

## COM738 – Data Science Foundations – Week 2 Practical

This practical will give you the experience in the application of the techniques covered in this week's lecture. It is recommended that you complete each of these activities in a Jupyter Notebook. It is recommended that you create and save a new Jupyter Notebook for each week in case you would like to refer to a previous exercise.

Note that some of the later questions in this practical will re-use functions you create at the start of the practical. It is therefore recommended to place each answer in a new Jupyter notebook cell, allowing you to make later additions.

### Defining and Using Functions

1. Write a function that will accept two numbers and return the sum of those two numbers
2. Write a function that will accept any amount of numbers provided as individual parameters (not a list) and return the sum of those numbers. Use a for loop in the function, not a list comprehension
3. Write a function that will return True if a provided string is a palindrome, and False if it is not.
4. Write a function that will return a list of all the odd numbers from a given list. Use a for loop, not a list comprehension.

### Errors and Exceptions

5. Amend your answer to question 4 to include a try...except clause which will print an informative error message if a list element which is not a number is encountered. The error message should be: "Type Error Caught: [the built-in exception message]"
6. Amend your answer to question 3 to include a try...except clause which will print an informative error message if a string is not supplied.
7. The following code example will not execute without an error. Amend the code to catch the specific exception thrown and print the following error message: "Key not found: [name of key]"

```
def returnValue(key):  
    animals = {'cow':'moo', 'cat':'meow', 'dog':'bark'}  
    return animals[key]  
  
returnValue('sheep')
```

## Iterators

8. Write a function which uses the *range* iterator to print every multiple of 5 between a two values passed to the function. Separate each printed number by a comma.
9. Use the 'enumerate' iterator within a function to output the index and character of every other character in the following sentence "Sentences are lists of characters." The output for each character should be "Character '[character]' is at index [index]"
10. Use the 'zip' iterator to print the follow 3 lists in the following format. "[Name] lives in [Town] and is [Age] years old."

```
Names = ["Sarah", "Andrew", "Paul", "Sally", "Sinead"]
Towns = ["Larne", "Carrickfergus", "Newtownabbey", "Bangor", "Holywood"]
Ages = [22, 30, 54, 42, 26]
```

11. Use the 'map' iterator within a function to capitalise all words in the sentence that are over 5 characters long "No words in this sentence are capitalised." Note: you will need to use the following:
  - a. The string split() method to obtain individual words.
  - b. An anonymous function called by the map iterator
  - c. The enumerate iterator to get the current index.
12. Amend your solution to the previous question so that the first word in the sentence ("No") is programmatically changed to "All"
13. Write a function that will use the filter enumerator to output the square root of all multiples of 6 between two given numbers. Output this as a string in the format of "Value is [value] and square root is [square root]"
14. Identify and use the correct iterator to merge the detail from the following lists. The first index of each list corresponds to one person, the second index from each list corresponds to the next person, etc. If detail is missing, insert the string "NOT SUPPLIED". For example: "James lives in NOT SUPPLIED and is 28 years old"

```
Names = ["Sarah", "Andrew", "Paul", "Sally", "Sinead", "James", "Dave"]
Towns = ["Larne", "Carrickfergus", "Newtownabbey"]
Ages = [22, 30, 54, 42, 26, 28]
```

### List Comprehensions

15. Re-create the following code using a list comprehension:

```
numbers = [1,2,3,4,5,6,7,8,9,10]

L = []
for number in numbers:
    number = number*2
    L.append(number)
L
```

16. Use a list comprehension to output only the negative numbers from a list.

17. Use a list comprehension that will exclude any of the following product codes that begin with TECH, and will replace the product code with the string "NOT SUPPLIED" if the product code is an empty string.

```
products = {"Banana": "FOOD1", "Apple": "", "Orange": "FOOD3",
"DVD": "TECH1", "Raspberry": "", "CD": "TECH2", "Grape": "FOOD5"}
```

If written correctly, the output will be:

```
['FOOD1', 'NOT SUPPLIED', 'FOOD3', 'NOT SUPPLIED', 'FOOD5']
```

### Further Tasks

18. Write a function that stores information about a car in a dictionary. The function should always receive a manufacturer and a model name. It should then accept an arbitrary number of keyword arguments. Call the function with the required information and two other name-value pairs, such as a color or an optional feature. Your function should work for a call like this one:

```
car = make_car('Vauxhall', 'Corsa', colour='blue', spoiler=True)
```

Print the dictionary to ensure all the information was stored correctly.

19. Write a function that will output one random colour from a list each time the function is called.