**Immutable and Mutable Objects**

Before commencing it is important to understand the difference between immutable and mutable objects.

* An immutable object is an [object](http://en.wikipedia.org/wiki/Object_(computer_science)) whose state cannot be modified after it is created. Strings and Integers are two examples of immutable objects.

* A mutable object is an [object](http://en.wikipedia.org/wiki/Object_(computer_science)) which can be modified after it is created. A list is an example of a mutable object.

*Determine the output of the following:*

|  |  |
| --- | --- |
| **def myFunction (x):**  **return x + 1**  **a=5**  **print(myFunction(a))**  **print(a)** | ***Result:***  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **#In this example the replace method is used to replace the substring 'off' with 'on'.**  **myString = 'The switch is off.'**  **print(myString.replace('off','on'))**  **print(myString)** | ***Result:***  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **# The append() method adds an item to the end of a list.**  **myList = ['abc', 'def']**  **print(myList.append('ghi'))**  **print (myList)** | ***Result:***  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

As a general rule variables are usually classed as local in Python, which means that they exist only in the location where they are declared (ie in the function or the module). They can be classed as global (ie used everywhere) but this needs to be done explicitly.

*Determine the output of the following:*

|  |  |
| --- | --- |
| **#Script A**  **a=5**  **def myFunction(x):**  **a=a+x**    **myFunction(5)**  **print (a)** | **#Script B**  **a=5**  **def myFunction(x):**  **a=5**  **a=a+x**    **myFunction(5)**  **print (a)** |
| ***Result:***  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | ***Result:***  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |
| --- |
| **a="Variable 'a'"**  **def test():**  **b="Variable 'b'"**  **c="Variable 'c'"**    **print('Inside function',a,'-', b,'-',c)**  **test()**  **print('Outside function',a,'-', b,'-',c)** |

***Result:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **a="Variable 'a'"**  **def test():**  **global b**  **b="Variable 'b'"**    **global c**  **c="Variable 'c'"**    **print('Inside function',a,'-', b,'-',c)**  **test()**  **print('Outside function',a,'-', b,'-',c)** |

***Result:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Built-in Methods**

**capitalize()**- The **capitalize()** method capitalises the first character of the string.

|  |  |
| --- | --- |
| **#Capitalise the first character of the string.**  **myString = 'python programming'**  **print(myString.capitalize())** | ***Result:***  **Python programming** |

**title()**- The **title()** method capitalises the first character of all words in the string.

|  |  |
| --- | --- |
| **#Capitalise the first character of all words in the string.**  **myString = 'python programming'**  **print(myString.title())** | ***Result:***  **Python Programming** |

**lower()**- The **lower()** method converts all uppercase characters in the string to lowercase.

|  |  |
| --- | --- |
| **#Convert all characters in the sting to lowercase.**  **myString = 'PYTHON PROGRAMMING'**  **print(myString.lower())** | ***Result:***  **python programming** |

**upper()**- The **upper()** method converts lowercase characters in the string to uppercase.

|  |  |
| --- | --- |
| **#Convert all characters in the sting to uppercase.**  **myString = 'python programming'**  **print(myString.upper())** | ***Result:***  **PYTHON PROGRAMMING** |

**swapcase()**- The **swapcase()** method converts uppercase and lowercase characters.

|  |  |
| --- | --- |
| **#Swap the case of all characters in the string, upper to lower and vice versa.**  **myString = 'Python Programming'**  **print(myString.swapcase())** | ***Result:***  **pYTHON pROGRAMMING** |

**len()**- The **len()** method find the length of an object.

|  |  |
| --- | --- |
| **#Finds how many objects are in the list.**  **myList = ['Python', 'Programming']**  **print (len(myList))** | ***Result:***  **2** |
| **#Finds how many characters are in the string.**  **myString = 'Python Programming'**  **print (len(myString))** | ***Result:***  **18** |

**bit\_length()** – The **bit\_length()** method returns the number of bits required to represent an integer in binary.

|  |  |
| --- | --- |
| **myInt=127**  **print(myInt.bit\_length())** | ***Result:***  **7**  *Note: 7 bits are required to represent 127 in binary (1111111).* |
| **myInt=128**  **print(myInt.bit\_length())** | ***Result:***  **8**  *Note: 8 bits are required to represent 128 in binary (10000000).* |

**count()**- The **count()** method counts how many times a substring appears in the sequence (such as a string or list).

|  |  |
| --- | --- |
| **#Counts how many times the given substring ('P') appears in the string.**  **myString = 'Python Programming'**  **print(myString.count('P'))** | ***Result:***  **2** |
| **#Counts how many times the given substring ('aaa') appears in the list.**  **myList = ['aaa','bbb','ccc','aaa']**  **print(myList.count('aaa'))** | ***Result:***  **2** |

**replace()** - The **replace()** method searches and replaces a given substring within a string.

|  |  |
| --- | --- |
| **#Replace the substring 'My' with the substring 'Your' in the string.**  **myString = 'My Program'**  **print(myString.replace('My','Your'))** | ***Result:***  **Your Program** |

**split()**- The **split()** method splits a sentence into separate words. It split words separated by a given defined separator, if no separator is defined, white spaces are used.

|  |  |
| --- | --- |
| **#Split the string. As no defined separator is give, white spaces are the default separators.**  **myString = 'Python Programming'**  **print(myString.split())** | ***Result:***  **['Python', 'Programming']** |
| **# In the following example the string ‘AB’ is the defined separator.**  **myString = 'ProgrammingABisABfun'**  **print(myString.split('AB'))**  **What is happening here?**  **myString = 'Programming, it is fun!'**  **print(myString.split(','))** | ***Result:***  **['Programming', 'is', 'fun']**  ***Result:***  **['Programming', ' it is fun!']** |

*Class discussion: Look at the output and determine what is happening with the following split methods:*

|  |  |
| --- | --- |
| **myString = 'one two three four five'**  **print(myString.split(' ',2))**  ***Result:***  **['one', 'two', 'three four five']** | **myString = 'one two three four five'**  **print(myString.split(' ',3))**  ***Result:***  **['one', 'two', 'three', 'four five']** |

**join()** - The **join()** method returns a string in which the string elements of a sequence (such as a string and a list) have been joined by given separator.

|  |  |
| --- | --- |
| **#Joint every character in the sting with a colon ':' between each character.**  **myString = 'Python Programming'**  **print(':'.join(myString))** | ***Result:***  **P:y:t:h:o:n: :P:r:o:g:r:a:m:m:i:n:g** |
| **#Joint every character in the sting with a blank space between each character.**  **myString = 'Python Programming'**  **print(' '.join(myString))** | ***Result:***  **P y t h o n P r o g r a m m i n g** |
| **#Join the elements of the list into a string, each element will be separated with a blank space.**  **words = ['Programming', 'is', 'fun']**  **print(' '.join(words))** | ***Result:***  **Programming is fun** |
| **#Join the elements of the list into a string, each element will not be separated from each other.**  **words = ['Programming', 'is', 'fun']**  **print(''.join(words))** | ***Result:***  **Programmingisfun** |
| **#Join the elements of the list into a string, each element will be separated with the substring 'AAA'.**  **words = ['Programming', 'is', 'fun']**  **print('AAA'.join(words))** | ***Result:***  **ProgrammingAAAisAAAfun** |

**find()** - The **find()** method is used to find the position of a substring in a string.

|  |  |
| --- | --- |
| **#Returns the position where the first substring match ('P') begins. If no match is found it will return -1.**  **myString = 'Python Programming'**  **print(myString.find('P'))** | ***Result:***  **0** |
| **#Returns the position where the first substring match ('hon') begins. If no match is found it will return -1.**  **myString = 'Python Programming'**  **print(myString.find('hon'))** | ***Result:***  **3** |
| **#rfind() searches for the last occurrence of the given substring.**  **myString = 'Python Programming'**  **print(myString.rfind('P'))** | ***Result:***  **7** |
| **#Find the index of the substring 'P', search between indexes 5 (start index) and 11 (end index).**  **myString = 'Python Programming'**  **print(myString.find('P', 5, 11))**  **print(myString.find('P', 5, len(myString)))** | ***Result:***  **7**  **#len(myString)? What is happening here?** |

*Exercise: Explain the output of the following statement:*

|  |  |
| --- | --- |
| **myString = 'Python Python Python'**  **char = 'P'**  **start = 0**  **while True:**  **index = myString.find(char, start)**  **if index == -1:**  **break**  **print(index)**  **start = index + 1** | ***Result:***  **0**  **7**  **14** |

**strip()** – The **strip()** method removes a defined substring from the right and left side of a string, if no substring is defined, white spaces are used. You can also specify which side (right or left) of the string the defined substring is remove from using the **rstrip()** and **lstrip()** methods.

|  |  |
| --- | --- |
| **#Removes the given substring ('A') from the right and left side of the string.**  **myString = 'AAAHello World!AAA'**  **print (myString.strip('A'))** | ***Result:***  **Hello World!** |
| **#As there is no defined substring, remove white spaces from the right and left side of the string.**  **myString = ' Hello World! '**  **print (myString.strip())** | ***Result:***  **Hello World!** |
| **#Removes the given substring ('A') from the right side of the string.**  **myString = 'AAA** **Hello World!AAA'**  **print (myString.rstrip('A'))** | ***Result:***  **AAAHello World!** |
| **#Removes the given substring ('A') from the left side of the string.**  **myString = 'AAA** **Hello World!AAA'**  **print (myString.lstrip('A'))** | ***Result:***  **Hello World!AAA** |

**reverse()**- The **reverse()** method reverses objects in a list.

|  |  |
| --- | --- |
| **#Reverse the objects in the list.**  **myList=['aaa','bbb','ccc']**  **print(myList)**  **myList.reverse()**  **print(myList)** | ***Result:***  **['aaa', 'bbb', 'ccc']**  ***Result:***  **['ccc', 'bbb', 'aaa']** |

*Exercise: Try the following:*

|  |  |
| --- | --- |
| **myString = 'Hello World'**  **myList = myString.split()**  **myList.reverse()**  **myList=' '.join(myList)**  **print(myList)** | **myString = 'Hello World'**  **print (''.join(reversed(myString)))** |
| ***Result:***  **World Hello** | ***Result:***  **dlroW olleH** |

**append()**- The **append()** method adds an item to the end of a list.

|  |  |
| --- | --- |
| **#Add 'ghi' to the end of the list.**  **myList = ['abc', 123, 'def', 456]**  **print (myList)**  **myList.append('ghi')**  **print (myList)** | ***Result:***  **['abc', 123, 'def', 456]**  **['abc', 123, 'def', 456, 'ghi']** |

**extend()** – The **extend()** method adds items from another sequence to the end of specified sequence.

|  |  |
| --- | --- |
| **#Add the items from myList1 into myList2.**  **myList1 = ['abc', 123]**  **myList2 = ['def', 456]**  **myList1.extend(myList2)**  **print(myList1)** | ***Result:***  **['abc', 123, 'def', 456]** |

**insert()** - The **insert()** method inserts an item at a given index, and move the remaining items to the right.

|  |  |
| --- | --- |
| **#Insert 'Python' into the second position.**  **myList1 = ['abc', 123]**  **myList1.insert(1, 'Python')**  **print(myList1)** | ***Result:***  **['abc', 'Python', 123]** |

**remove()** – The **remove()** method searches for an item, and removes the first matching item from the list.

|  |  |
| --- | --- |
| **#Remove 'Python' from the list.**  **myList1 = ['abc', 'Python', 123]**  **myList1.remove('Python')**  **print(myList1)** | ***Result:***  **['abc', 123]** |

**pop()** – The **pop()** method removes and returns the last object from the list. You can also specify the index position of the object you want removed.

|  |  |
| --- | --- |
| **#Remove and return the last item from the list.**  **myList = ['abc', 123, 'def', 456]**  **print (myList.pop())** | ***Result:***  **456** |
| **#Remove and return the 3rd item from the list.**  **print (myList.pop(2))** | ***Result:***  **def** |
| **#Print the list.**  **print(myList)** | ***Result:***  **['abc', 123]** |

**del** - The **del** statement remove an item from a list given its index. The **del** statement can also be used to remove slices from a list or clear the entire list.

|  |  |
| --- | --- |
| **myList = [2, 3, 5, 7, 22, 1, 34, 65]**  **print(myList)** | ***Result:***  **[2, 3, 5, 7, 22, 1, 34, 65]** |
| **#Delete first item from the list.**  **del myList [0]**  **print(myList)** | ***Result:***  **[3, 5, 7, 22, 1, 34, 65]** |
| **#Delete first 3 items from the list.**  **del myList [0:3]**  **print(myList)** | ***Result:***  **[22, 1, 34, 65]** |
| **#Delete all item from the list.**  **del myList [:]**  **print(myList)** | ***Result:***  **[]** |

*Exercise: Try the following:*

|  |  |
| --- | --- |
| **#Script A**  **myList = ['xyz','mno','abc','ghi']**    **myList.sort()**  **print(myList)**  **myList.reverse()**  **print(myList)** | **#Script B**  **myList = [20, 1, 6, 2, 84, 54]**  **myList.sort()**  **print(myList)**  **myList.reverse()**  **print(myList)** |
| ***Result:***  **['abc', 'ghi', 'mno', 'xyz']**  **['xyz', 'mno', 'ghi', 'abc']** | ***Result:***  **[1, 2, 6, 20, 54, 84]**  **[84, 54, 20, 6, 2, 1]** |

**Boolean Values (True or False)**

A string in Python can be tested for truth value. The return type will be in Boolean value (True or False).

*Consider the following:*

|  |  |
| --- | --- |
| **myString = 'Python Programming'**  **print (myString.startswith('P'))**  **myString = 'Python Programming'**  **print (myString.startswith('t'))** | ***Result:***  **True**  **False** |
| **myString = 'Python Programming'**  **print (myString.endswith('ing'))**  **myString = 'Python Programming'**  **print (myString.endswith('ed'))** | ***Result:***  **True**  **False** |
| **myString = '123'**  **print (myString.isdigit())**  **myString = 'Python Programming'**  **print (myString.isdigit())** | ***Result:***  **True**  **False** |
| **myString = 'Python'**  **print(myString.isalpha())**  **myString = 'Python Programming'**  **print(myString.isalpha())** | ***Result:***  **True**  **False #False as it contains a space.** |
| **myString = 'Python123'**  **print(myString.isalnum())**  **myString = 'Python 123'**  **print(myString.isalnum())** | ***Result:***  **True**  **False #False as it contains a space.** |
| **myString = 'Python Programming'**  **print (myString.istitle())**  **myString = 'python programming'**  **print (myString.istitle())** | ***Result:***  **True**  **False** |
| **myString = 'PYTHON PROGRAMMING'**  **print (myString.isupper())**  **myString = 'python programming'**  **print (myString.isupper())** | ***Result:***  **True**  **False** |

|  |  |
| --- | --- |
| **myString = 'python programming'**  **print (myString.islower())**  **myString = 'PYTHON PROGRAMMING'**  **print (myString.islower())** | ***Result:***  **True**  **False** |
| **myString = ' ';**  **print (myString.isspace())**  **myString = 'Python Programming'**  **print (myString.isspace())** | ***Result:***  **True**  **False** |
| **myString = ''**  **if not myString:**  **print('Empty String')**  **myString = 'Python'**  **if myString:**  **print('String not Empty')** | ***Result:***  **Empty String**  **String not Empty** |