print - {0}, {1}, {2}, {3},  {4}'.format(sample\_string, sample\_int,  sample\_float, sample\_bool, sample\_none))  if \_\_name\_\_ == '\_\_main\_\_':  main() | Sample print – RedTeam Nation, 4444, 3.14, True, None |
| 1. Defines function main() 2. Defines variables to use in function 3. Function is to print variables 4. If/else statement says if the name of the file is == to main.py (which it is), then run the function 5. Function will run and therefore print the variables | |

|  |  |
| --- | --- |
| **Cast.py – Def, Variables, If/Else, Type Casting** | |
| def cast():  tmp\_string = 5  result = 5 + int(tmp\_string)  print(result)  print(type(tmp\_string))  if \_\_name\_\_ == '\_\_cast\_\_':  cast() | 10  <type ‘int’> |
| 1. Define function with variables and code 2. Includes type casting ‘tmp\_string’ into an integer as the variable was also set to an int 3. The type of variable it is now will also be printed as well: <class ‘int’> | |

|  |  |
| --- | --- |
| **IfElse.py – If/Else, Else/If, Operators** | |
| def IfElse():  age = 23  name = 'Tyler'  male = True  if('Tyler' in name):  print('Here')  if('Tyler' == name):  print('Here')  if(male == True):  print('Is Male')  if(age >= 25):  print('Of Age')  else:  print('Not of Age')  if(age =23 and 'Tyler' == name):  print('Authorised')  if(age >=25):  print('Of Age')  elif('Tyler' == name):  print('Not of age but Authorised')  if \_\_name\_\_ == '\_\_IfElse\_\_':  IfElse() | Here  Here  Is Male  Not of Age  Authorised  Not of age but Authorised |
| 1. Variables defined, age, name and gender 2. Is statements creating conditionals, if the string ‘Tyler’ is in the name variable then print ‘Here’ – it will print 3. If the variable age is more than 25, then print ‘Of Age’ or else, print ‘Not of Age) – it will print ‘Not of Age’ as the age variable is set to 23 4. The next statements are combined. Although the person is ‘Not of Age’, if their age is 23 and their name is Tyler (specific person), then they are authorised so print ‘Authorised’ 5. Another way to write this if you didn’t want to limit it to the year that I’m 23, this year I’m 24 for we use this code: 6. If the age is less than 25 and (but) the name is ‘Tyler’, then print ‘Not of Age but Authorised’ | |

|  |  |
| --- | --- |
| **ForLoops.py – For, in, lists** | |
| def ForLoops():  tmp\_string = [1,2,3,4,5]  for char in tmp\_string:  print(char)    List = (‘A’,’B’,’C’,’D’,’E’)  for item in List:  print(item)  print(item)  print(item)  for char in tmp\_string:  for item in List:  print(char, item)  if \_\_name\_\_ == '\_\_ForLoops\_\_':  ForLoops() | 1  2  3  4  5  A  A  B  B  C  C  D  D  E  E  E  1 A  1 B  1 C  1 D  1 E  2 A  2 B  2 C  2 D  2 E  3 A  3 B  3 C  3 D  3 E  4 A  4 B  4 C  4 D  4 E  5 A  5 B  5 C  5 D  5 E |
| 1. A For loop will tell python to iterate over the variable 2. Saying “for char in tmp\_string, print(char)” will cause python to iterate over each character in the variable 3. So, the code will run for the first character, then the next character and so on 4. One by one for each item in the collection 5. Can be done with lists, dictionaries, tuples, set, strings 6. Can also be nested to iterate through multiple sets at once | |

|  |  |
| --- | --- |
| **WhileLoops.py – While Loops, If** | |
| X = 0  While X = 0  Print(X)  X = 0  While X < 10  Print(X)  X += 1  Print(‘Finished!’)  X = 0  While X < 10  X += 1  Print(X)  If X == 5  Break  Print(‘Finished!’) | 0  0  0  0  Forever iterating  1  2  3  4  5  6  7  8  9  Finished!  1  2  3  4  5  Finished! |
| 1. In the first example, because X is always 0, the print will indefinitely loop 2. In the second example parameters have been set so that after ever loop/iteration, 1 is added to the value of X and when X reaches 10, stop looping and print ‘Finished!’ 3. In the third example, break is used to break out of the loop (without changing the conditions of the loop) when X gets to 5 4. ‘Continue’ can be used in the opposite way to break | |

|  |  |
| --- | --- |
| **Dictionary.py – Dictionary key and value pairs, add, pop** | |
| Tmp\_dict = {  'Name': 'Tyler'  'Age': 23  }  Print(Tmp\_dict(Age))  If('Lewis' in Tmp\_dict):  Print('Name Found')  Else:  Print('Name Not Found')  If('Tyler' in Tmp\_dict):  Print('Tyler Found')  Tmp\_dict('gender') = 'Male'  Print(Tmp\_dict)  Tmp\_dict.pop('gender')  Print(Tmp\_dict)  For key in tmp\_dict:  Print(key)  Print(‘\nPrint Values’)  For item in tmp\_dict.values():  Print(item)  print('\nPrint Values') for key,value in Tmp\_dict.items():  print('Key: {0}\nValue: {1}'.format(key, value)) | 23  Name Not Found  Tyler Found  {Name: Tyler, Age: 23, Gender: Male}  {Name: Tyler, Age: 23}  Name, Age  Print Values  Tyler  23  Print Values  Key: Age  Value: 26 |
| 1. Setting up a dictionary with key and value pairs 2. Printing a specific key in the dictionary – print(tmp\_dict(age)) 3. Adding key/value pairs to the dictionary – tmp\_dict(‘gender’) = ‘male’ 4. Removing key/value pair form dictionary – tmp\_dict.pop(‘gender’) | |

|  |  |
| --- | --- |
| **Lists.py – Lists, Index, for, if, loops, append, remove** | |
| tmp\_list = ['Tyler', 23, 'Surrey']  print(tmp\_list[2])  name = tmp\_list[0]  print(name)  age = tmp\_list[1]  tmp\_list[1] = 24  print(age)  age = 23  print(tmp\_list[1])  print(age)  tmp\_list.append('CEO')  print(tmp\_list)  tmp\_list.remove('CEO')  print(tmp\_list)  print('\n')  for i in range(len(tmp\_list)):  print(tmp\_list[i])  for i in range(len(tmp\_list)):  if i == 1:  tmp\_list[i] = 25  print(tmp\_list) | Surrey  Tyler  23  24  23  ['Tyler', 24, 'Surrey', 'CEO']  ['Tyler', 24, 'Surrey']  Tyler  24  Surrey  ['Tyler', 24, 'Surrey']  ['Tyler', 25, 'Surrey']  ['Tyler', 25, 'Surrey'] |
| 1. Defining the list 2. Items in list are referred to by their index starting at 0: [2] 3. Items in the list can be assigned to variables so they can be called by their var 4. The variable and the list index are tied, so if you change the var and then the item in the list will change as well 5. .append and .remove allows modifications to the list 6. i index can be used to manoeuvre a through a list too | |

|  |  |
| --- | --- |
| **Tuples.py** | |
| Tmp\_tuple = (‘Tyler’, 23, ‘Surrey’)  Print(tmp\_tuple)  Tmp\_tuple(1) = 24  Print(tmp\_tuple) | (‘Tyle’r, 23, ‘Surrey’)  error |
| 1. Defining items in tuple 2. Items in tuple cannot be modified 3. They can be deleted and wiped, but not modified | |

|  |  |
| --- | --- |
| **Modules.py – modules, methods** | |
| Import urllib2  Import os  Content = urllib2.urlopen(‘https://redteamnation.com’)  Print(content.read()) | “runs module and the urlopen method” |
| 1. Import module 2. Use the module and methods attached to it 3. Anything outside of python’s standard lib | |

|  |  |
| --- | --- |
| **Class.py –** | |
|  |  |
|  | |

|  |  |
| --- | --- |
|  | |
|  |  |
|  | |

**394/m 127%**