Trimester 1 Assignment

Course: Principles of Data Intelligence Programming (P18719)

Portfolio of Activities

Submitted by:

Josiah Olukayode

Student ID No.: 100067107

Faculty of SESS

School of Engineering, Technology and Design

Module Team:

Dr. Scott Turner

Module Team

Declaration:

I hereby affirm that I have mentioned all my sources and no part of my assignment uses any unacknowledged material.

Table of Contents

1.0 Activities 4.2: List
1.1. Lists in Python
1.2. Characteristics of Lists
1.3. Different Types of Lists
1.4. Operations in Lists
   1.4.1 Use Cases for Variables
2.0 Task 1
   a. Creating a list of Products
   b. Task 1B: Adding elements of different data types in a List
   c. Task 1C: Negative Indexing and Range
   d. Applying for Loop
2.1 Task 2: Tuples
2.2 Task 3: Set
   a. Other Operations in Set
      i. add() method
      ii. update() method
      iii. remove() method
      iv. union() method
      v. pop() method
2.3 Loop
2.4 Task 4: Dictionary
   2.4.1 Task 4.1: Cleaning of dictionary code
   2.4.2 Task (4.2) What does dictionary do?
   2.4.3 Task (4.3) What you can do with a dictionary?
   2.4.4 Characteristics of Dictionary
2.5 Task 5a: Modelling the List of Canterbury Campuses in Kent and Applying Some of the Methods in Tasks 1 to 5
   2.5.1 Creating the List of Canterbury Campuses as a List
   2.5.2 Applying for loop() to list
   2.5.3 Applying Negative index
   2.5.3 Adding strings to make a statement to meet up with the request to recommend one of the universities for data science
2.6 TASK 5b: Creating a System/Examples demonstrating Lists, Tuples, Sets and Dictionaries
   2.6.1 Modelling the list of top 5 cheapest camp sites in Kent
   2.6.2 SETS: Modelling product availability in two different named Store using Sets data type Storage
   2.6.3 Combining List and Set
   2.6.4 Modelling cheapest camp site in Kent using Dictionary data type Storage
3.0 Activity: Introduction of Activity 5.2
3.1 Task 1: Explanation on Creating a New Jupyter Notebook to refine Activity 5.2
   3.1.1 Task 1a: Creating a New Jupyter Notebook to refine Activity 5.2
   3.1.2 Task 1b: Making a Tutorial to Explain the Matplotlib code for Visualization
   3.1.4 Task 2: Explanation of Matplotlib code and what it does.
   3.1.5 Task 3: Refining the notebook on the task 1 of Activity 5.2
3.2 Task 2: Replacing the lines Marked # with Lines of Code not Marked with #
   3.2.1: Experimenting with different parameters in the ' ' (i.e Character Strings) Sections of the Code
   3.2.2: Adding Axes, Labels and Titles to the Plots
   3.2.3: Experimenting with different styles of line graphs
3.3 Task 4: Numpy
   3.3.1 Cleaning of the code
   3.3.2 Explaining the Numpy Code
   3.3.3 Applying Task 3 Skills to Enhance Histogram Plots
   3.3.4 Extension of the Codes not Discussed in the Class
4.0 Task 1: Introduction of Activity 8
4.1 Task 2: Specifying API, Importing the required Library and specifying Location

4.2 Task 3: Breaking of code into chunks and explanation
4.2.1 Task 4: Getting Current Weather Data for Gravesend
4.2.2 Task 5: Getting Current Weather Data for Gillingham
4.2.3 Task 6: Getting Current Weather Data for Essex
4.2.4 Task 7: Getting Current Weather Data for Canterbury
4.2.5 Task 8: Getting Current Weather Data for Stoke
4..2.6 Task 9: Getting Current Weather Data for London
4.3 TASK 2, QUESTION 3: Combining Current Weather API Data with Data from Other API
4.4 TASK 2, QUESTION 4: Combining Current weather data output from 6 different cities in a Tabular and Graphical forms.
4.4.1: Converting the current data into pandas dataframe
4.4.2 Getting the Metadata and some statiscal information about the current weather data dataframe
4.4.3 Plotting of the current weather data
5.0 Introduction of Activity 9
5.1 Other Tasks
5.1.1 Downloading of the file
5.1.2 Unzipping of the File
5.1.3 Uploading of the File to Jupyter Notebook
5.2 What is VADER?
5.3 What does VADER do?
5.4 Explain the Different Techniques/ Approaches of VADER
5.5 Explanation of different symbols, operators and Methods
5.6 Exploration of some pandas code on tweet_activity.csv
6.0 References

Portfolio of Activities

January 10, 2023

ACTIVITIES 4.2: List

1.1. Lists in Python:

It is one of the four built-in collection data types in the Python programming language. It is the first storage method which is "Ordered" and "Changeable". It is a sequence of items stored in a single object (Scott, 2022).

List allows storing of elements or strings in an order having positions starting from 0. Other collection of data objects are Tuple, Set and Dictionary. Each of these collection data types have different charactersitics, qualities and usage. Lists are used to store multiple items in a variable, and they are different from an Array in that, while the latter can contain objects of a single type, the former can contain a mixture of objects. (W3 Schools, 2022).

Lists are separated with a comma , and wrapped in a square bracket []

1.2. Characteristcis of Lists

(a). Ordered: items are arranged in a sequence i.e Items have defined position which cannot change. When a new item is added to the list, it takes the last position.

(b) Changeable: They can be updated (in other words, it is mutable or alterable. It can be added to or remove from after it has been created);

(c) Allow duplicate values: Because items in a list are indexed, it can accommodate same or different types of data with the same value repeated severally.

(d) Indexed – first item [0], second [1], etc

(e) Lists are dynamic and can have different types

(f) Looping: A list can be iterated by usinf for - in loop

(g) It can accommodate different data types: e. g Strings, Integers and boolean values.

(h) Lists can be nested to arbitrary depth

1.3. Different Types of Lists: Lists can be created from several types of collections

(a) Create lists from string, tuple, and lists
(b) Create lists from set and dictionary
(c) Create a list from an iterator object

1.4. Operations in Lists: There are different operations that can be performed in list by using different functions which are:

(a) Creating a list: a list is created by specifying a variable and listing element in square bracket []
(b) append()
(c) extend()
(d) insert()
(e) remove()
(f) pop()
(g) slice or range()
(h) reverse()
(i) min()
(j) max()
(k) concatenate()
(l) count()
(m) multiply()
(n) sort()
(o) index()
(p) clear()
(q) len()
(r) type()

Variable: A variable is like a tag or container that is used to hold a series of character

a. Why do we need them in python:Variables are needed in python because:

(i) Python variable allows programmers to store elements of different data types.
(ii) They hold references to objects
(iii) variables need not be declared or defined in advance, as is the case in many other programming languages. To create a variable, just assign it a value and then start using it. Real Python (https://realpython.com/python-variables/)

1.5 Use Cases: Below are some of the examples use cases for variables:

[24]:
```
#Assigning variable to an integer

n = 300
print(n)

#Python also allows chained assignment, which makes it possible to assign the␣
 ↪same value to several
#variables simultaneously:

x=y=z=790
print(x,y,z)

#Variable can be reassigned: Variable can be reassigned from an intger to a␣
 ↪string

var = 250
print(var)
```

```python
#Now reassigning the sames variable to a string

var = ("The variable 'var' that was previously assigned to the integer "+␣
↪str(var)+ " is now a string")
print(var)

#Another variable use case

name = "Phill Eagle"
age = 56
dislikes = "dishonesty"
print("My name is " + name + ", I am "+str(age)+"years old. I hate␣
↪"+(dislikes)+".")
```

```
300
790 790 790
250
The variable 'var' that was previously assigned to the integer 250 is now a
string
My name is Phill Eagle, I am 56years old. I hate dishonesty.
```

Create a list

a. What is a list: The list is a serial data sturcture in python. It can contain a group of characters which can be of different data types.

For exmple we can have a list of food items held in a variable named 'products', we can have a list of 'clothes type' held in a variable tagged "clothes_types'.

b. Example of a list in a variable

In the example below, a variable named "products" was created which icncludes items of products names. This variable was used to store a list containing food items.

products = ["flour","apple","punnet of raspberries","bread", "apple"] decimal_number = [2.3, 4.4, 5.3.2.1, 8.0,.5.6,7.8]

c. Why do we need a list? List in python are useful for:

(i) Storing a group of related items: Lists can be used to store a group of items that are related to each other. For example, the names of students in a class or the items in a shopping list.

(ii) Performing operations on a group of items: Lists support various operations, such as appending new items, inserting items at a specific position, removing items, and sorting items. These operations van be used to manipulate the items in a list.

(iii) Iterating over a group of items: Lists are iterable, which means that a for loop to can be used to iterate over the items in a list and perform a specific operation on each item.

(iv) Storing data in a structured way: Lists allow storing data in a structured way, with each item in the list having a specific position (called an index). This can make it easier to access and manipulate specific items in the list. CS with VK 2022.

d. Use Case: Some of the use-cases where we generally use lists are,

List of active players in the game. List of items in the shopping cart. List of the running processes in the system. List of messages received by the server. etc.

TASK 1.

In Class/Activity 4.2 Lists:

This In-class activity aims at practically demonstrating how to create lists types stored in variables and carrying out the different operations associated with lists.

These is demonstrated by carrying out Tasks 1 to 4 as discussed in the class.

Task 5 is to make another system/example that combines lists, tuples, sets and dictionaries. It also includes using each of the methods mention above to carry out several operations in python.

2.0 TASK 1A:

a. Creating a list of Products

Steps:

(i.) A variable named products was created with the list of the names of items (which are of string data type and can be in either a single or double inverted comma) in a square [] bracket, separated by comma; (ii). Use the print() function to print out the list (iii). Use the len() to know the numbers of items in the list

In addition to the above, the type method type() to know the data structure of the products variable

[25]:
```python
products = ["flour","apple","punnet of raspberries","bread", "apple"]
print(products)
print(len(products))
print(type(products))
```

```
['flour', 'apple', 'punnet of raspberries', 'bread', 'apple']
5
<class 'list'>
```

b.

Task 1B: ADDING ELEMENTS OF DIFFERENT DATA TYPES IN A LIST

This step is used to demonstrate the ability of list to contain values of different data types. The variable product_detail contain several items of different data types ranging from strings, integers and floats.

a. The variable product-detail is storing the details of the product flour, the price, the aisle number where it can be found within the supermarket, the storeroom where it can be found, and quantity in stock.
b. The index position or number for each of the products were specified
c. The function ‘str’ was used to convert the price which is of data type integer to data type string.

d. The corresponding statement was printed out.

```
[26]: product_detail = ["flour", 3.59,"aisle 1","storeroom aisle A", 34]
print(product_detail)
print("item: "+product_detail[0]+" costs: £"+str(product_detail[1])+" can be␣
 ↪found at "+product_detail[2])
print("item: "+product_detail[0]+" costs: £",product_detail[1]," can be found␣
 ↪at "+product_detail[2])
print("item: "+product_detail[0]+" costs: £"+str(product_detail[1])+" can be␣
 ↪found at "+product_detail[3]+
      " there are "+ str(product_detail[4])+" in stock")
```

```
['flour', 3.59, 'aisle 1', 'storeroom aisle A', 34]
item: flour costs: £3.59 can be found at aisle 1
item: flour costs: £ 3.59  can be found at aisle 1
item: flour costs: £3.59 can be found at storeroom aisle A there are 34 in stock
```

c.

TASK 1C: NEGATIVE INDEXING AND RANGE

a. Negative Indexing: Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on. This step is used to demonstrate the ability of list allowing negative indexing i.e printing from the reverse order. This can be achieved by specificying the negative index position or number of each of the items/elements in the product variable. e.g counting from the last item, the items or elements 'bread' and 'punnet of raspberries' occupies the index number [-2] and [-3] respectively on the list of items in the product variable.

```
[27]: products = ["flour","apple","punnet of raspberries","bread", "apple"]
print(products[-2])
print(products[-3])
negative_indexing = (("Based on negative indexing, the items "+ products[-2] +␣
 ↪" and " + products[-3] +
                      " are elements in the product variable, they occupy␣
 ↪second last and third last positions respectively."))
print(negative_indexing)
```

```
bread
punnet of raspberries
Based on negative indexing, the items bread and punnet of raspberries are
elements in the product variable, they occupy second last and third last
positions respectively.
```

b. Range: This is called Slicing i.e accessing tuple element (s) using slicing. It's used to access an element or range of element in a tuple by using the slicing operator colon i.e :

For example:

(i) print(products[2:4) will return a list with 2 elements from index 2 to index 3 only
(ii) print(products[:4) will return a list with 4 elements from index 0 to index 3 only

(iii) print(products[2:) will return a list with 3 elements from index 2 to every other elements in the list.

```
[28]: products = ["flour","apple","punnet of raspberries","bread", "apple"]
      print(products[2:4])
      print(products[:4])
      print(products[2:])
```

```
['punnet of raspberries', 'bread']
['flour', 'apple', 'punnet of raspberries', 'bread']
['punnet of raspberries', 'bread', 'apple']
```

2.1 TASK 2: TUPLES

TUPLES: Are multiple items stored in a single variable, they are Ordered, Indexed and Immutable (Unchangable) i.e elements or items cannot be changed, added or removed once created however, this can be done by making the tuple a list and converting back to tuple. List uses square bracket [] while Tuple uses (). Scott (2022).

product_detail = (“flour”, 3.59,“aisle 1”,“storeroom aisle A”, 34) print(product_detail)

a. Converting above code into Tuple:

In converting the above code i.e product_detail variable, we must keep all items in brackets() with each element separated by comma and inside a double or single quotation or inverted comma.

For example, converting the product_detail variable list into tuple, we have the resulting code below:

```
[29]: product_detail = ("flour", 3.59,"aisle 1","storeroom aisle A", 34)
      print(product_detail)
```

```
('flour', 3.59, 'aisle 1', 'storeroom aisle A', 34)
```

```
[30]: product_detail[1]=3.784
      print(product_detail)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [30], in <cell line: 1>()
----> 1 product_detail[1]=3.784
      2 print(product_detail)

TypeError: 'tuple' object does not support item assignment
```

Answer: The code in line 22 above did not work out because in Tuple, items are unchangeable after it has been created which is the case in the above example. The item price 3.59 was replaced with 3.784.

2.2 TASK 3: SET

Python sets are identical to mathematical sets – they are unordered but have unique elements. They also have identical operations which are union, intersection, disjoint, subset, etc. Sets are useful only when we want to perform the mentioned operations on our datasets.

Example:

i. The code below represents a python set of string.

ii. It has duplicates elements which set does not allow. So, only one apple will be returned in the list because it does not allow duplicates.

iii. Prinitng the length of our element will return only 4 becasue it does not recognise duplicates.
iv. Set cannot have mutable items. The code below explains the above statement:

[31]:
```
products = {"flour","apple","punnet of raspberries","bread", "apple"}
print(products)
print(len(products))
```

```
{'bread', 'apple', 'punnet of raspberries', 'flour'}
4
```

3b. For further explanation, the code below is another example with elements of data type integers. It has duplicate element 2 but the output will only return only one element 2.

[32]:
```
my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90}
print(my_set1)
print(len(my_set1))
```

```
{1, 2, 3, 4, 5, 7, 90}
7
```

a. OTHER OPERATIONS IN SET

b. add(): is a method in python that is used to add a single element to a set

ii. update(): is a method used to add multpile elements to a set i.e it can add a list to a set
iii. remove()
iv. union(): used to combine two different sets together returning all items from both sets, duplicates are excluded.

i. ADD

[33]:
```
products = {"flour","apple","punnet of raspberries","bread", "apple"}
print(len(products))
print(products)
```

```
4
{'bread', 'apple', 'punnet of raspberries', 'flour'}
```

i. add(): It's a method in python that is used to add a given element to a set. In the example below after returning our original list, we added a new element 'banabas' which return a new set with the item added.It can add only 1 item per time. Applying it to strings

```
[34]: products = {"flour","apple","punnet of raspberries","bread", "apple"}
      products.add("bananas")
      print(len(products))
      print(products)
```

```
5
{'punnet of raspberries', 'bananas', 'bread', 'apple', 'flour'}
```

Applying it to integers with only element added

```
[35]: my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90}
      my_set1.add(60)
      print(my_set1)
```

```
{1, 2, 3, 4, 5, 7, 90, 60}
```

Applying it to integers with several items added

```
[127]: my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90}
       my_set1.add([23,44,56,87])
       print(my_set1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [127], in <cell line: 2>()
      1 my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90}
----> 2 my_set1.add([23,44,56,87])
      3 print(my_set1)

TypeError: unhashable type: 'list'
```

Several items cannot be added to a set using add() method, the above code returned an error.

ii. update(): Is a method used to overcome the limitation of the add() method. It's used to add several elements to a python set. It has advantage over add() becasue it can be used to add several elements.

```
[128]: my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90, 2.3}
       my_set1.update([23,44,56,87])
       print(my_set1)
```

```
{1, 2, 3, 4, 5, 2.3, 7, 44, 87, 23, 56, 90}
```

(iii) pop(): Is a method which removes random element from a set (W3 Schools, 2022)

```
[129]: products = {"flour","apple","punnet of raspberries","bread", "apple"}
       products.pop()
       print(len(products))
       print(products)
```

```
3
{'apple', 'punnet of raspberries', 'flour'}
```

Another example to demostrate how pop() works is applying it to print names of week days.

[130]:
```
week_days = {"monday", "tuesday", "wednesday", "thursday", "friday",␣
  ↪"saturday", "sunday"}
week_days.pop()
print(week_days)
```

```
{'monday', 'thursday', 'sunday', 'tuesday', 'wednesday', 'friday'}
```

iii. remove(): It's a method used to remove a specified item in a python set e.g we can specify to remove the 'monday' from the variable week_days

[131]:
```
week_days = {"monday", "tuesday", "wednesday", "thursday", "friday",␣
  ↪"saturday", "sunday"}
week_days.remove("monday")
print(week_days)
```

```
{'saturday', 'thursday', 'sunday', 'tuesday', 'wednesday', 'friday'}
```

iv. Union: applying union to the piece of set below will combine them together while removing dupicates

[132]:
```
my_set1 = {1, 2, 3, 4, 3, 2, 5, 2, 7, 90, 2.3}
my_set2 = {2.3, 3.5, 4.5, 6.6}
print(len(my_set1))
print(len(my_set2))
```

```
8
4
```

Based on the number of items in variables my_set1 (8) and my_set2 (4), one will expect the union of the two variables to have ouput of 12 however, it will return 11 items becasue set does not permit duplicates. In this case item 2.3 is a duplicate common to both sets. This is described in the piece of code below

[133]:
```
combine = my_set1.union(my_set2)
print(combine)
print(len(combine))
```

```
{1, 2, 3, 4, 5, 2.3, 7, 3.5, 4.5, 6.6, 90}
11
```

3d. Adding List and set (float data type)

[134]:
```
my_set2 = {2.3, 3.5, 4.5, 6.6}
my_set2.update([44, 25, 33, 100])
print(my_set2)
```

```
{33, 2.3, 3.5, 4.5, 100, 6.6, 44, 25}
```

2.3

LOOP: A python for loop is used to iterate over a set. It goes over each of the item in a set and return each of the items in their order. Loop takes into cognisance integer first, followed by float then strings.

```
[135]: product_detail = {"flour", 3.59,"aisle 1","storeroom aisle A", 34}
       print(product_detail)
       for loop in product_detail:
           print(loop)
```

```
{34, 3.59, 'aisle 1', 'storeroom aisle A', 'flour'}
34
3.59
aisle 1
storeroom aisle A
flour
```

2.4

TASK 4: DICTIONARY

Dictionary is a python container that stores mappings of unique keys to values. It is unordered and mutable/changeable. They are written with curly brackets {}, with key-value pairs separated by commas (,) and a colon (:) separating each key from its value. Amanda, 2020.

2.4.1 Task 4.1: Cleaning of dictionary code

Based on the class task, below is a dictionary code to display this.

```
[136]: product_detail = {"product":"flour","price":3.59,"store location":"aisle␣
         ↪1","storeroom location":"storeroom aisle A","no. in stock":34}
       print(type(product_detail))
       print(product_detail)
       print(product_detail["product"])
       print(product_detail["price"])
```

```
<class 'dict'>
{'product': 'flour', 'price': 3.59, 'store location': 'aisle 1', 'storeroom
location': 'storeroom aisle A', 'no. in stock': 34}
flour
3.59
```

2.4.2 Task (4.2) What does dictionary do?

Python dictionary stores elements in key-value pairs. Keys are unique identifiers that are associated with each value. This data structure is best when considering storing several pieces of information about an element.

2.4.3

Task (4.3) What you can do with a dictionary?

a. Dictionary permits storing elements in key-value pairs

b. Elements are indexable, so we can call for a value by specifying its key.
c. It gives the best of both sets and lists.
d. Dictionaries can also be nested so that the value that a key maps to is a dictionary rather than a number or a string. They map keys, which can be any immutable type, to values, which can be any type (heterogeneous), just like the elements of a list or tuple.

2.4.4 Characteristics of Dictionary

i. Elements are unordered therefore they cannot be indexed. An item or a value cannot be called for by specifying the index position.

2.5a

MODELLING THE LIST OF CANTERBURY CAMPUSES IN KENT AND APPLYING SOME OF THE MODELS IN TASKS 1 TO 4:

2.5.1

Considering simulating a request by International students for the list of Canterbury Campuses in Kent. The output must contain the name of the campuses, the addresses, the post code and the region which they are located.

[137]:
```
canterburycampuses_kent = ["Canterbury Christ Church University, Northwood␣
 ↪Road, CT10 2WA, Broadstairs, Kent, South East Region",
                           "Canterbury Christ Church University, North Holmes␣
 ↪Road, CT1 1QU, Canterbury, Kent, South East Region",
                  "Canterbury Christ Church University, Broomhill Road, TN3␣
 ↪0TG, Tunbridge Wells, Kent, South East Region",
                "Canterbury Christ Church University, 30 Pembroke Court, Rowan␣
 ↪Williams Court, ME4 4UF, Chatham, Kent, South East Region"]
print(len(canterburycampuses_kent))
print(canterburycampuses_kent)
print(type(canterburycampuses_kent))
```

```
4
['Canterbury Christ Church University, Northwood Road, CT10 2WA, Broadstairs,
Kent, South East Region', 'Canterbury Christ Church University, North Holmes
Road, CT1 1QU, Canterbury, Kent, South East Region', 'Canterbury Christ Church
University, Broomhill Road, TN3 0TG, Tunbridge Wells, Kent, South East Region',
'Canterbury Christ Church University, 30 Pembroke Court, Rowan Williams Court,
ME4 4UF, Chatham, Kent, South East Region']
<class 'list'>
```

2.5.2: Applying for loop() to list

Looking at the above list output, it is not easily readable by the students. This can be overcome by applying the for loop to have the list created to make it readable hence another variable was created to allow for the readablitiy of the list output.

[138]:
```
list_canterburycampuses2 = ["Canterbury Christ Church University, Northwood␣
 ↪Road, CT10 2WA, Broadstairs, Kent, South East Region",
                           "Canterbury Christ Church University, North Holmes␣
↪Road, CT1 1QU, Canterbury, Kent, South East Region",
                  "Canterbury Christ Church University, Broomhill Road, TN3␣
↪0TG, Tunbridge Wells, Kent, South East Region",
                "Canterbury Christ Church University, 30 Pembroke Court, Rowan␣
↪Williams Court, ME4 4UF, Chatham, Kent, South East Region"]
print(type(list_canterburycampuses2))
for loop in list_canterburycampuses2:
    print(loop)
```

```
<class 'list'>
Canterbury Christ Church University, Northwood Road, CT10 2WA, Broadstairs,
Kent, South East Region
Canterbury Christ Church University, North Holmes Road, CT1 1QU, Canterbury,
Kent, South East Region
Canterbury Christ Church University, Broomhill Road, TN3 0TG, Tunbridge Wells,
Kent, South East Region
Canterbury Christ Church University, 30 Pembroke Court, Rowan Williams Court,
ME4 4UF, Chatham, Kent, South East Region
```

2.5.3 Applying Negative index

[139]:
```
list_canterburycampuses2 = ["Canterbury Christ Church University, Northwood␣
↪Road, CT10 2WA, Broadstairs, Kent, South East Region",
                           "Canterbury Christ Church University, North Holmes␣
↪Road, CT1 1QU, Canterbury, Kent, South East Region",
                  "Canterbury Christ Church University, Broomhill Road, TN3␣
↪0TG, Tunbridge Wells, Kent, South East Region",
                "Canterbury Christ Church University, 30 Pembroke Court, Rowan␣
↪Williams Court, ME4 4UF, Chatham, Kent, South East Region"]
print(list_canterburycampuses2[-2])
print(list_canterburycampuses2[-3])
print(list_canterburycampuses2[-1])
print(list_canterburycampuses2[-4])
```

```
Canterbury Christ Church University, Broomhill Road, TN3 0TG, Tunbridge Wells,
Kent, South East Region
Canterbury Christ Church University, North Holmes Road, CT1 1QU, Canterbury,
Kent, South East Region
Canterbury Christ Church University, 30 Pembroke Court, Rowan Williams Court,
ME4 4UF, Chatham, Kent, South East Region
Canterbury Christ Church University, Northwood Road, CT10 2WA, Broadstairs,
Kent, South East Region
```

2.5.3 Adding strings to make a statement to meet up with the request to recommend one of the universities for data science

```
[140]: recommendation = (("Based on your request that i should recommend a university␣
        ↪to you, " + list_canterburycampuses2[-3] + " is " +
                            "the university where I am currently studying for my MSC␣
        ↪program in Data Intelligence."
                            " It's a university with experienced lecturers and course␣
        ↪directors in Data Science field. You can access the schools "
                         "website by clicking the website https://www.canterbury.ac.
        ↪uk/ "))
       print(recommendation)
       print(type(recommendation))
```

```
Based on your request that i should recommend a university to you, Canterbury
Christ Church University, North Holmes Road, CT1 1QU, Canterbury, Kent, South
East Region is the university where I am currently studying for my MSC program
in Data Intelligence. It's a university with experienced lecturers and course
directors in Data Science field. You can access the schools website by clicking
the website https://www.canterbury.ac.uk/
<class 'str'>
```

2.6

TASK 5b: CREATING A SYSTEM/EXAMPLES THAT DEMONSTRATES LISTS, TUPLES, SETS AND DICTIONARIES

2.6.1 Modelling the list of top 5 cheapest camp sites in Kent

Looking at another scenario, modelling the list of top 5 cheapest camp sites in Kent as requested by the students of Canterbury Christ Church University for their December holiday.

This was achieved by creating a variable kentcamp_sites with 5 camp sites that meets with their budget.

```
[141]: kentcamp_sites = ("Kits Coty Glamping", "Ramkins Farm", "Knight's Glamping",
                         "The Finches Caravan & Camping Site", "Glamping at Gravel Pit␣
        ↪Farm")
       print(type(kentcamp_sites))
       print(len(kentcamp_sites))
       print(kentcamp_sites)
```

```
<class 'tuple'>
5
('Kits Coty Glamping', 'Ramkins Farm', "Knight's Glamping", 'The Finches Caravan
& Camping Site', 'Glamping at Gravel Pit Farm')
```

b. Looking at the above list, it is not easily readable by the students, so applying the for loop to have the list created separately from each other to make it readable hence another variable was created to allow for the readablitiy of the list as requested by the students representative.

```
[142]: kentcamp_sites2 = ("Kits Coty Glamping", "Ramkins Farm", "Knight's Glamping",
                     "The Finches Caravan & Camping Site", "Glamping at Gravel Pit␣
↪Farm")
print(len(kentcamp_sites))
for loop in kentcamp_sites2:
    print(loop)
```

```
5
Kits Coty Glamping
Ramkins Farm
Knight's Glamping
The Finches Caravan & Camping Site
Glamping at Gravel Pit Farm
```

c. The cheapest camp site which meets the student's budget was selected from the list of camp sites by adding elements of different data types in a list.

[143]:
```
cheapest_camp = ["Kits Coty Glamping", 2.70,"Aylesford, Maidstone, Kent.␣
↪","costs per hour", 52, "on the average", 20]
print("The cheapest camp site that fits your budget as students is:␣
↪"+cheapest_camp[0]+", located at "+
      cheapest_camp[2]+"It's " +cheapest_camp[3]+ " is "+␣
↪"£"+str(cheapest_camp[4]) + ". \nOther camp sites are: "+kentcamp_sites2[1]+
      ', '+kentcamp_sites2[2]+', '+kentcamp_sites2[3]+',and␣
↪'+kentcamp_sites2[4]+ ". \nThey cost "+"£"+str(cheapest_camp[6])
      +" "+str(cheapest_camp[5]+"."))
print(type(cheapest_camp))
```

```
The cheapest camp site that fits your budget as students is: Kits Coty Glamping,
located at Aylesford, Maidstone, Kent. It's costs per hour is £52.
Other camp sites are: Ramkins Farm, Knight's Glamping, The Finches Caravan &
Camping Site,and Glamping at Gravel Pit Farm.
They cost £20 on the average.
<class 'list'>
```

2.6.2

SETS: MODELLING PRODUCT AVAILABILITY IN TWO DIFFERENT NAMED STORES USING SET DATA TYPE STORAGE

The example below shows the list of groceries available in Lidl store which includes only 4 items and the class type is set

[144]:
```
lidl_groceries = {"milk", "cheese", "candies","cookies"}
print(lidl_groceries)
print(len(lidl_groceries))
print(type(lidl_groceries))
```

```
{'cookies', 'milk', 'cheese', 'candies'}
4
<class 'set'>
```

2.6.3 Combining List and Set

This second example shows the list of items available in Lidl store which includes only 7 items

[145]:
```python
asda_items = {"milk", "pepper", "vegtables","toiletries", "toilet soaps",␣
↪"bathing soaps", "body cream"}
print(asda_items)
print(len(asda_items))
print(type(asda_items))
```

```
{'pepper', 'body cream', 'milk', 'vegtables', 'toilet soaps', 'bathing soaps',
'toiletries'}
7
<class 'set'>
```

Combining list and set: this example is to show how to add several items (of different data types) to an already created set by using the update() method.

It took the different data types in this order: float, followed by integer then strings.

[146]:
```python
lidl_groceries = {"milk", "cheese", "candies","cookies"}
lidl_groceries.update([44, 0.75, 33, 0.23])
print(lidl_groceries)
print(type(asda_items))
```

```
{0.75, 33, 0.23, 'candies', 'cookies', 44, 'milk', 'cheese'}
<class 'set'>
```

Merging or combinig the two sets together using the union() method: Because set does not permit duplicates, it will remove one of the duplicated item, returning only one in the output i.e Milk

Intersection(): Is a method in python set that is used to identify which item is common to both set

[57]:
```python
lidlgroceries_updated = {"milk", "cheese", "candies","cookies", 0.75, 33}
asda_items = {"milk", "pepper", "vegtables","toiletries", 0.23, 44, "toilet␣
↪soaps", "bathing soaps", "body cream"}
combination = lidlgroceries_updated.union(asda_items)
print(len(combination))
print(len(lidlgroceries_updated.intersection(asda_items)))
print(lidlgroceries_updated.intersection(asda_items))
print(type(lidlgroceries_updated))
```

```
14
1
{'milk'}
<class 'set'>
```

The output is saying that there are 14 items in both sets with only 1 item being common to both sets and it is Milk.

```
[58]: lidlgroceries_updated = {"milk", "cheese", "candies","cookies", 0.75, 33}
asda_items = {"milk", "pepper", "vegtables","toiletries", 0.23, 44, "toilet␣
  ↪soaps", "bathing soaps", "body cream"}
combination = lidlgroceries_updated.union(asda_items)
print(combination)
print(len(combination))
print(type(lidlgroceries_updated))
```

```
{0.75, 33, 0.23, 'vegtables', 'candies', 'cookies', 44, 'toilet soaps', 'bathing
soaps', 'pepper', 'body cream', 'milk', 'cheese', 'toiletries'}
14
<class 'set'>
```

2.6.4

Modelling cheapest camp site IN Kent using Dictionary data type Storage

DICTIONARY: Using the exmple of the cheapest camp site requested by CCCU students to create a dictionary to reflect the camp site that site that meets their budget.

Dictionary was used for the data storage type because it includes key-value pairs.

```
[59]: cheapestcamp_detail = {"site name":"Kits Coty Glamping","price per hour":2.
  ↪70,"Location": "Aylesford, Maidstone, Kent","status":"Available"}
print(type(cheapestcamp_detail))
print(cheapestcamp_detail)
print("The site name is "+cheapestcamp_detail["site name"])
print("It costs "+str(cheapestcamp_detail["price per hour"])+ " per hour")
print("It's located at "+cheapestcamp_detail["Location"])
print("Rental status: "+cheapestcamp_detail["status"])
```

```
<class 'dict'>
{'site name': 'Kits Coty Glamping', 'price per hour': 2.7, 'Location':
'Aylesford, Maidstone, Kent', 'status': 'Available'}
The site name is Kits Coty Glamping
It costs 2.7 per hour
It's located at Aylesford, Maidstone, Kent
Rental status: Available
```

```
[60]: print("The cheapest camp site that fits your budget as students is␣
  ↪"+cheapestcamp_detail["site name"]+"." + " It's located at "
      +cheapestcamp_detail["Location"]+ ". \nThe cost per hour is␣
  ↪£"+str(cheapestcamp_detail["price per hour"])+
      ".  It's "+cheapestcamp_detail["status"]+ " for rental. Booking online 4␣
  ↪days to the date gets you 20% discount.")
```

```
The cheapest camp site that fits your budget as students is Kits Coty Glamping.
It's located at Aylesford, Maidstone, Kent.
The cost per hour is £2.7.  It's Available for rental. Booking online 4 days to
the date gets you 20% discount.
```

3. 0

INTRODUCTION OF ACTIVITY 5.2

This activity is all about exploring the power of pandas to read CSV file and converting it to reading and manipulatable dataset in python. It also includes investigating simple plots using the matplotlib module and combine that with using CSV file handling.

a. What is a DataFrame: A Pandas DataFrame is a table with rows and columns. Pandas DataFrames makes data manipulation easy within python. It has some in built function which makes it easy for selecting or replacing columns and indices to reshaping of data. They come with pandas library which are defined as two-dimensional labeled data structures with columns of potentially different data types. Datacamp, 2022.

b. Characteristics of Dataframe: Dataframe has the foloowing characteristics:

c. Each column can have a label name which can contain data of different types.

ii. Each rows has its own index within a range beginning from [0]. The rows index can also be set as labels.

iii. All cells have both a row index and a column index.

iv. Cells can be selected on location-based indexing. Data can be queried within dataframe beased on specific values for data within dataframes based on specific values.

v. Some cells must have null values because of inherent tabular structure.

vi. Arithmetic operations can be performed on rows and columns.

vii. It has labeled axes.

viii. It has mutable size.

ix. It is heterogeneous.

x. Each column in a dataframe is a series. ProgramsBuzz, 2021.

xi. Usefulness of Dataframe in Python: Below are some of the ussefullness of the dataframe in-built functions in data manipulation within the python environment.

xii. It's useful for data manipulation: It can be used to correct the data type a column should have. For example, a csv data may have a date column being parsed as a string. Using the parse_dates optional parameter of the Pandas read_csv method converts the specified column names to the correct datetime data type.

xiii. Configuring Options and Settings: Pandas has a set of in-built functions which allow users to configure options and setting. For example, a user can configure that anytime a csv file is being read by pandas, it should always display the top ten dataset. It can also be used to specify the number of rows and columns shown, and to what precision floating point numbers are displayed. This is a huge productivity booster function since they permit a user to tailor pandas environment to the way the user wants it.

xiv. It's useful in combining two dataframes by using wither the Concatenation or Merging.

xv. Reshaping DataFrames: There are different methods with in-built functions that can be used for reshaping. Some of them are: Transpose, Groupby and Stacking,

xvi. Working with time data: Pandas comes with a function called ‘to_datetime()’ that can compress and convert multiple DataFrame columns into a single Datetime object. Once it’s in that format, it will have all the flexibility of the Datetime library at your disposal.

xvii. Mapping Items into Groups: Mapping is a function that helps with organizing categorical data. For example, a big DataFrame that has thousands of rows, where one of the columns has items we wish to categorize. Categoriizing the items will help in simplify both the training of Machine Learning models and visualizing the data effectively. George, 2019.

3.1.1 Other Tasks

TASK 1a: CREATING A NEW JUPYTER NOTEBOOK TO REFINE ACTIVITY 5.2

From the in-class activity 5.2 which is a part of this assignment, a covid file of csv (comma separated value) extension was downloaded, exracted and uploaded to the jupter notebook. This same task was repeated here in order to refine the in-class activity. The data was used to demonstrate some of the characteristics dataframe and in-built functions of the pandas dataframe in manilupating data.

Steps:

(a). Loading Data to Jupyter notebook: The covid csv file was uploaded to Jupyter notebook. This was done by clicking on load icon (upload files: an arrow facing up with a base) at the top left hand corner of the file browser menu. This connects to the dialogue table where the directory of the extracted file was specified. The file name uploaded was “covid_1.csv”.

(b) Importing of the needed Library: The library needed to successfully read the covid_1.csv is pandas. This library was imported into jupyter i.e ‘import pandas as pd’

Note: From this point, for each code written, the run button must be clicked for jupyter to identify the library.

```
[61]: import pandas as pd
```

(c) dataframe (df): A variable named dataframe (df) was created, this acts as a container which for the code that was used to command pd to read the covid_1.cvs file.

The code interpretation: pd.read_csv says “pandas read covid_1.csv extension”. This helps to create a pandas dataframe for the covid data.

```
[62]: df = pd.read_csv("covid_1.csv")
```

(d) After runining the code in step (iii) above, it will be noticed that it’s as if nothing has happend (because no output was returned). The successful reading of the csv file was checked by using the ‘df.info()’ function to check the metadata of the covid_1.csv. The output confirms that pandas has correctly read the covid_1.csv file if it does not return an error.

```
[63]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
```

```
---  ------      --------------  -----
 0   1           158 non-null    int64
 1   0.00067309  158 non-null    float64
 2   0.0003806   158 non-null    float64
dtypes: float64(2), int64(1)
memory usage: 3.8 KB
```

The interpretation of the above verified information (metadata) about the dataset are:

1. RangeIndex: 158 entries, 0 to 157: There are 158 entries (rows) taking the index number starting from 0 to 157 which has no null entries (no empty cell).
2. There are 3 columns with 2 columns having float as the data type while the last one has integer data type

(e) The code below i.e df.head(10) was used to command dataframe to read the return the top ten rows of the dataset

[64]:
```
df.head(10)
```

[64]:
```
    1  0.00067309  0.0003806
0   2    0.000696   0.000414
1   3    0.000720   0.000449
2   4    0.000746   0.000484
3   5    0.000775   0.000527
4   6    0.000807   0.000572
5   7    0.000843   0.000624
6   8    0.000886   0.000678
7   9    0.000937   0.000741
8  10    0.000995   0.000809
9  11    0.001064   0.000888
```

(f) Just like df.head(10) returned top ten rows, df.tail(10) also returns the bottom ten rows. Thiese two codes helps to affirm that all the data were returned without any row being truncated.

[65]:
```
df.tail(10)
```

[65]:
```
       1  0.00067309  0.0003806
148  150    0.014304   0.018622
149  151    0.013918   0.017867
150  152    0.013545   0.017095
151  153    0.013191   0.016297
152  154    0.012863   0.015485
153  155    0.012560   0.014691
154  156    0.012263   0.013905
155  157    0.011978   0.013141
156  158    0.011700   0.012447
157  159    0.011435   0.011755
```

g. The robustness of python dataframe permits users to run some descriptive statistics. This can be done by using one of the pandas dataframe in-built functions i.e 'df.describe()'. This code returns all the major statistical measures relating to the entire dataset.

```
[66]: df.describe()
```

```
[66]:                1  0.00067309   0.0003806
      count  158.000000  158.000000  158.000000
      mean    80.500000    0.009439    0.012779
      std     45.754781    0.005514    0.007603
      min      2.000000    0.000696    0.000414
      25%     41.250000    0.005688    0.007028
      50%     80.500000    0.008524    0.012666
      75%    119.750000    0.013676    0.018399
      max    159.000000    0.019054    0.026756
```

The above output describes the statistical measures of the entire dataset in the dataframe. Below is a brief description of each of the measures:

a. count: Means that there are 158 data points (rows) in each of the column

b. mean: Is the average of the data in each of the column

c. std: Is the standard deviation for each of the column

d. min: Returns the minimum value in each of the column of the dataset

e. 25%: It is pronounced 25th percentile. Each column have 25 precent of the values less than or equal to 41.25, 0.005688 and 0.007028 for each of the data.

f. 50%:It's pronounced 50th percentile. It means that 50% of the dataset in each column have the values 80.5, 0.008524, and 0.012666 which means that 50% of the dataset are less than or equal to 80.5, 0.008524, and 0.012666. This is the same as measuring the median.

g. 75%: It's pronounced 75th percentile. It means that 75% of the dataset in each column have the values 119.75, 0.0013676, and 0.0118399 which constitutes 75% of the dataset are less than or equal to 119.75, 0.0013676, and 0.0118399.

h. max: Returns the maximum within the dataset in each column. Stackoverflow, 2020

3.1.2 TASK 1(b): MAKING A TUTORIAL TO EXPLAIN THE MAAPTLOTLIB CODE FOR VISUALIZATION

The next step in task 1 of activity 5.2 is visualization of the dataset. Data visualization is a way of presenting dataset pictorially, it helps to find trends and actionable insights from a dataset. Python have some data visualization modules which can be explored to make the data more visually appealing to project pre-defined key performance indicators at a glance. Some of these in built libraries are Matplotlib Seaborn, Plotnine(ggplot), Bokeh, pygal, Plotly, geoplotlib, Gleam, missingno, Leather, Altair, Folium etc. M Blog, 2022

In this activity, some of these libraries was used to demonstrate the robustness of the python data visualization libraries.

Below are the steps invloved in converting the dataset to visualization using the python matplotlib libraries:

a. 'import csv': This was used to import the stored covid_1.csv data for matplotlib to have it handy for visualization. If this was not done, matplotlib will not have a data to reference which may result in error from the code.

b. This was followed by specifying the data source for the library pyplot to reference the variable where the data was stored i.e variable named data

[67]:
```
import csv
```

[68]:
```
data = df
```

3.1.3 Task2 REFORMATING THE CODE TO ALIGN WITHT THE RIGHT INDENTING

Then, pyplot was imported from matplotlib: pyplot was used to plot line graph for the data. Matplotlib is useful for basic graph plotting like line charts, bar graphs, etc

data1[], data2[], and data0[] were variables for each of the columns in the dataset

'with open' is a python library which has the 'r' string which is used to read only the file 'covid_1.csv'

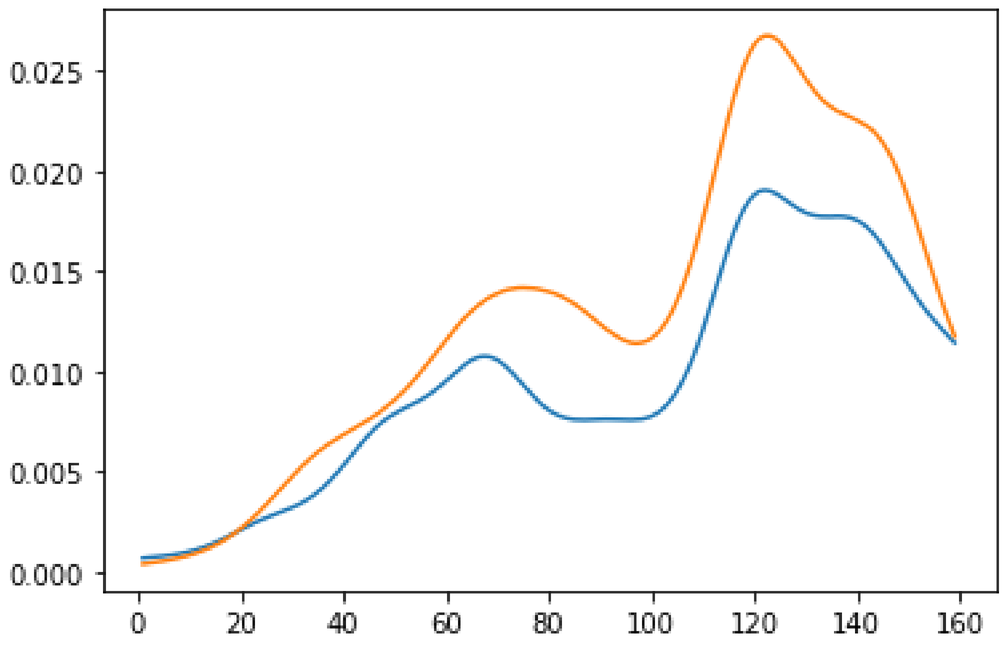
'reader = csv.reader(file): was a command that uses the function reader to read the 'covid_1.csv' file

The data from columns 1 and 2 (which has the float data type) were appended to variables named 'data1' and 'data2' while the dataset from column 0 (which have integer data type) was appended to a variable named 'data0'. The reading of the file was closed by using 'file.close()'.

plt.plot(data0,data1) and plt.plot(data0,data2) were commands used to plot dataset in the variable data0 against each of the other dataset in variables data1 and data2.

plt.show(): was used to command the showing of the plotted line graphs.

[69]:
```
from matplotlib import pyplot as plt
data1=[]
data2= []
data0= []
with open('covid_1.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        data1.append(float(row[1]))
        data2.append(float(row[2]))
        data0.append(int(row[0]))
file.close()
plt.plot(data0,data1)
plt.plot(data0,data2)
plt.show()
```



3.1.4 Task 3: BREAKING THE CODE UP INTO CHUNKS TO EXPLAIN WHAT IS HAPPENING AND WHAT THE CODE DOES.

Below is the explanation of what is happening and what each of the code line does.

a. code line 1: ‘from matplotlib import pyplot as plt’: this code imports the function needed to prepare jupyter notebook for visualization

b. code line 2: ‘data1=[]’: variable data1 was created to store data (list) from column 1

c. code line 3: ‘data2’= []: variable data1 was created to store data (list) from column 2

d. code line 4: ‘data0= []’: variable data1 was created to store data (list) from column 0

e. code line 5: ‘with open(’covid_1.csv’, ‘r’) as file’:‘with open’ is a python function which has the ‘r’ string which is used to read only the file stored i.e ‘covid_1.csv’

f. reader = csv.reader(file): command by reader to read the csv file

for row in reader: data1.append(float(row[1])): append command was used to add the data from row 1 of float data type data2.append(float(row[2])): append command was used to add the data from row 2 of float data type data0.append(int(row[0])): append command was used to add the data from row 0 of integer data type

g. file.close(): after appending, the command reader was closed

h. plt.plot(data0,data1):a command to plot data0 against data1 as a line graph

i. plt.plot(data0,data2): a command to plot data0 against data2 as a line graph

j. plt.show(): a command to the line graph

3.1.5 Task 1(e)

REFINING THE NOTEBOOK ON THE TASK 1 OF ACTIVITY 5.2:

The aim of refining the notebook was to fashion out how to communicate what is going on the Task 1 of activity 5.2

Summarily, what task 1 is all about is how to store a csv file in jupyter notebook, reading the file using the python pandas dataframe, manipulating the data by creating variables which contains a list extracted from the stored csv file and plotting line graphs of different shapes and colours using the pyplot function from the mapplotlib library. It also includes showing the plotted line graph.

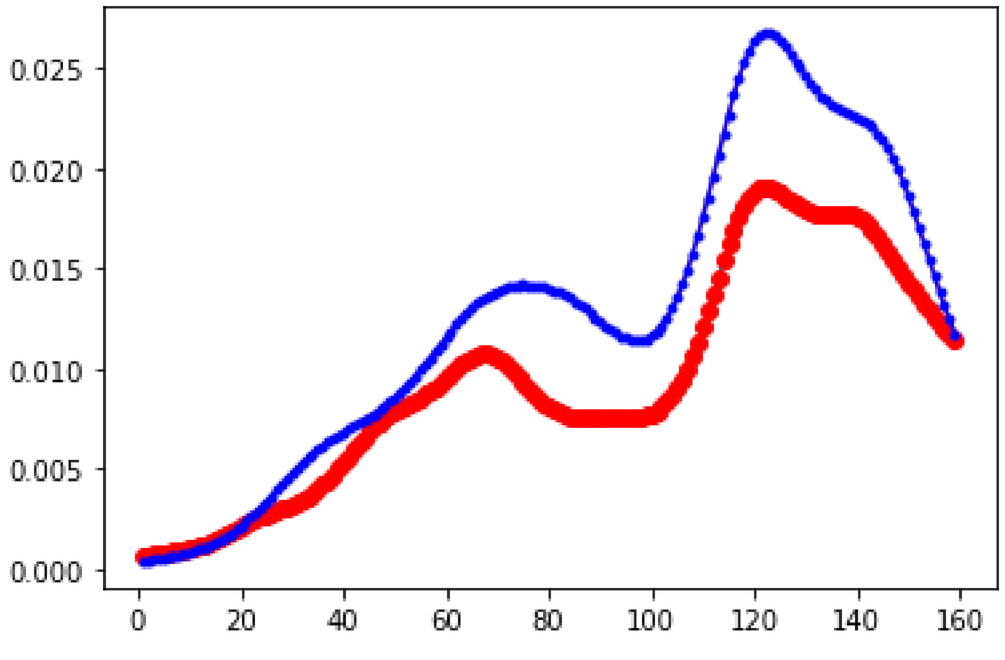
3.2

TASK 2: REPLACING THE LINES MARKED WITH # BELOW WITH LINES OF CODES NOT MARKED WITH #

(i) The task in this first step was to replace the lines marked with # in the code below with lines of code not marked with #. This is needed because the lines of code cannot be run for python operation until it is corrected by removing the comment out sign (#) and putting the code lines in the corrent indent.

#plt.plot(data0,data1)#plt.plot(data0,data2)plt.plot(data0, data1,'ro',data0,data2,'b.-')plt.show()

[70]:
```
plt.plot(data0,data1)
plt.plot(data0,data2)
plt.plot(data0, data1,'ro',data0,data2,'b.-')
plt.show()
```



3.2.1 (iii) EXPERIMENTING WITH DIFFERENT PARAMETERS IN THE ' '(i.e CHARACTER STRINGS) SECTIONS OF THE CODE.

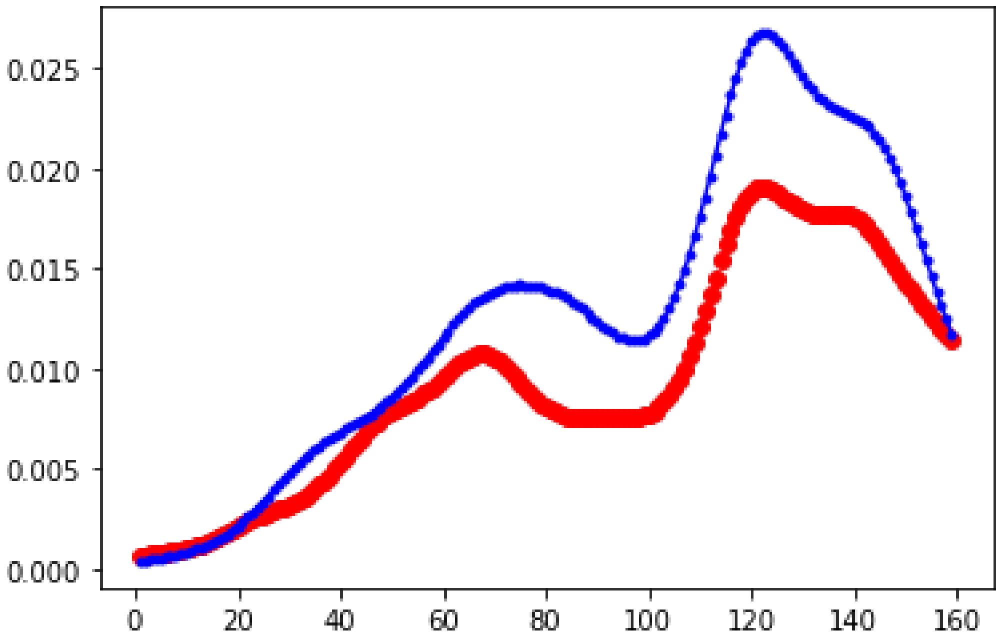
In the previous task above, the line of codes were corrected by eliminating the # in the code. In this task, the description of what the new line of code does, replacing and exploring with different character strings and explaining what each character strings is meant to achieve in the line of codes are of importance.

The new line of code in this case is: plt.plot(data0, data1,'ro',data0,data2,'b.-') with the adiitions of charactier strings 'ro' and 'b.-'

The line graph that was plotted in Task 1 (ii) was further enhanced by adding a third argument to each of the codes for the line graphs. Character strings in python are used to specify the characateristics of the line graphs.

In the line graphs' codes, the third arguments 'ro' is specifying that the first line graph (data0 plotted against data1) should be 'a red solid coloured line graph with circle', while 'b.-' is specifying that the second line graph (data0 plotted against data2) should be 'a blue solid coloured line graph with circle'.

```
[71]: plt.plot(data0, data1,'ro',data0,data2,'b.-')
      plt.show()
```



As required in this section of the project, further research reveals that various line graph types can be achieved by applying different plot symbols and colors while plotting with plot(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted

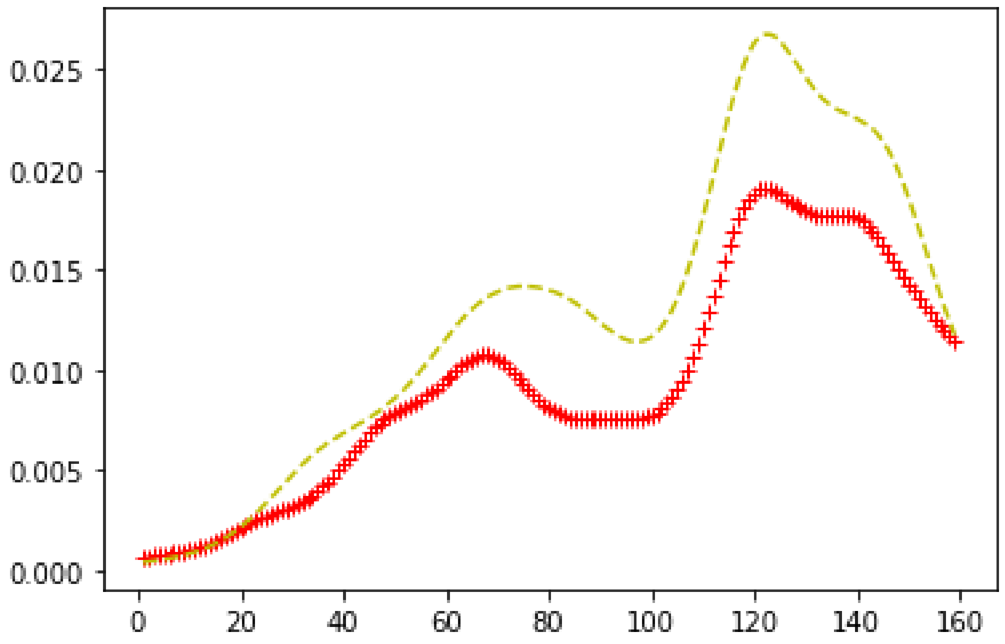
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

John, 2016.

For example, as stated in the activity 5.2 Task 2 (iii): Experiment with different parameters in the ' ' sections of the line, change colours and shapes, some line graphs are shown below with a combination of some character strings beyond what was taught in the class seesions.

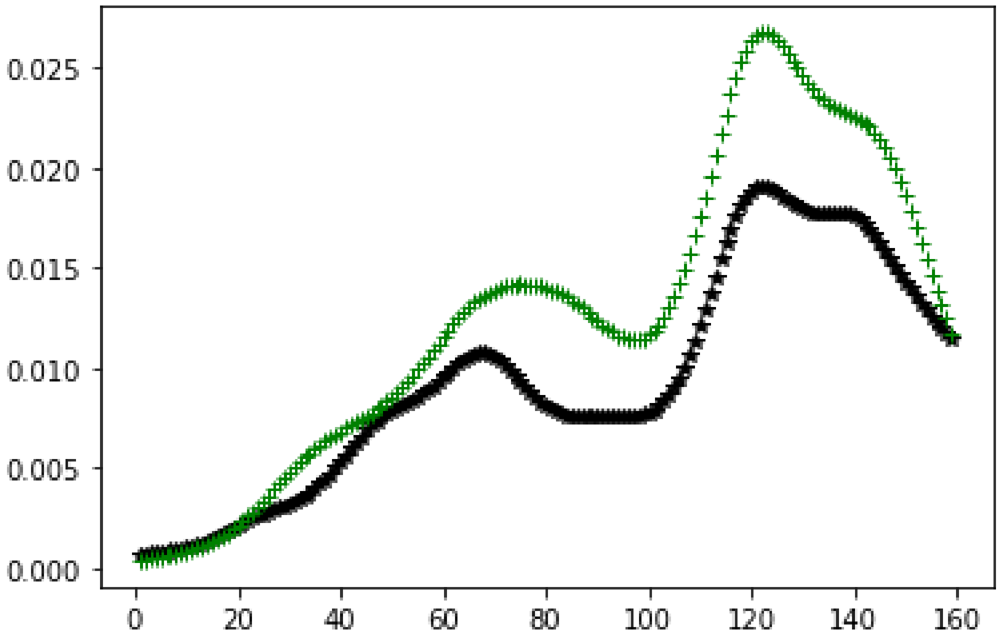
a. Plotting line graphs that have red colour with x mark and yellow colour with dashed lines.

```
[72]: plt.plot(data0, data1,'r+',data0,data2,'y--')
      plt.show()
```



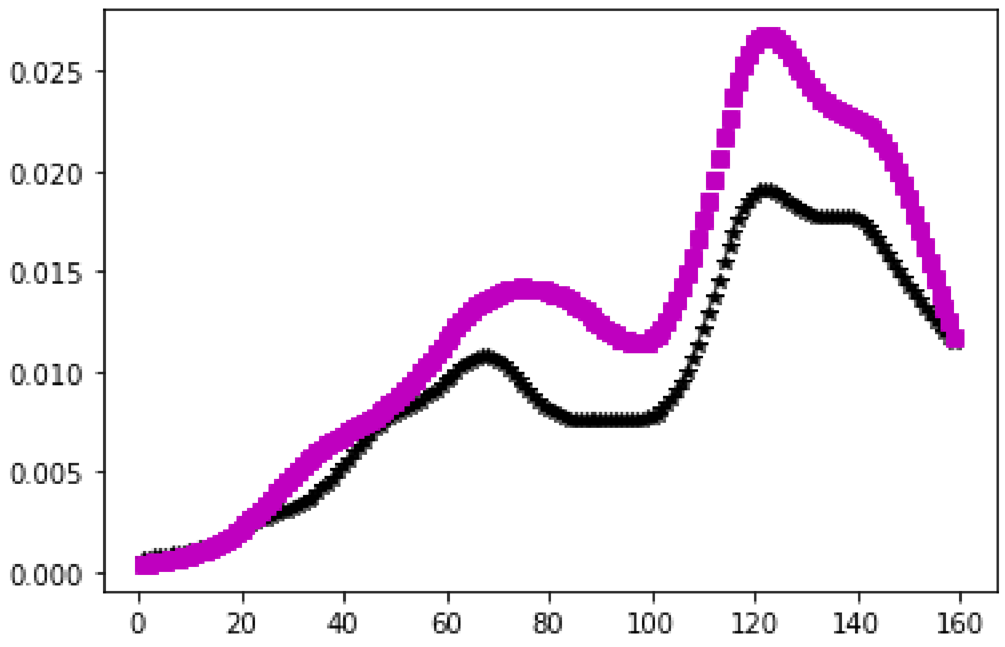
b. Plotting line graphs that have green colour with + mark and black colour with * shaped lines.

[73]:
```
plt.plot(data0, data1,'k*',data0,data2,'g+')
plt.show()
```



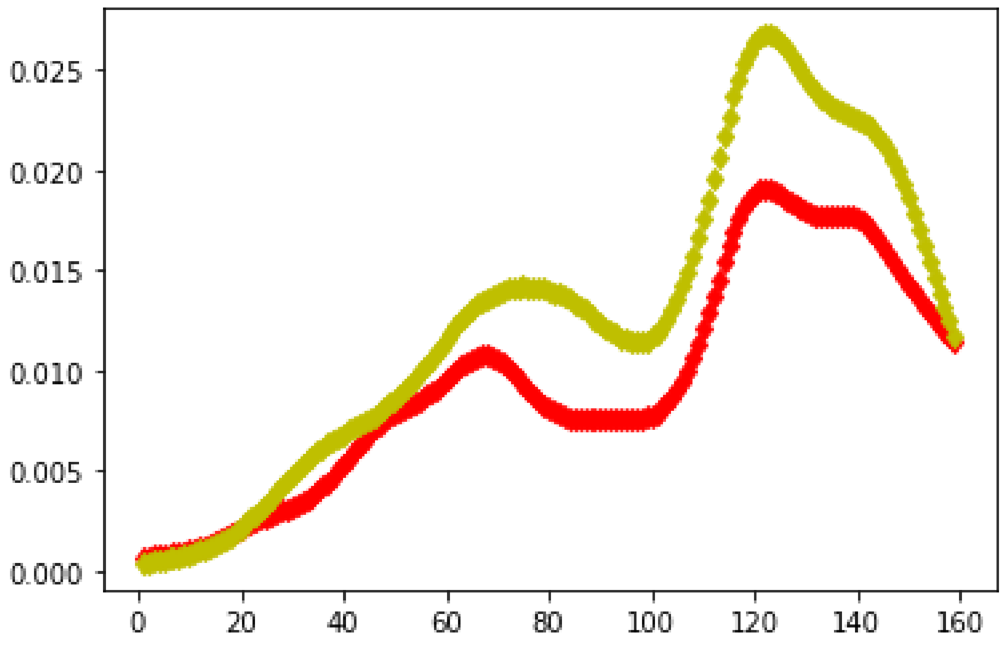
c. Plotting line graphs that have black colour with * mark and cyan colour with square shaped lines.

[74]:
```
plt.plot(data0, data1,'k*',data0,data2,'ms')
plt.show()
```



d. Plotting line graphs that have red colour with square mark and yellow colour with square shaped lines.

[75]:
```
plt.plot(data0, data1,'rd',data0,data2,'yd')
plt.show()
```



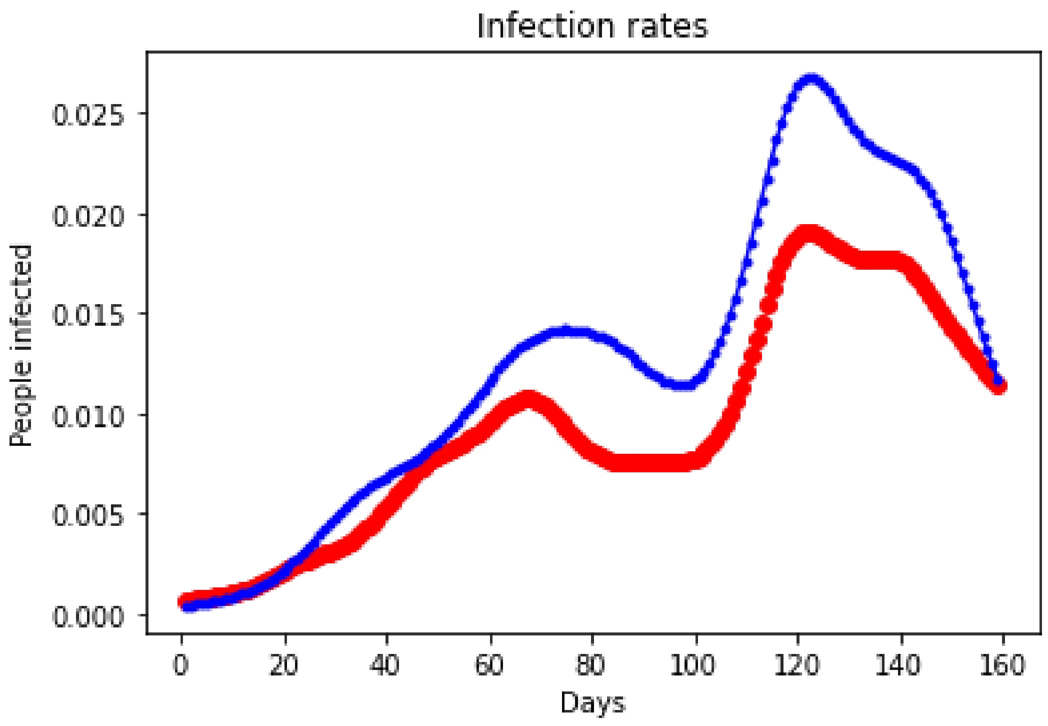
3.2.2 ADDING AXES LABELS AND TITLE TO THE PLOTS

This step is about allocating labels to each of the columns (i.e the axes on the line graphs). The in-built functions of pyplot i.e xlabel() and ylabel() functions makes it possible to set a label for the x- and y-axes.

This was done by specifying the column names and added a title to the graph.

The title for the graph is ‘Infection rates’. The y axis on which data1 and data2 were plotted was labelled as ‘People infected’ while the x axis on which data0 was plotted was labelled as ‘Days’.

[76]:
```
plt.plot(data0, data1,'ro',data0,data2,'b.-')
plt.xlabel("Days")
plt.ylabel("People infected")
plt.title("Infection rates")
plt.show()
```



3.2.3 EXPERIMENTING WITH DIFFERENT STYLES OF LINE GRAPHS

In this exercise, https://www.w3schools.com/python/matplotlib_line.asp was used as a reference source to experiment with different styles of line grpahs.

In addition to the above source, below is an additional reference which itemise the list of different character string that can be used to experiment with different styles of line graphs.

As required in this section of the project, further research reveals that various line graph types can be achieved by applying different plot symbols and colors while plotting with plot(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:

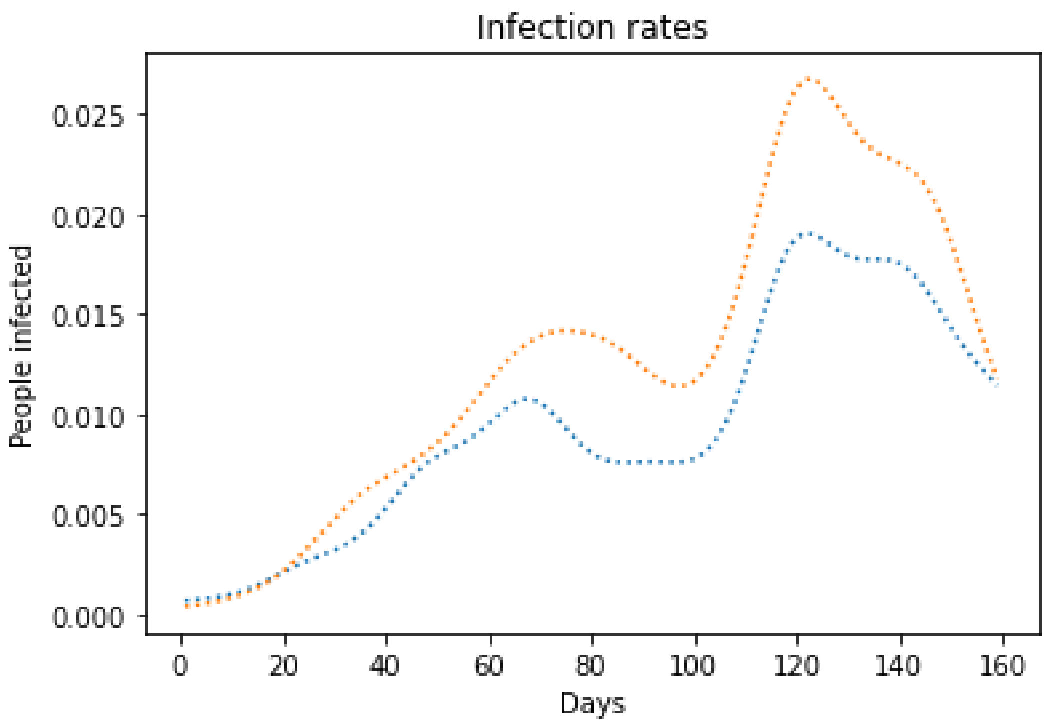
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

John, 2016.

For example, as stated in the activity 5.2 Task 2 (iii): Experiment with different parameters in the ' ' sections of the line, change colours and shapes, some line graphs are shown below with a combination of some character strings beyond what was taught in the class seesions.

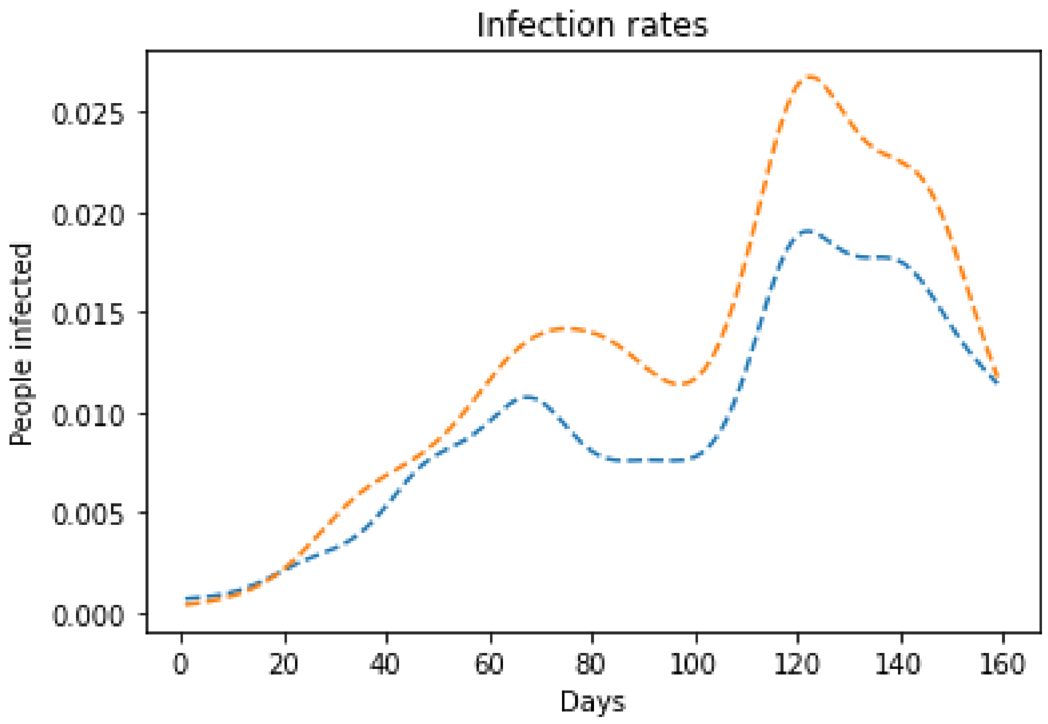
a. Creating line graphs with dotted lines

```
[77]: plt.plot(data0, data1, data0, data2, linestyle = 'dotted')
      plt.xlabel("Days")
      plt.ylabel("People infected")
      plt.title("Infection rates")
      plt.show()
```



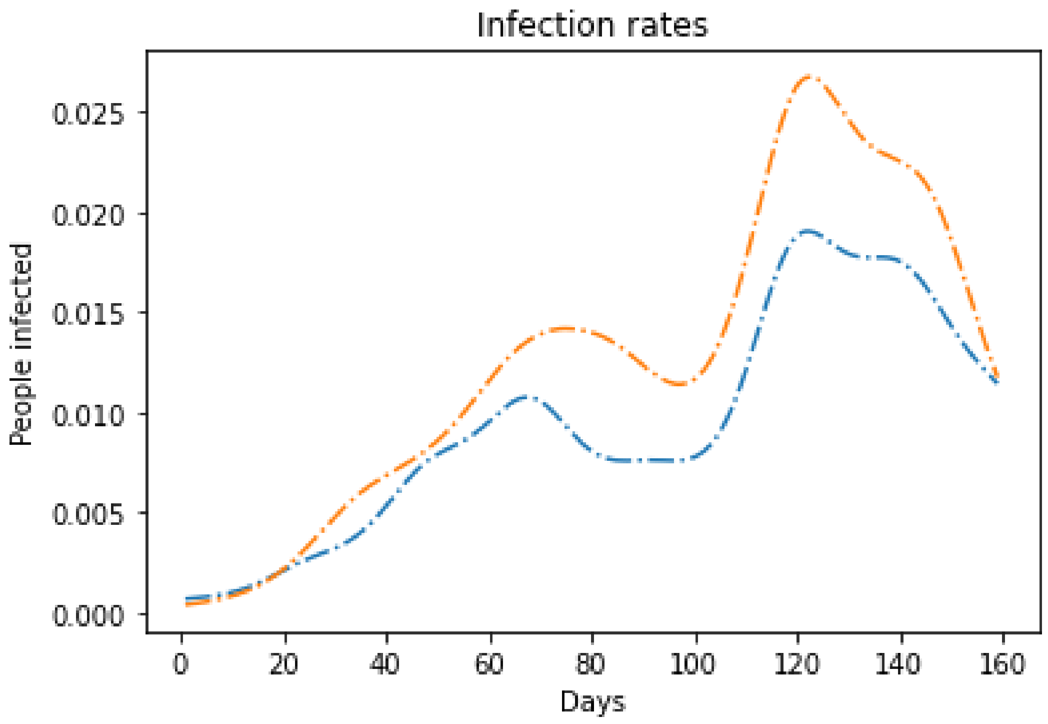
b. Creating line graphs with dashed lines

[78]:
```
plt.plot(data0, data1, data0, data2, ls = 'dashed')
plt.xlabel("Days")
plt.ylabel("People infected")
plt.title("Infection rates")
plt.show()
```



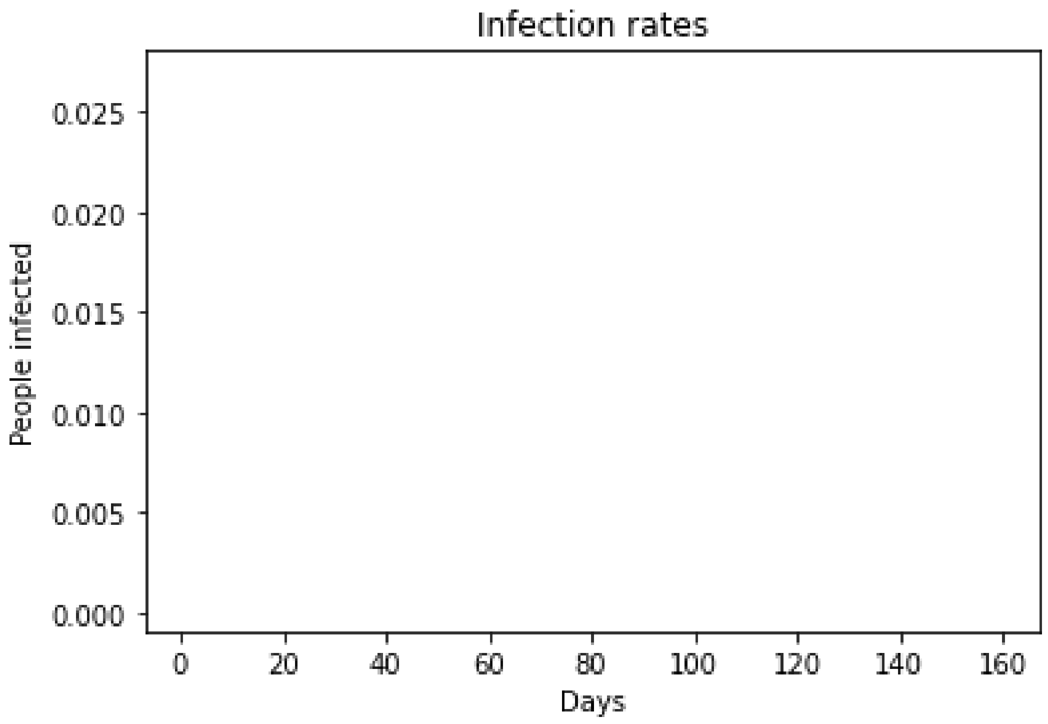
c. Creating line graphs witha combination of dotted and dashed lines

```
[79]: plt.plot(data0, data1, data0, data2, ls = '-.')
      plt.xlabel("Days")
      plt.ylabel("People infected")
      plt.title("Infection rates")
      plt.show()
```



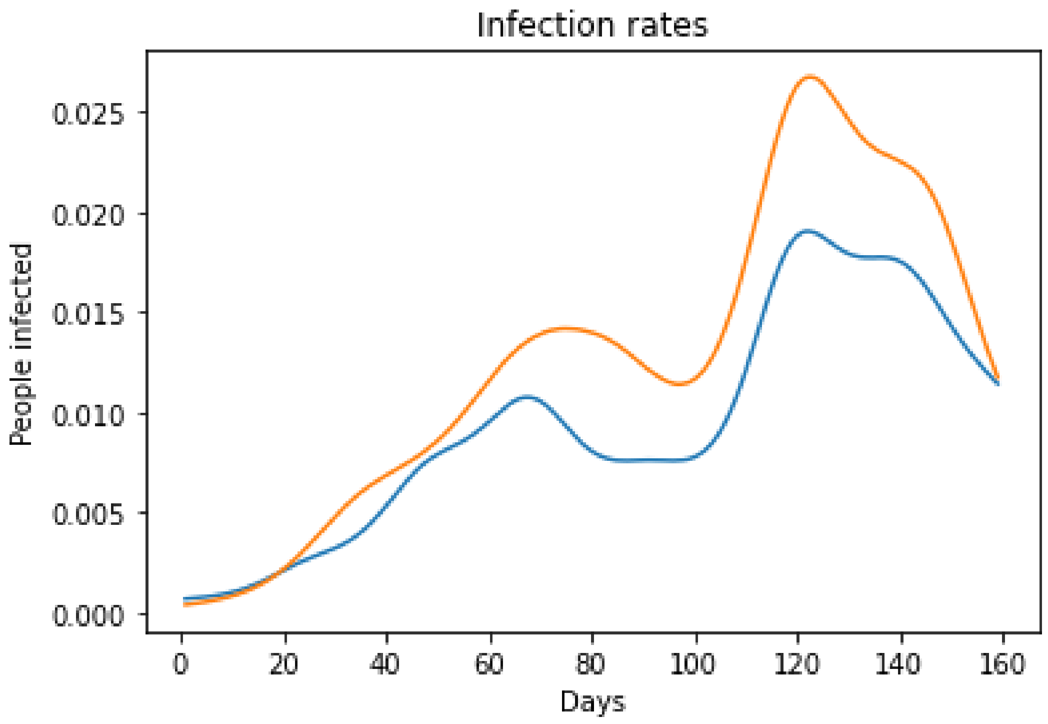
d. Creating line graphs not specifying type of line

[80]:
```
plt.plot(data0, data1, data0, data2, ls = '')
plt.xlabel("Days")
plt.ylabel("People infected")
plt.title("Infection rates")
plt.show()
```



e. Creating line graphs with solid lines

```
[81]: plt.plot(data0, data1, data0, data2, linestyle = 'solid')
      plt.xlabel("Days")
      plt.ylabel("People infected")
      plt.title("Infection rates")
      plt.show()
```

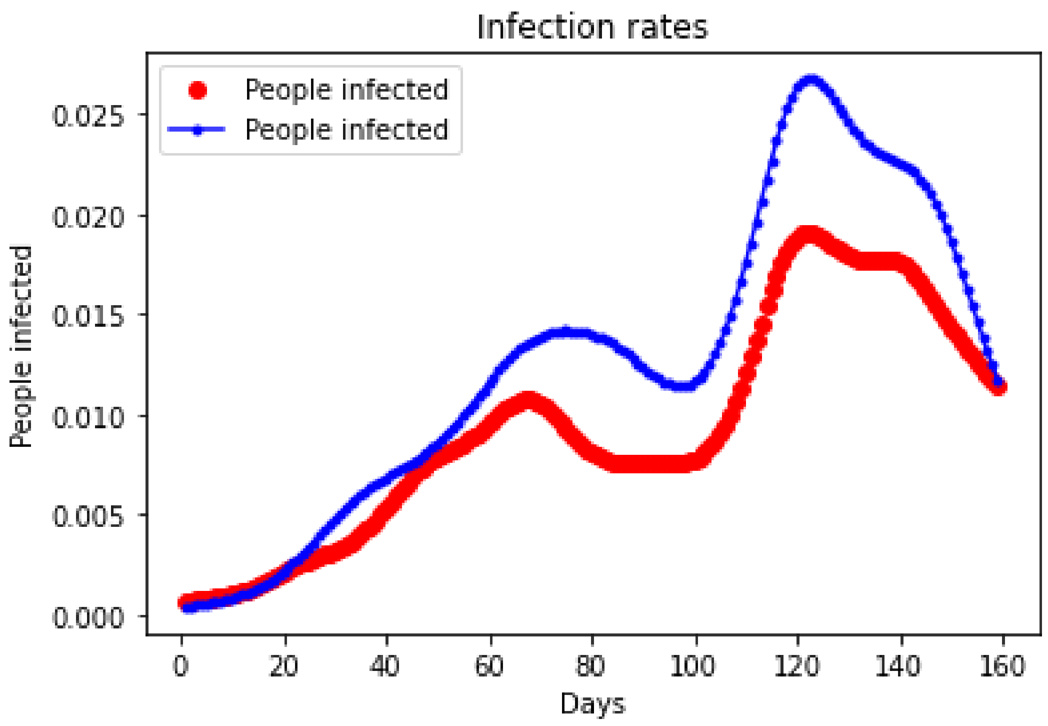


TASK 3: In this task, using https://www.w3schools.com/python/matplotlib_intro.asp as a reference and starting point and all the links to MATPLOTLIB in the left-hand column, more features were added to the graphs. For examples grids, markers and anything that are appropriate to make the graphs more informative.

This task is all about matplotlib. Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

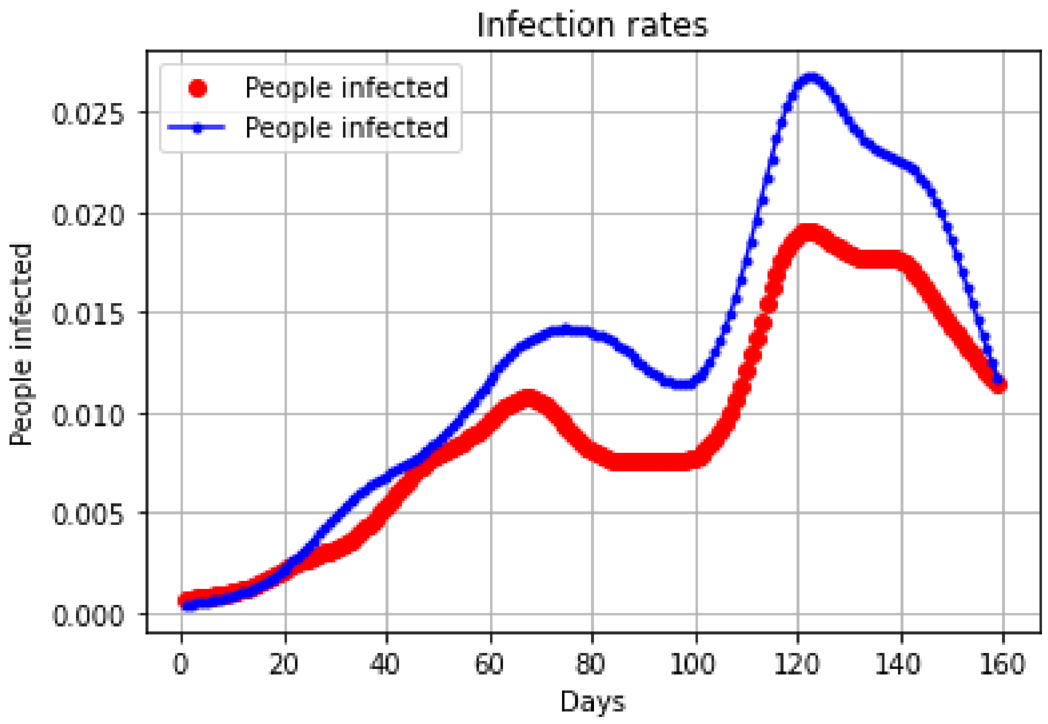
(i) Adding legend to the plot

[82]:
```
plt.plot(data0, data1,'ro',data0,data2,'b.-', label='People infected')
plt.xlabel("Days")
plt.ylabel("People infected")
plt.title("Infection rates")
plt.legend(loc = 'best')
plt.show()
```



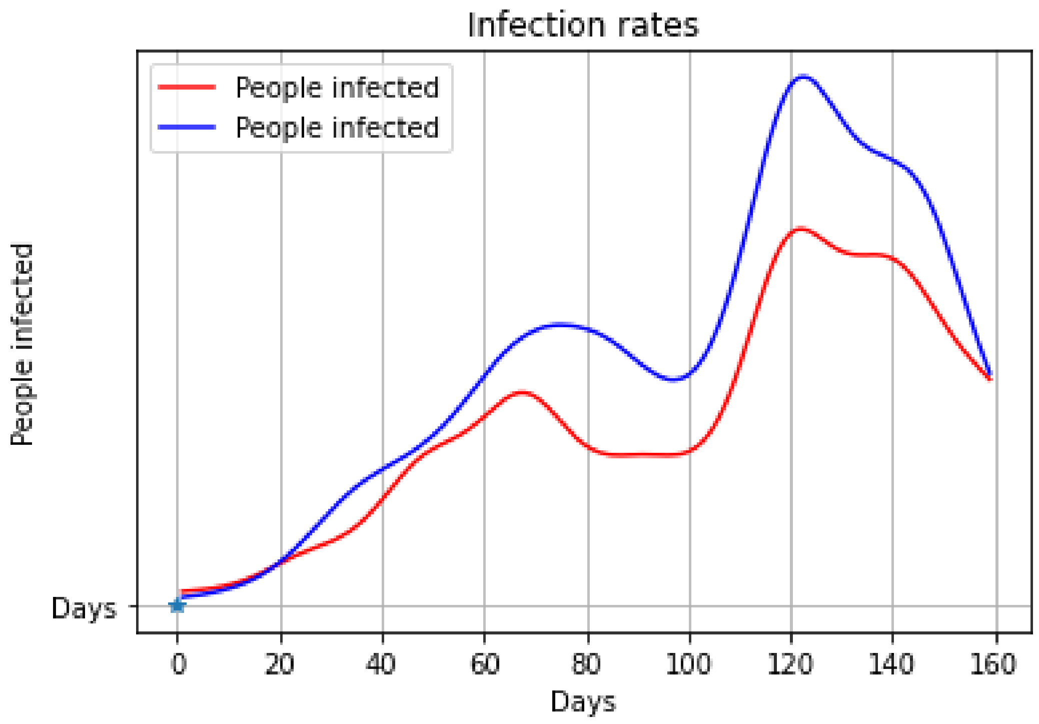
(ii) Adding grid to the plot

```
[83]: plt.plot(data0, data1,'ro',data0,data2,'b.-', label='People infected')
      plt.xlabel("Days")
      plt.ylabel("People infected")
      plt.title("Infection rates")
      plt.legend(loc = 'best')
      plt.grid()
      plt.show()
```



(iii) Adding marker to the plot

```
[84]: plt.plot(data0, data1,'r',data0,data2,'b', label="People infected")
plt.xlabel("Days")
plt.ylabel("People infected")
plt.title("Infection rates")
plt.legend(loc = 'best')
plt.grid()
plt.plot("Days", marker = '*')
plt.show()
```



3.3 TASK 4: NUMPY

Another library named numpy was imported that is useful in data science is the numpy.

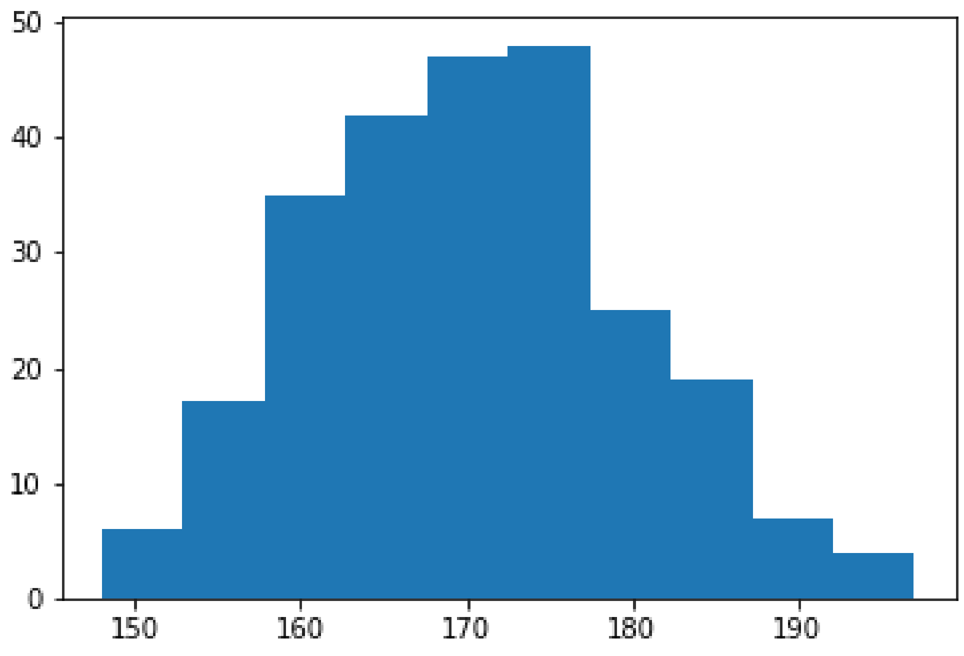
What is Numpy: "It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more." Numpy 2008-2022.

Numpy is basically known as Numerical python. It is the standard library for scientific computing in Python.

Applying the Task 3 skills of plotting line graphs, numpy was used to plot another type of graph (in this case histogram)

3.3.1 Cleaning of the code

[85]:
```
import numpy as np
from matplotlib import pyplot as plt
npx = np.random.normal(170, 10, 250)
plt.hist(npx)
plt.show()
```



3.3.2 EXPLAINING THE CODE:

import numpy as np: numpy was imported which is the library needed for numerical computation

from matplotlib import pyplot as plt: for plotting of the graph

npx = np.random.normal(170, 10, 250): npx is just a variable holding the data for plotting

plt.hist(npx): command to plot the npx data as histogram

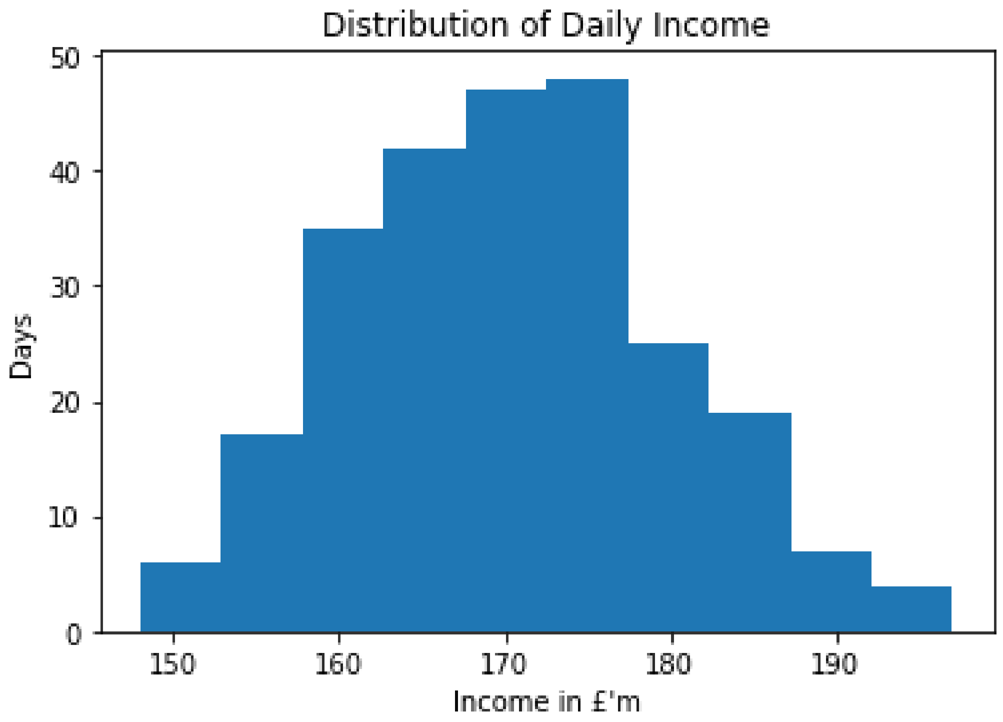
plt.show(): command to show the histogram graph

(ii) numpy is another module that is mostly used in data science (check https://www.w3schools.com/python/numpy/default.asp for details). Data can be plotted in arrays, probability distributions, etc. Applying some of the learnt in task 3, appropriate axes labels and grids were added to the graphs to make it much more understandable.

3.3.3 APPLYING TASK 3 SKILLS:

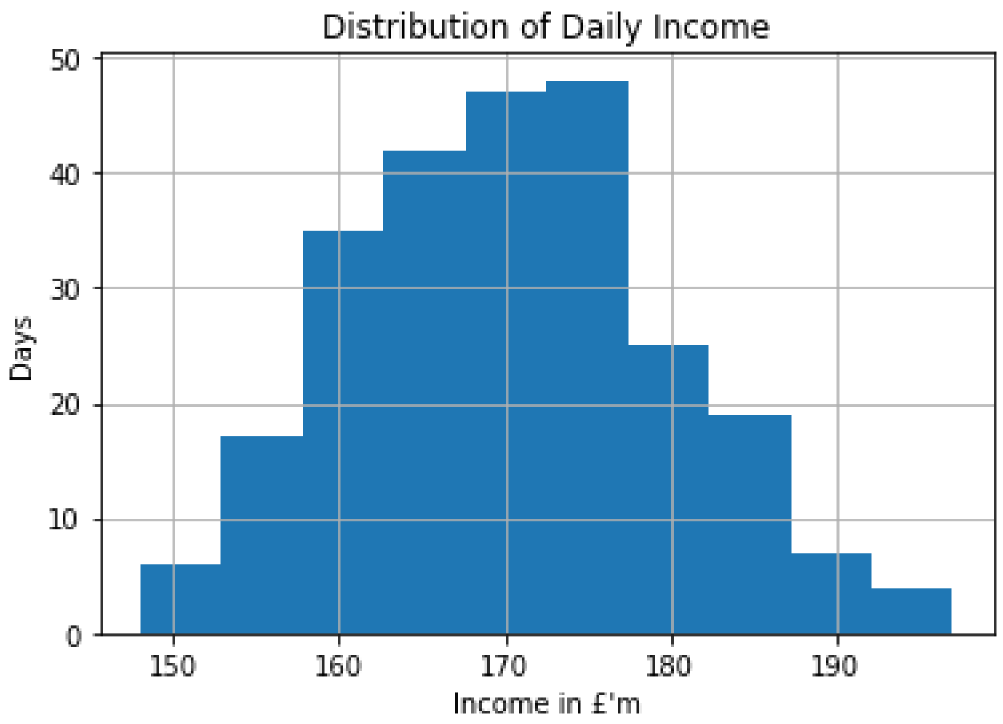
a. ADDING AXES LABELS AND TITLE

[86]:
```
plt.hist(npx)
plt.xlabel("Income in £'m")
plt.ylabel("Days")
plt.title("Distribution of Daily Income")
plt.show()
```



a. ADDING GRID TO THE GRAPH

[87]:
```
plt.hist(npx)
plt.xlabel("Income in £'m")
plt.ylabel("Days")
plt.title("Distribution of Daily Income")
plt.grid()
plt.show()
```

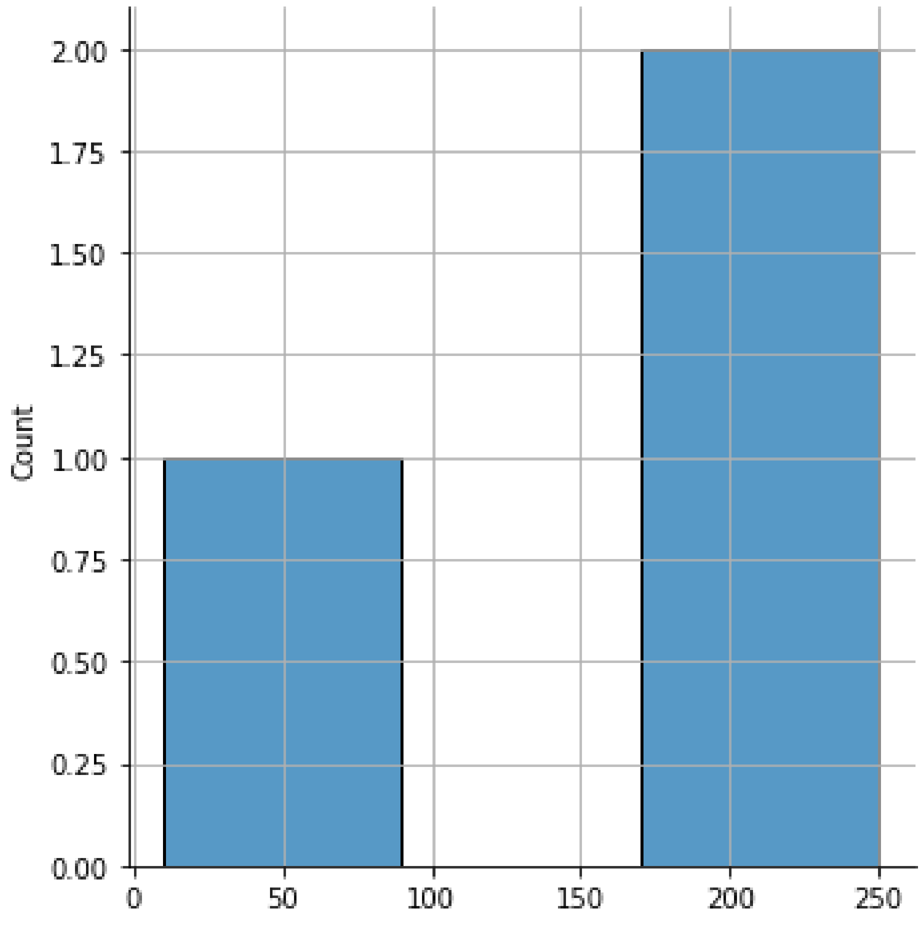


3.3.4 Extension of the Codes not Discussed in the Class

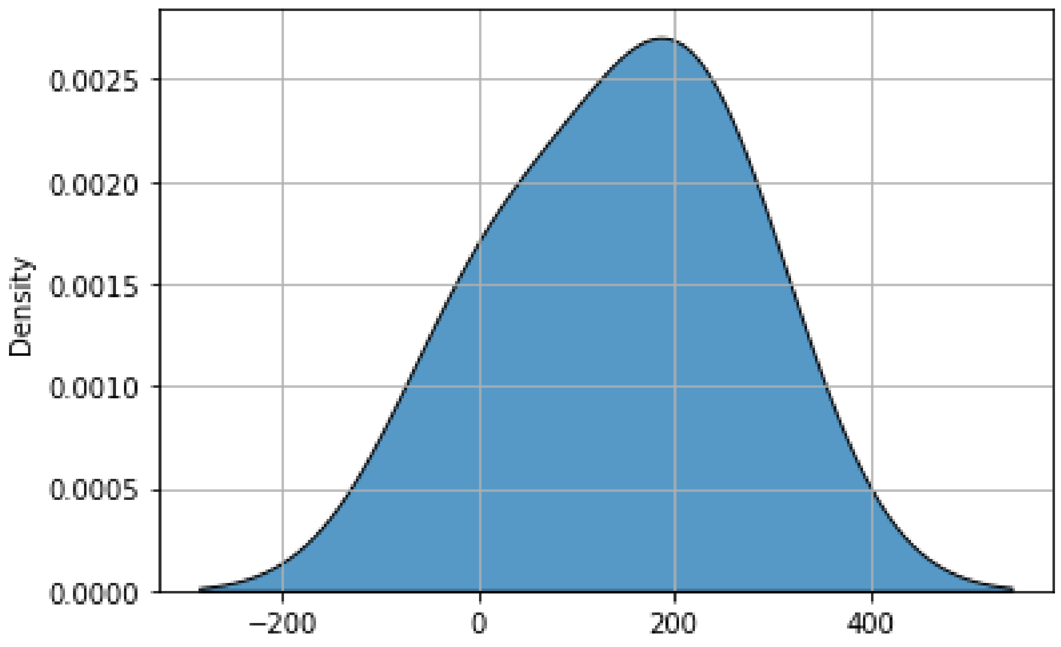
Based on further research, below are some of the new plots that were discovered which was not part of the ones discussed or covered in the class session. This includes using the seaborn library (i.e sns.diplot()) to plot random distribution graph of the dataset.

[88]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot([170, 10, 250])
plt.grid()
plt.show()
```



[89]:
```
sns.kdeplot([170, 10, 250], x="flipper_length_mm", multiple="stack")
#Seaborn, Available from: https://seaborn.pydata.org/tutorial/function_overview.
  ↪html
plt.grid()
plt.show()
```



4.0

TASK 1 INTORDUCTION OF ACTIVITY 8

This part entails fetching current weather data from a particular website. This was achieved by going to a website (https://openweathermap.org/api/) to subscribe for free API.

PROCESS OF REQUESTING FOR API

a. Click on the website https://openweathermap.org/api/ which opens on a separate browser
b. Subscribe for free API access registration by providing an email address. API generated will be sent to the to the registered email address. This has to be for latter use i.e (395e708b71e8ede98016b9e1ee1dd64f)

4.1

TASK 2: SPECIFYING API, IMPORTING THE REQUIRED LIBRARY AND SPECIFYING LOCATION

This task entails bringing data from web sources (in this case, current weather data)

Below are the steps followed to achieve this:

a. SPECIFYING API, IMPORTING THE REQUIRED LIBRARY AND SPECIFYING LOCATION

In this task, a modified version of "Project: Fetching Current Weather Data" from "Automate the boring stuff with Python" by Al Sweigart e-book available at https://ebookcentral.proquest.com/lib/canterburychristchurch/detail.action?docID=4503140 pages 383-387 was used.

STEPS:

(i) Specifying API: This was done by creating a variable named 'APP_ID' which acts as a container to keep the generated API

(ii) Importing the Required Library: The library required for this task is json which comes with 'request' system function. This is the function that is used by the API to pull the weather data for the specified location.

(iii) Define your location of choice in a variable named 'Location' with the country name in this case the Location is @Gravsend' and the country is United. This will help the request function to use the API to request for the current weather data of the specified location.

4.2 Task 3: Breaking of code into chunks and explanation

import json, requests

DEFINING VARIABLES AND CONSTANTS

1 constant APP_ID definition

APP_ID= "395e708b71e8ede98016b9e1ee1dd64f"

2 variable Location definition

Location='Gravesend,United Kingdom'

3 variable url definition

url='http://api.openweathermap.org/data/2.5/weather?q='+Location+'&APPID='+APP_ID+'&units=metric'
print(url)

4 Sends a GET request.

5 :param url: URL for the new Request object.

response= requests.get(url)

6 checks and raises HTTPError if any occured

response.raise_for_status()

#prints the string value of the response to a stream print(response.text)

7 Deserialize the string value to a python object and assign to a variable "weatherData"

weatherData = json.loads(response.text)

8 extract the value of the key "weather" from the variable "weatherData" and assign this value to variable "w"

w=weatherData['weather']

9 assign the temp_max from the weatherData python object to a vairable "max_degree_in_celsuis"

max_degree_in_celsuis = weatherData["main"]["temp_max"]

10 Converting to Fahrenheit

max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

11 Prints out to the stream or console

print('Current weather in %s:' % (weatherData["name"])) print(w[0]['description']) print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis)) print('The maximum temperature in Fahrenheit is %s:' %(max_degree_in_fahrenheit))

References: # https://openweathermap.org/current (OpenWeatherMap API documentation) # https://www.metric-conversions.org/temperature/celsius-to-fahrenheit.htm (for the conversion from Degree Centigrade to Fahrenheit formula)

[10]:
```
# 4.2.1 Task 4: Getting Current Weather Data for Gravesend



import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='Gravesend,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
 ↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)
```

```
# Deserialize the string value to a python object and assign to a variable␣
↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
↪"max_degree_in_celsuis"
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=Gravesend,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
{"coord":{"lon":0.3737,"lat":51.4414},"weather":[{"id":501,"main":"Rain","descri
ption":"moderate rain","icon":"10d"}],"base":"stations","main":{"temp":7.27,"fee
ls_like":3.31,"temp_min":5.94,"temp_max":8.38,"pressure":1012,"humidity":92,"sea
_level":1012,"grnd_level":1010},"visibility":10000,"wind":{"speed":7.38,"deg":19
4,"gust":13.2},"rain":{"1h":1.15},"clouds":{"all":100},"dt":1673340424,"sys":{"t
ype":2,"id":2009659,"country":"GB","sunrise":1673337652,"sunset":1673367025},"ti
mezone":0,"id":2648187,"name":"Gravesend","cod":200}
Current weather in Gravesend:
moderate rain
The maximum temperature in Centigrade is 8.38:
The maximum temperature in Fahrenheit is 47.084:
```

[11]:
```
# 4.2.2 Task 5: Getting Current Weather Data for Gillingham

import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='Gillingham,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)

# Deserialize the string value to a python object and assign to a variable␣
 ↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
 ↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
 ↪"max_degree_in_celsuis"
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
 ↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=Gillingham,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
{"coord":{"lon":0.5486,"lat":51.3891},"weather":[{"id":500,"main":"Rain","descri
ption":"light rain","icon":"10d"}],"base":"stations","main":{"temp":7.15,"feels_
like":6.08,"temp_min":5.9,"temp_max":8.25,"pressure":1010,"humidity":91},"visibi
lity":10000,"wind":{"speed":1.79,"deg":0,"gust":4.92},"rain":{"1h":0.87},"clouds
":{"all":100},"dt":1673340427,"sys":{"type":2,"id":2009659,"country":"GB","sunri
se":1673337595,"sunset":1673366998},"timezone":0,"id":2648657,"name":"Gillingham
","cod":200}
Current weather in Gillingham:
light rain
The maximum temperature in Centigrade is 8.25:
The maximum temperature in Fahrenheit is 46.85:
```

[12]:
```
# 4.2.3 Task 6: Getting Current Weather Data for Essex
```

```python
import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='Essex,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
 ↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)

# Deserialize the string value to a python object and assign to a variable␣
 ↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
 ↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
 ↪"max_degree_in_celsuis"
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
 ↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=Essex,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
```

```
{"coord":{"lon":-
74.2163,"lat":40.7834},"weather":[{"id":802,"main":"Clouds","description":"scatt
ered clouds","icon":"03n"}],"base":"stations","main":{"temp":1.49,"feels_like":-
0.79,"temp_min":-
1.4,"temp_max":4.26,"pressure":1015,"humidity":70},"visibility":10000,"wind":{"s
peed":2.06,"deg":200},"clouds":{"all":40},"dt":1673340521,"sys":{"type":2,"id":2
029393,"country":"US","sunrise":1673353232,"sunset":1673387259},"timezone":-
18000,"id":5097707,"name":"Essex","cod":200}
Current weather in Essex:
scattered clouds
The maximum temperature in Centigrade is 4.26:
The maximum temperature in Fahrenheit is 39.668:
```

[40]:

```python
# 4.2.4 Task 7: Getting Current Weather Data for Canterbury

import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='Canterbury,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
 ↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)

# Deserialize the string value to a python object and assign to a variable␣
 ↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
 ↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
 ↪"max_degree_in_celsuis"
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
  ↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=Canterbury,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
```

```
{"coord":{"lon":1.0799,"lat":51.279},"weather":[{"id":500,"main":"Rain","descrip
tion":"light rain","icon":"10d"}],"base":"stations","main":{"temp":8.04,"feels_l
ike":5.81,"temp_min":7.68,"temp_max":8.94,"pressure":1008,"humidity":98},"visibi
lity":10000,"wind":{"speed":3.58,"deg":158,"gust":9.39},"rain":{"1h":0.87},"clou
ds":{"all":100},"dt":1673351064,"sys":{"type":2,"id":2001810,"country":"GB","sun
rise":1673337438,"sunset":1673366901},"timezone":0,"id":2653877,"name":"Canterbu
ry","cod":200}
```

```
Current weather in Canterbury:
light rain
The maximum temperature in Centigrade is 8.94:
The maximum temperature in Fahrenheit is 48.092:
```

[34]:
```
# 4.2.5 Task 8: Getting Current Weather Data for Stoke

import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='Stoke,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
  ↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)

# Deserialize the string value to a python object and assign to a variable␣
  ↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
  ↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
  ↪"max_degree_in_celsuis"
```

```
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
 ↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=Stoke,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
```

```
{"coord":{"lon":-
2.8667,"lat":53.25},"weather":[{"id":501,"main":"Rain","description":"moderate r
ain","icon":"10d"}],"base":"stations","main":{"temp":6.6,"feels_like":1.9,"temp_
min":5.68,"temp_max":8.58,"pressure":1001,"humidity":92},"visibility":8000,"wind
":{"speed":9.26,"deg":160,"gust":15.95},"rain":{"1h":1},"clouds":{"all":75},"dt"
:1673343340,"sys":{"type":2,"id":2004398,"country":"GB","sunrise":1673338955,"su
nset":1673367278},"timezone":0,"id":2636868,"name":"Stoke","cod":200}
```

```
Current weather in Stoke:
moderate rain
The maximum temperature in Centigrade is 8.58:
The maximum temperature in Fahrenheit is 47.444:
```

[15]:

```
# 4.2.6 Task 9: Getting Current Weather Data for London

import json, requests

# Defining Variables and Constants
APP_ID= "395e708b71e8ede98016b9e1ee1dd64f" #constant APP_ID definition
Location='London,United Kingdom' #variable Location definition
url='http://api.openweathermap.org/data/2.5/weather?
 ↪q='+Location+'&APPID='+APP_ID+'&units=metric' #variable url definition
print(url)

# Sends a GET request.
# :param url: URL for the new Request object.
response= requests.get(url)

# chcecks and raises HTTPError if any occured
response.raise_for_status()

#prints the string value of the response to a stream
print(response.text)
```

```
# Deserialize the string value to a python object and assign to a variable␣
↪"weatherData"
weatherData = json.loads(response.text)
# extract the value of the key "weather" from the variable "weatherData" and␣
↪assign this value to variable "w"
w=weatherData['weather']

# assign the temp_max from the weatherData python object to a vairable␣
↪"max_degree_in_celsuis"
max_degree_in_celsuis = weatherData["main"]["temp_max"]

# Converting to Fahrenheit
max_degree_in_fahrenheit = max_degree_in_celsuis * 9/5 + 32

# Prints out to the stream or console
print('Current weather in %s:' % (weatherData["name"]))
print(w[0]['description'])
print('The maximum temperature in Centigrade is %s:' % (max_degree_in_celsuis))
print('The maximum temperature in Fahrenheit is %s:'␣
↪%(max_degree_in_fahrenheit))
```

```
http://api.openweathermap.org/data/2.5/weather?q=London,United
Kingdom&APPID=395e708b71e8ede98016b9e1ee1dd64f&units=metric
{"coord":{"lon":-
0.1257,"lat":51.5085},"weather":[{"id":500,"main":"Rain","description":"light ra
in","icon":"10d"}],"base":"stations","main":{"temp":7.3,"feels_like":4.62,"temp_
min":5.69,"temp_max":8.2,"pressure":1010,"humidity":92},"visibility":7000,"wind"
:{"speed":4.12,"deg":170},"rain":{"1h":0.29},"clouds":{"all":75},"dt":1673340405
,"sys":{"type":2,"id":2075535,"country":"GB","sunrise":1673337790,"sunset":16733
67127},"timezone":0,"id":2643743,"name":"London","cod":200}
Current weather in London:
light rain
The maximum temperature in Centigrade is 8.2:
The maximum temperature in Fahrenheit is 46.76:
```

4.3 TASK 2, QUESTION 3: Combining Current Weather API Data with Data from Other API

Combining the API with data from others APIs to extend what the software does. You will have to get a new API key for the new APIs you use.

4.4 TASK 2, QUESTION 4: COMBINING THE WEATHER OUTPUTS FOR 6 DIFFERENT CITIES IN TABULAR AND GRAPHICAL FORM.

a. Converting the current weather data: The weather data in the 6 locations were converted in dictionary data storage type give tne ouput below.

[16]:
```
weather_info = [{"Location": "Gravesend", "Current Weather": "moderate rain",
                "Maximum Temperature Degree Centigrade": 8.38, "Maximum␣
↪Temperature in Fahrenheit": 47.084},
                {"Location": "Gillingham", "Current Weather": "light rain",
                "Maximum Temperature Degree Centigrade": 8.25, "Maximum␣
 ↪Temperature in Fahrenheit": 46.85},
                {"Location": "Essex", "Current Weather": "scattered clouds",
                "Maximum Temperature Degree Centigrade": 4.26, "Maximum␣
 ↪Temperature in Fahrenheit": 39.668},
                {"Location": "Canterbury", "Current Weather": "light rain",
                "Maximum Temperature Degree Centigrade": 8.91, "Maximum␣
 ↪Temperature in Fahrenheit": 48.038},
                {"Location": "Stoke", "Current Weather": "moderate rain",
                "Maximum Temperature Degree Centigrade": 8.58, "Maximum␣
 ↪Temperature in Fahrenheit": 47.444},
                {"Location": "London", "Current Weather": "light rain",
                "Maximum Temperature Degree Centigrade": 8.2, "Maximum␣
 ↪Temperature in Fahrenheit": 46.76}]
print(weather_info)
```

```
[{'Location': 'Gravesend', 'Current Weather': 'moderate rain', 'Maximum
Temperature Degree Centigrade': 8.38, 'Maximum Temperature in Fahrenheit':
47.084}, {'Location': 'Gillingham', 'Current Weather': 'light rain', 'Maximum
Temperature Degree Centigrade': 8.25, 'Maximum Temperature in Fahrenheit':
46.85}, {'Location': 'Essex', 'Current Weather': 'scattered clouds', 'Maximum
Temperature Degree Centigrade': 4.26, 'Maximum Temperature in Fahrenheit':
39.668}, {'Location': 'Canterbury', 'Current Weather': 'light rain', 'Maximum
Temperature Degree Centigrade': 8.91, 'Maximum Temperature in Fahrenheit':
48.038}, {'Location': 'Sittingbourne', 'Current Weather': 'moderate rain',
'Maximum Temperature Degree Centigrade': 9.46, 'Maximum Temperature in
Fahrenheit': 49.028}, {'Location': 'London', 'Current Weather': 'light rain',
'Maximum Temperature Degree Centigrade': 8.2, 'Maximum Temperature in
Fahrenheit': 46.76}]
```

4.4.1 Converting the current data into pandas dataframe: The current weather data in the 6 cities were converted into dataframe by first importing the pandas library. This was followed by carrying out some data description about the dataset (metadata) followed by using python basic statistical computation by using the .info() and .describe() resepctively.

```
[45]: import pandas as pd

df3 =[{'Location': 'Gravesend', 'Current Weather': 'moderate rain', 'Max.␣
 ↪Temperature in Degree Centigrade': 8.38, 'Max. Temperature in Fahrenheit':␣
 ↪47.084},
      {'Location': 'Gillingham', 'Current Weather': 'light rain', 'Max.␣
 ↪Temperature in Degree Centigrade': 8.25, 'Max. Temperature in Fahrenheit':␣
 ↪46.85},
      {'Location': 'Essex', 'Current Weather': 'scattered clouds', 'Max.␣
 ↪Temperature in Degree Centigrade': 4.26, 'Max. Temperature in Fahrenheit':␣
 ↪39.668},
    {'Location': 'Canterbury', 'Current Weather': 'light rain', 'Max.␣
↪Temperature in Degree Centigrade': 8.91, 'Max. Temperature in Fahrenheit':␣
↪48.038},
    {'Location': 'Stoke', 'Current Weather': 'moderate rain', 'Max.␣
↪Temperature in Degree Centigrade': 8.58, 'Max. Temperature in Fahrenheit':␣
↪47.444},
    {'Location': 'London', 'Current Weather': 'light rain', 'Max. Temperature␣
↪in Degree Centigrade': 8.2, 'Max. Temperature in Fahrenheit': 46.76}]

df3 = pd.DataFrame.from_dict(df3)
print(df3)

#geeksforgeeks, How to convert Dictionary to Pandas Dataframe?,
#Available from: https://www.geeksforgeeks.org/
↪how-to-convert-dictionary-to-pandas-dataframe/, Acessed on: January 2023.
```

```
     Location   Current Weather  Max. Temperature in Degree Centigrade  \
0   Gravesend     moderate rain                                   8.38
1  Gillingham        light rain                                   8.25
2       Essex  scattered clouds                                   4.26
3  Canterbury        light rain                                   8.91
4       Stoke     moderate rain                                   8.58
5      London        light rain                                   8.20

   Max. Temperature in Fahrenheit
0                          47.084
1                          46.850
2                          39.668
3                          48.038
4                          47.444
5                          46.760
```

4.4.2 Getting the Metadata and some statiscal information about the current weather data dataframe

[47]:
```
df3.info()
df3.describe()
```

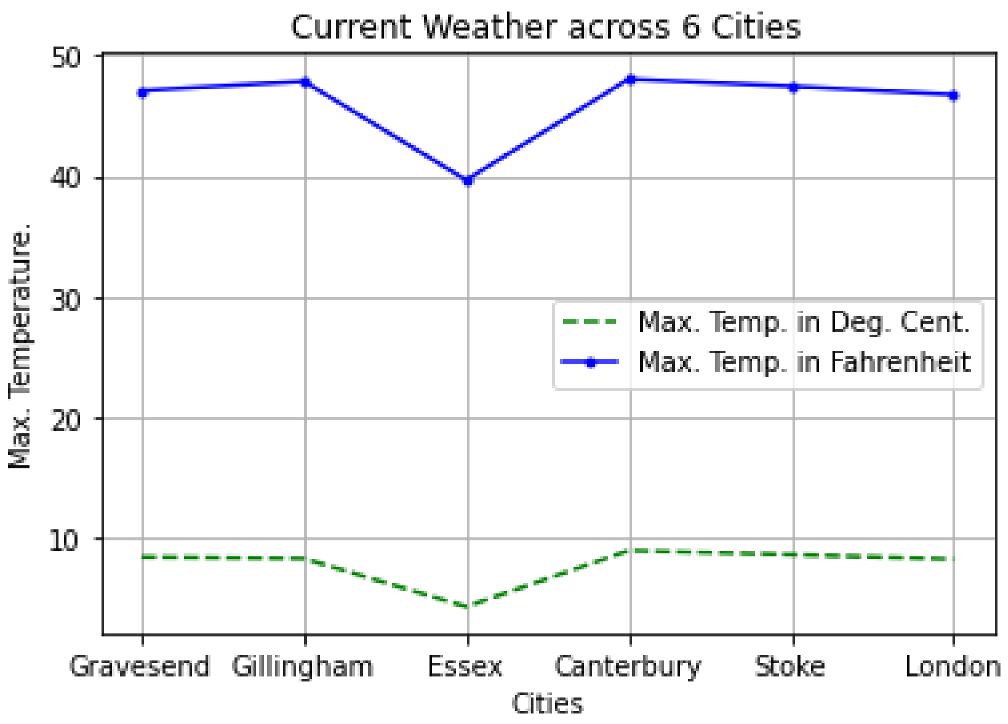
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Location                               6 non-null      object
 1   Current Weather                        6 non-null      object
 2   Max. Temperature in Degree Centigrade  6 non-null      float64
 3   Max. Temperature in Fahrenheit         6 non-null      float64
dtypes: float64(2), object(2)
```

```
memory usage: 320.0+ bytes
```

[47]:
```
       Max. Temperature in Degree Centigrade  Max. Temperature in Fahrenheit
count                               6.000000                        6.000000
mean                                7.763333                       45.974000
std                                 1.735680                        3.124225
min                                 4.260000                       39.668000
25%                                 8.212500                       46.782500
50%                                 8.315000                       46.967000
75%                                 8.530000                       47.354000
max                                 8.910000                       48.038000
```

4.4.3 Plotting of the current weather data

[39]:
```
from matplotlib import pyplot as plt
weather_location = ["Gravesend", "Gillingham", "Essex", "Canterbury", "Stoke",␣
 ↪"London"]
weather_maxtempinDegCent = [8.38, 8.25, 4.26, 8.91, 8.58, 8.20]
weather_maxtempinFahrenheit = [47.084, 47.850, 39.668, 48.038, 47.444, 46.760]
plt.plot(weather_location, weather_maxtempinDegCent, 'g--', label='Max. Temp.␣
 ↪in Deg. Cent.')
plt.plot(weather_location, weather_maxtempinFahrenheit, 'b.-', label='Max. Temp.
 ↪ in Fahrenheit')
plt.xlabel("Cities")
plt.ylabel("Max. Temperature.")
plt.title("Current Weather across 6 Cities")
plt.legend(loc = 'best')
plt.grid()
plt.show()
```



5.0 INTORUCTION OF ACTIVITY 9

The goals of this activity are to:

(1) To introduce how extra libraries can be loaded inside the jupyter notebook.
(2) Get an Idea of sentiment analysis and a worked example

5.1 TASKS:

5.1.1 Downloading of the file: The file was downloaded from the in-class/activity 9 and saved in the downloaded folder of the computer desktop.

5.1.2 Unzipping of the File: It was unzipped/extracted by right clicking on the zipped foler, then selected ‘extract to’ by specifying the folder where it should be saved for easy access (i.e Data Intelligence Programming Folder).

5.1.3 Uploading of the File to Jupyter Notebook: The file was uploaded to Jupyter notebook by expanding the ‘file browser menu’. Under the menu, the upload button was selected which opened up a dialogue box that helped to navigate the location of the extracted file.

Note: After uploading the file to the jupyter notebook, there was no notable happening with regards to the uploaded file within the jupyter notbook except seeing it being uploaded to the file menu.

5.2 WHAT IS VADER: The full meaning for the acronym VADER is (Valence Aware Dictionary for sEntiment Reasoning). It’s a model that was introduced 2014 which is used to carry out text sentiment analysis. Pio (2017)

5.3 WHAT DOES VADER DO: VADER carries out Text Sentiment Analysis (TSA) by calculating the sentiment score of an input text. TSA is a measure of both the sensitivity of text to both polarity (positive/negative) and intensity (strength) of peoples' feeling. For example, considering a statement by a man about a woman: 'I love her, but I hate her attitude'. From this statement example, there is a sense of feeling that the statement implied i.e the first part of the statement conveyed a positive feeling while the second part sends a negative feeling. VADER uses a human-centric approach for text sentiment analysis, combining qualitative analysis and empirical validation by using human raters and the wisdom of the crowd. Pio (2017)

5.4 TECHNIQUES/APPROACHES: VADER uses two approaches to carry out TSA viz: the lexical approach and the machine learning approach. On the one hand, the lexical approach maps words, texts and or statement to sentiments. Sentiments can be either Categorial (Positive, Neutral, Positive) or Numerical (a range of score or intensities). VADER is an example of TSA which uses dictionary of emotions to analyse and group texts into Categoical or numerical values. This approach do not need to train a model using labeled data because everything is available in the dictionary of emotions or sentiment. On the other hand, the Machine Learing (ML) techniques uses trained labelled data to compare between old and new text inputs. The latter have advantage over the former in that the latter achieve greater result of prediction/classification when compared with the result of the former in a large . However, unlike lexical approaches, the ML techniques requires previously labeled data in order to the desired result. Pio (2017).

5.5 Explanation of different symbols, operators and Methods: At an appropriate point explain what the following do (with appropriate references if needed):

(a) ! is an operator that is used in a Jupyter Notebook or other interactive Python environments to prefix a command that should be run in the command line, rather than in Python while pip is a command-line tool used to install and manage packages written in Python. It stands for "Pip Installs Packages."

(b) % is a symbol that is used to represent a command in a Jupyter Notebook. It is known as the "magic" operator or magic command and it's provided by the Ipython kernel.

(c) iloc: Is an attribute of a Pandas DataFrame that is used to index data in the DataFrame using integer-based location. It stands for "integer location". It is used to select rows and columns of a DataFrame based on their integer index position. For example, df.iloc[0] would select the first row of the DataFrame df.

(d) the_data.head(): Is a function used to return the number of rows specified from the top of a dataframe. The output also comes with the corresponding columns.

(e) the_data['Compound score']=score_compound: It is one of sentiment score returned by the VADER library when running sentiment analysis. It's the normalized compound score which calculates the sum of all lexicon ratings and takes values from -1 to 1.

```
[ ]: 5.6 Exploration of some pandas code on tweet_activity.csv
```

import pandas as pd df2 = pd.read_csv("tweet_activity.csv") df2.info() df2.describe()

```
[115]: df2.iloc[0]
```

```
[115]: Tweet text    Digging into Big Provenance (with SPADE) https…
       Name: 0, dtype: object
```

```
[116]: df2.iloc[2]
```

```
[116]: Tweet text    Computational Thinking for Professionals https…
       Name: 2, dtype: object
```

```
[117]: df2.iloc[:]
```

```
[117]:
                                                  Tweet text
       0    Digging into Big Provenance (with SPADE) https…
       1    The Road Ahead for Augmented Reality https://t…
       2    Computational Thinking for Professionals https…
       3    What makes university students steer clear of …
       4    Caught on camera: Using AI to combat street cr…
       ..                                                …
       218  Researchers develop interface for underwater r…
       219  How do you deradicalise an incel? https://t.co…
       220  How does 'normal' Internet browsing look today…
       221           @cpjobling62 Hi  welcome back to twitter
       222  Saving seaweed with machine learning https://t…

       [223 rows x 1 columns]
```

```
[118]: df2.iloc[-6:]
```

```
[118]:
                                                  Tweet text
       217  Earned a badge today Thank you @STEMLearningUK…
       218  Researchers develop interface for underwater r…
       219  How do you deradicalise an incel? https://t.co…
       220  How does 'normal' Internet browsing look today…
       221           @cpjobling62 Hi  welcome back to twitter
       222  Saving seaweed with machine learning https://t…
```

(d) the_data.head()

```
[121]: df2.head()
```

```
[121]:
                                                Tweet text
       0  Digging into Big Provenance (with SPADE) https…
       1  The Road Ahead for Augmented Reality https://t…
       2  Computational Thinking for Professionals https…
       3  What makes university students steer clear of …
       4  Caught on camera: Using AI to combat street cr…
```

```
[122]: df2.head(5)
```

```
[122]:
                                                Tweet text
       0  Digging into Big Provenance (with SPADE) https…
```

```
1  The Road Ahead for Augmented Reality https://t…
2  Computational Thinking for Professionals https…
3  What makes university students steer clear of …
4  Caught on camera: Using AI to combat street cr…
```

References

1. Amanda Iglesias Moreno, 2020. 15 things you should know about Dictionaries in Python https://towardsdatascience.com/15-things-you-should-know-about-dictionaries-in-python-44c55e75405c Published in Towards Data Science Accessed on: December 2022.

2. Datacamp 2022 (Online) Pandas Tutorial: DataFrames in Python, Available from: https://www.datacamp.com/tutorial/pandas-tutorial-dataframe-python Accessed on: January 2023.

3. Geeksforgeeks, How to convert Dictionary to Pandas Dataframe?, Available from: https://www.geeksforgeeks.org/how-to-convert-dictionary-to-pandas-dataframe/, Acessed on: January 2023.

4. George, S 2021 (online) 5 Advanced Features of Pandas and How to Use Them, Available from: https://www.kdnuggets.com/2019/10/5-advanced-features-pandas.html Accessed on: January 2023.

5. John, D. 2016, (online) what is the 'ro' in plot() mean? READ THE HELP! The important part is here: Available from: https://www.mathworks.com/matlabcentral/answers/282940-what-is-the-ro-in-plot-mean Accessed on: January 2023.

6. Melissa, B. 2022 (Online) 12 Python Data Visualization Libraries to Explore for Business Analysis, Available from: https://mode.com/blog/python-data-visualization-libraries/ Accessed on: January 2023.

7. Metric Conversions (Online), Available from: https://www.metric-conversions.org/temperature/celsius-to-fahrenheit.htm, Accessed on: January 2023.

8. Numpy 2008-2022 (Online) What is NumPy?, Available from: https://numpy.org/doc/stable/user/whatisnumpy.html Accessed on: January 2023.

9. OpenWeather (Online) Weather API, Available from: https://openweathermap.org/api/, Accessed on: January 2023

10. OpenWeather (Online) Current Weather Data, Available from: https://openweathermap.org/current, Accessed on: January 2023

11. Pio C. 2017 (Online) VADER Sentiment Analysis Explained, Available from: https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd, Date Accessed: January 2023.

12. Priya P. List Operations in Python: Introduction to List Operations in Python, Available from: https://www.educba.com/list-operations-in-python/

13. ProgramssBuzz 2021 (Online) Characteristics of Pandas Dataframes, Available from: https://www.programsbuzz.com/article/characteristics-pandas-dataframes Accessed on: January 2023.

14. Programmiz Python list(), Available from: https://www.programiz.com/python-programming/methods/built-in/list Accessed on: December 2022.

15. Quora 2022 (Online) What are the importance of List in Python, Available from: https://www.quora.com/What-is-the-importance-of-list-in-python, Accessed on: January 2023.

16. RealPython (Online) Variables in Python, Available from: Real Python (https://realpython.com/python-variables/), Accessed on: January 2023.

17. Scott, T 2022. Principles of Data Intelligence Programming (P18719), Availabel from: Objects and List things. Session 4, pgs 2, 8.

18. Stackoverflow, 2020 (online) What are 25%,50%,75% values when we describe a grouped dataframe?, Available from: https://stackoverflow.com/questions/57869926/what-are-25-50-75-values-when-we-describe-a-grouped-dataframe Accessed on: January 2023.

19. Stackoverflow 2022 (Online) What is the difference between ! and % in Jupyter notebooks?, AVailable from: https://stackoverflow.com/questions/45784499/what-is-the-difference-between-and-in-jupyter-notebooks, Accessed on January 2023.

20. W3 Schools, 2022 (Online) Python Collections (Arrays), Available from: https://www.w3schools.com/python/python_lists.asp#:~:text=There%20are%20four%20collection%20data
Acessed in: January 2023.

21. W3 Schools Matplotlib Labels and Title, Available from: https://www.w3schools.com/python/matplotlib_labels.asp Accessed on: January 2023.

22. W3 Schools Matplotlib Markers, Available from: https://www.w3schools.com/python/matplotlib_markers.as
Accessed on: January 2023.plt.bar

23. W3 Schools 2022 Python Lists, Available from: https://www.w3schools.com/python/python_lists.asp
Accessed on: January 2023.

24. W3 Schools 2022 Python For Loops https://www.w3schools.com/python/python_for_loops.asp

25. W3 Schools Seaborn, Available from: https://www.w3schools.com/python/numpy/numpy_random_seaborn
Accessed on: January 2023.

26. W3 Schools 2022, Python Set pop() Method Available from: https://www.w3schools.com/python/ref_set_pop.asp, Accessed on: January 2023.