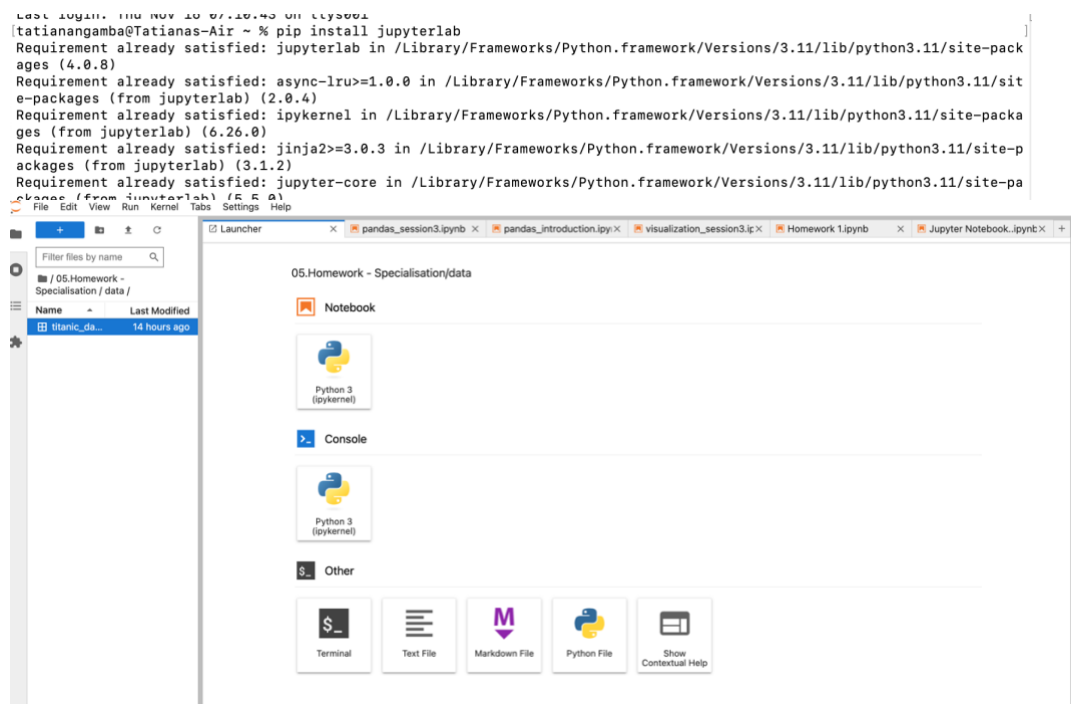


Homework Week 1

Question 1 [20 points]

In no more than 500 words, explain the differences between a Jupyter Notebook (.ipynb) extension) and a Python file (.py extension). Focus on the advantages and disadvantages of one or the other, particularly for data analysts/scientists. You are encouraged to use examples and screenshots of the different scripts to support your arguments.

Jupyter Notebook (.ipynb) is a web-based interactive development environment for notebooks, data and code. It is used for writing and running programming languages such as Python code. It can be easily installed via pip.



Advantages

1. **User-friendly**- provides a seamless user experience by allowing users to quickly execute code and efficiently evaluate the results. This iterative approach is essential

```
[44]: import pandas as pd
import matplotlib.pyplot as plt
# !pip install seaborn

import seaborn as sns

data = pd.read_csv('data/movie_metadata.csv')

data.head(5)
```

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.2	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	...	num_user_for_reviews	language	country
0	0	0	0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	...	3054.0	English	USA
1	1	1	1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	...	1238.0	English	USA
2	2	2	2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	...	994.0	English	USA
3	3	3	3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	...	2701.0	English	USA
4	4	5	5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	...	738.0	English	USA

5 rows x 31 columns

for data analytics because it makes it easy for users to test new ideas by combining code, visualisations, equations and analysing large sets of data while documenting their results.

2. **Accommodate multiple coding languages**- the system can accommodate multiple coding languages and supports several coding languages, including Python, SQL, and Julia.

```
[44]: import pandas as pd
import matplotlib.pyplot as plt
# !pip install seaborn

import seaborn as sns

data = pd.read_csv('data/movie_metadata.csv')

data.head(5)
```

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	...	num_user_for_reviews	language	cou
0	0	0	0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	...	3054.0	English	
1	1	1	1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	...	1238.0	English	
2	2	2	2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	...	994.0	English	
3	3	3	3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	...	2701.0	English	
4	4	5	5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	...	738.0	English	

5 rows x 31 columns

3. **Markdown Support**- This functionality enables users to combine their codes, explanations, and visualizations. This also allows them to present their findings in a clean, formatted notebook in a consistent manner.

```
# Homework 1 - Tatiana Ngamba

### Question 5 [50 points] Each sub-question is worth 10 points.

[ ]: # Using the titanic dataset which you can read into your notebook using the following code,
# import pandas as pd
# titanic=pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-
# data/master/titanic.csv')
# answer the following questions:
```

▼ A. How many columns and rows does the data have?

```
[96]: import pandas as pd

titanic=pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv')

titanic
```

4. **Versatility** By allowing a wide range of tasks to be performed. The range of tasks varies from data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning and collaboration & sharing.

```
[7]: import matplotlib.pyplot as plt

x = [1,2,3,4,5]

y1 = [50,40,70,80,20]
y2 = [80,20,20,50,60]
y3 = [70,20,60,40,60]
y4 = [80,20,20,50,60]

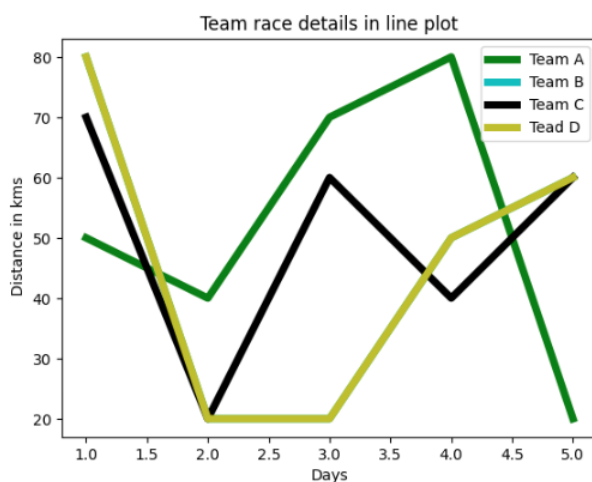
plt.plot(x,y1,'g',label='Team A', linewidth=5)
plt.plot(x,y2,'c',label='Team B', linewidth=5)
plt.plot(x,y3,'k',label='Team C', linewidth=5)
plt.plot(x,y4,'y',label='Tead D', linewidth=5)

plt.title('Team race details in line plot')
plt.ylabel('Distance in kms')
plt.xlabel('Days')

plt.legend()

# Various lines present in the graph have unique colors, and each of them denotes details of different teams.
# The line representing Team B is overwritten by the line representing Team D,
# since both vehicles have covered the same distance in their respective days.
```

```
[7]: <matplotlib.legend.Legend at 0x1180a8d10>
```



5. **Privacy** it protects **users' privacy** by allowing them to execute code on their local workstation, it runs locally by default at <http://localhost:8888>.

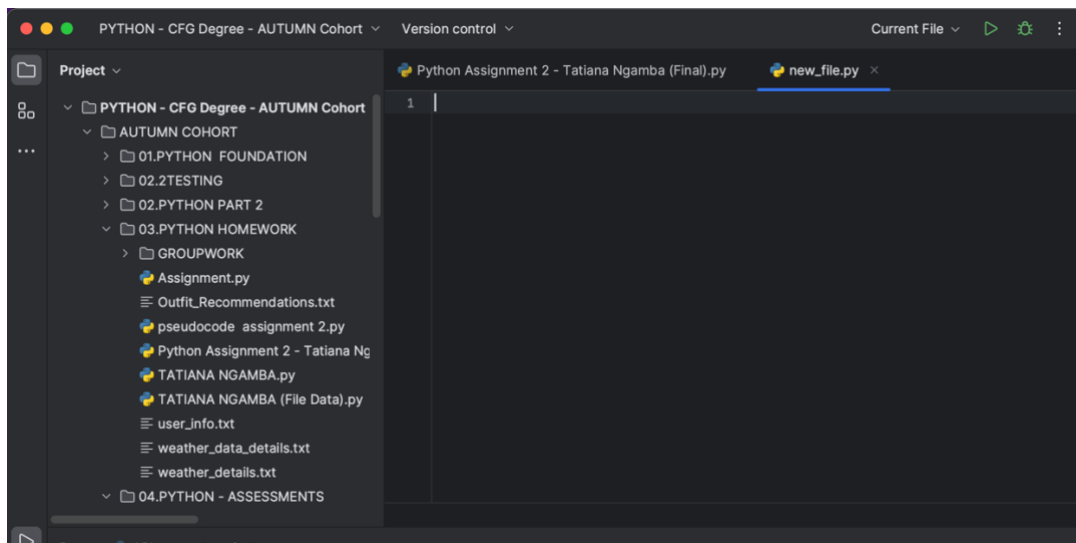
localhost:8888/lab/tree/01.pandas/pandas_introduction.ipynb

2.CODE ACADEM... 3.Tatiana-Ngamba... 04.JupyterLab NOTION Data 2 - Autumn...

Disadvantages

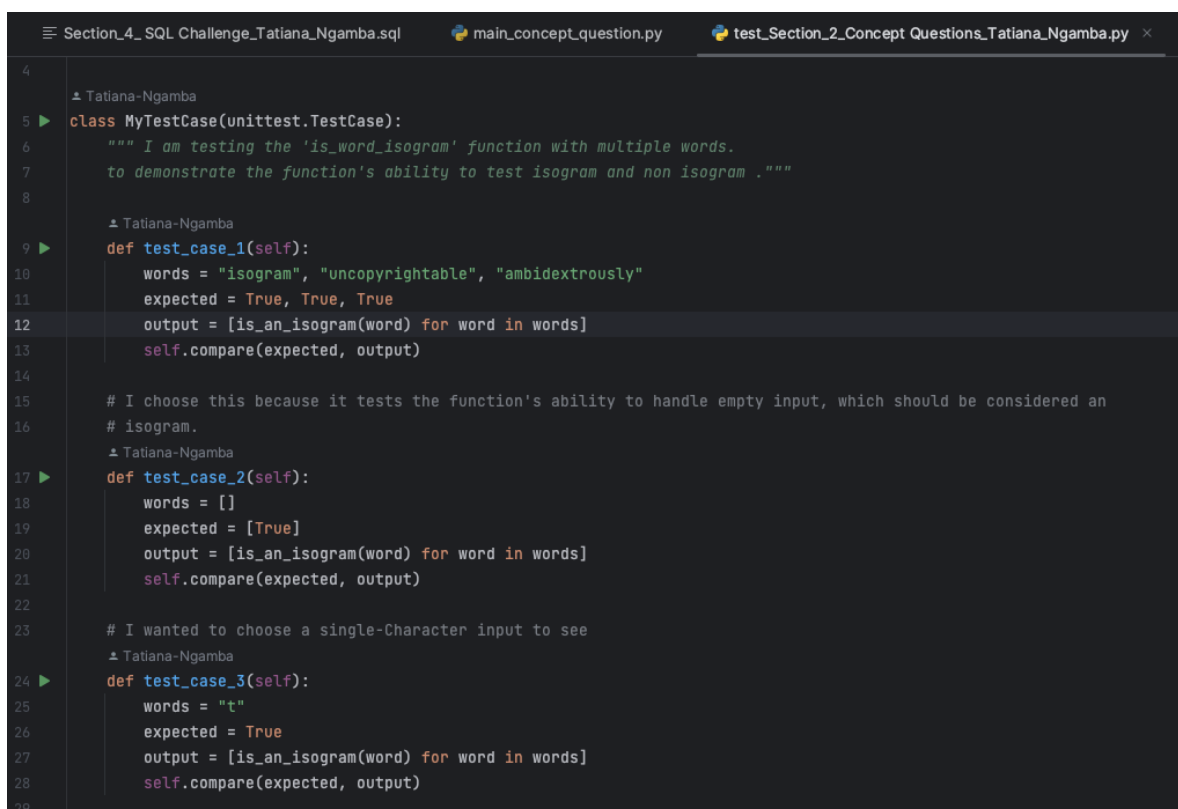
1. **Version Control:** Jupyter Notebooks are stored in JSON format, which makes version control more challenging than traditional Python scripts.
2. **Testing:** Testing code in Jupyter Notebooks can be more challenging and not as straightforward compare to testing Python scripts in .py files.

Python file (.py) contains a Python programme or script. PY files provide data analysts and scientists with the tools to build code in an organised and consistent way.

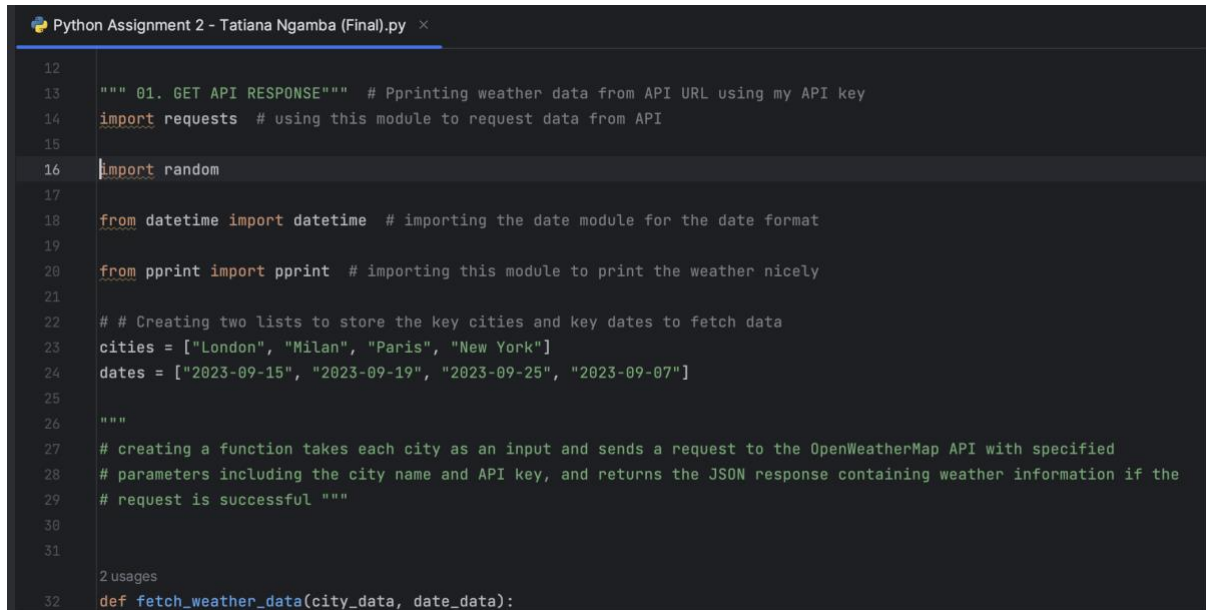


Advantages

1. **Testing and Automation** Python files make unit testing more manageable by allowing for systematic testing of functions and modules. The added benefit of Python scripts is their seamless integration into automated workflows and scheduled tasks.



2. **Modular Code Organisation** Python files are an excellent way for data scientists and data analysts to organise their codes in a modular manner. For example, functions and classes can be separated into different files, which contributes to code maintenance, readability, and team collaboration.



```

12
13 """ 01. GET API RESPONSE""" # Pprinting weather data from API URL using my API key
14 import requests # using this module to request data from API
15
16 import random
17
18 from datetime import datetime # importing the date module for the date format
19
20 from pprint import pprint # importing this module to print the weather nicely
21
22 # # Creating two lists to store the key cities and key dates to fetch data
23 cities = ["London", "Milan", "Paris", "New York"]
24 dates = ["2023-09-15", "2023-09-19", "2023-09-25", "2023-09-07"]
25
26 """
27 # creating a function takes each city as an input and sends a request to the OpenWeatherMap API with specified
28 # parameters including the city name and API key, and returns the JSON response containing weather information if the
29 # request is successful """
30
31
32 2 usages
32 def fetch_weather_data(city_data, date_data):

```

3. **Version Control** Python files are perfectly compatible with Git and other version control systems. This feature guarantees code versioning and collaboration, as well as the ability to revert to previous versions when necessary.
4. **Production Code** Python files are ideal to use in production. Data scientists may use frameworks for their analysis. To make predictions available over HTTP, data scientists frequently deploy machine learning models as RESTful APIs using Flask.

Disadvantages

1. **Not Interactive:** When it comes to interactivity, Python files are typically less efficient than Jupyter Notebooks. Python scripts are often implemented as a whole, which can hinder the exploration process and slow things down.
2. **Not Visual:** One disadvantage of using Python files to share your results is that it may not be as convenient for stakeholders who are not familiar with coding. In comparison, Jupyter Notebook is a more comprehensive and user-friendly narrative that can be easily shared with a wider audience.

Question 2 [10 points]

What is the difference between a Pandas DataFrame and a Pandas series. Show an example of how you create each of them.

A Pandas Series is a one-dimensional array that can contain any data type. Think of it like a single column within a table, where each element has its own index. By using the index, you can easily access any element within the Pandas Series. A Pandas series index needs to be defined. A series can be used to represent a single variable or characteristic.

Whereas a Pandas DataFrame is a two-dimensional data structure, think of it like SQL tables, Excel spreadsheets, or a table with rows and columns. You are also able to name your own indexes by using the index argument. A Pandas DataFrame index can be optional. DataFrames are used for larger datasets with many variables.

Example of how to create a Pandas DataFrame

- I have created a Pandas DataFrame called **Results**. The DataFrame has two columns, "Students" and "grades" with corresponding values.
- Output - As you can see each row represents a student, and the columns contain their names and respective grades.

```
: # importing pandas
import pandas as pd

# Creating a Pandas DataFrame with a custom index
data = {
    "Students": ["Tatiana", "Chris", "Jordan", "Cynthia"],
    "grades": [100, 85, 25, 99]
}

#loading data into a DataFrame object
results = pd.DataFrame(data)

# Displaying the DataFrame
results
```

```
:   Students  grades
0   Tatiana    100
1    Chris     85
2    Jordan     25
3  Cynthia     99
```

Example of how to create a Pandas Series

- I have created a Pandas Series named **grades & student grades** from a list.
- This Series is a one-dimensional labelled array, and has a default index of '0, 1, 2, 3' that is automatically allocated.

```
: # importing pandas
import pandas as pd

# Creating a Series from a list
grades = pd.Series([100, 85, 25, 99])

# Displaying the Series
grades
```

```
: 0    100
1     85
2     25
3     99
dtype: int64
```

```
import pandas as pd

# Creating a Pandas Series from a list
grades_series = pd.Series([100, 85, 25, 99], name='Student Grades')

# Displaying the Series
grades_series
```

0	100
1	85
2	25
3	99

Name: Student Grades, dtype: int64

Question 3 [10 points]

Starting from the argument that a Pandas DataFrame represents rectangular data, use the internet and other resources to describe in no more than a few sentences the difference between rectangular and non-rectangular data.

Rectangular data is arranged in a tubular format with rows and columns, similar to a spreadsheet or relational database table (SQL). And it is commonly represented as a Pandas DataFrame. Each row corresponds to a given record within the dataset, while each column represents a variable.

Whereas non-rectangular data, are not logically grouped into rows and columns. Non-rectangular data is difficult to present in a conventional tabular format because it may include layered structures, uneven forms, or hierarchical relationships.

Example of a 'non-rectangular data' in a JSON format:

```
[20]: # Example of a non-rectangular data in a JSON format
import json

# Your JSON data
json_data = '''
{
  "person": {
    "name": "Joe Bloggs",
    "age": 45,
    "address": {
      "street": "32 Avenue Street",
      "city": "New-York",
      "zip": "54345"
    },
    "contacts": [
      {
        "type": "email",
        "value": "joe.bloggs@example.com"
      },
      {
        "type": "phone",
        "value": "566-1234-08"
      }
    ]
  }
}
'''

# Load JSON into a Python variable
data = json.loads(json_data)
data
```

```
[20]: {'person': {'name': 'Joe Bloggs',
  'age': 45,
  'address': {'street': '32 Avenue Street',
  'city': 'New-York',
  'zip': '54345'},
  'contacts': [{'type': 'email', 'value': 'joe.bloggs@example.com'},
  {'type': 'phone', 'value': '566-1234-08'}]}}
```

Example of a 'rectangular' data using DataFrame for each category:

```
# Example of a 'rectangular' data using DataFrame for each category
import pandas as pd
```

```
# Creating DataFrames for each category
```

```
produce_data = {
    'item': ['apples', 'bananas'],
    'price': [3.99, 0.49]
}
```

```
produce = pd.DataFrame(produce_data)
```

```
condiments_data = {
    'item': ['peanut_butter', 'mayonnaise'],
    'price': [2.18, 3.89]
}
```

```
condiments = pd.DataFrame(condiments_data)
```

```
canned_goods_data = {
    'item': ['black_beans', 'tomato_sauce'],
    'price': [0.99, 0.69]
}
```

```
canned_goods = pd.DataFrame(canned_goods_data)
```

```
grains_data = {
    'item': ['bread', 'pasta'],
    'price': [2.99, 1.99]
}
```

```
grains = pd.DataFrame(grains_data)
```

```
dairy_data = {
    'item': ['milk', 'butter'],
    'price': [2.73, 2.57]
}
```

```
dairy = pd.DataFrame(dairy_data)
```

```
# Combining individual DataFrames into a single DataFrame
```

```
groc_df = pd.concat([produce, condiments, canned_goods, grains, dairy], keys=['produce', 'condiments', 'canned_goods', 'grains', 'dairy'])
```

```
# Resetting index for a clean DataFrame
```

```
groc_df.reset_index(level=0, inplace=True)
```

```
groc_df.rename(columns={'level_0': 'category'}, inplace=True)
```

```
# Displaying the combined DataFrame
```

```
print(groc_df)
```

	category	item	price
0	produce	apples	3.99
1	produce	bananas	0.49
0	condiments	peanut_butter	2.18
1	condiments	mayonnaise	3.89
0	canned_goods	black_beans	0.99
1	canned_goods	tomato_sauce	0.69
0	grains	bread	2.99
1	grains	pasta	1.99
0	dairy	milk	2.73
1	dairy	butter	2.57

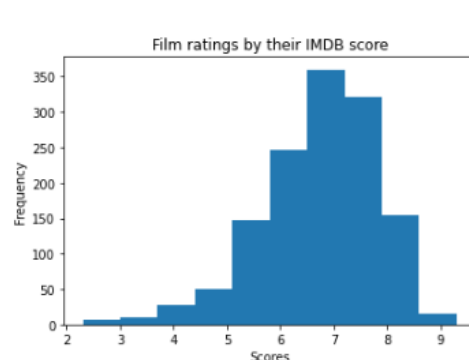
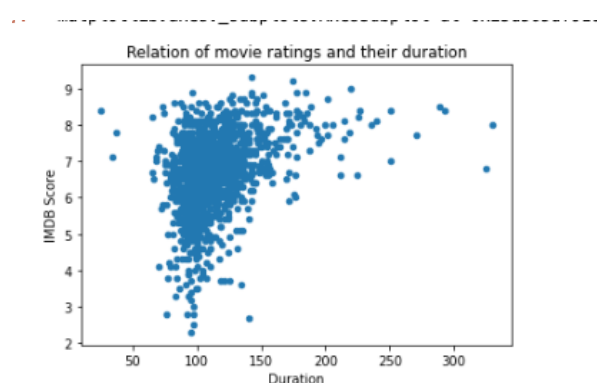
```
[ ]:
```

Question 4 [10 points]

Starting from the data visualisation usage from Session 3, give examples of when figures could be:

a. Of use to the data scientist to identify patterns in the data/highlight the important parts of a data set.

Exploratory Data Analysis - involves the use of visualizations, such as **scatter plots** and **histograms**, by data scientists to delve into datasets. These visual tools aid in analysing data, unveiling patterns, and summarising key characteristics, facilitating a comprehensive understanding of the dataset's underlying structure and insights.



Time series Analysis & and Forecasting - involves the use of visualizations, such as **Time series plots** by data scientists to visualising trends and patterns over time. These visual tools play a crucial role in forecasting, segmentation, classification, descriptive analysis, and intervention analysis within time-dependent datasets.



b. Tell a story and create business presentations.

Sales Performance Analysis using Time Series Plots:

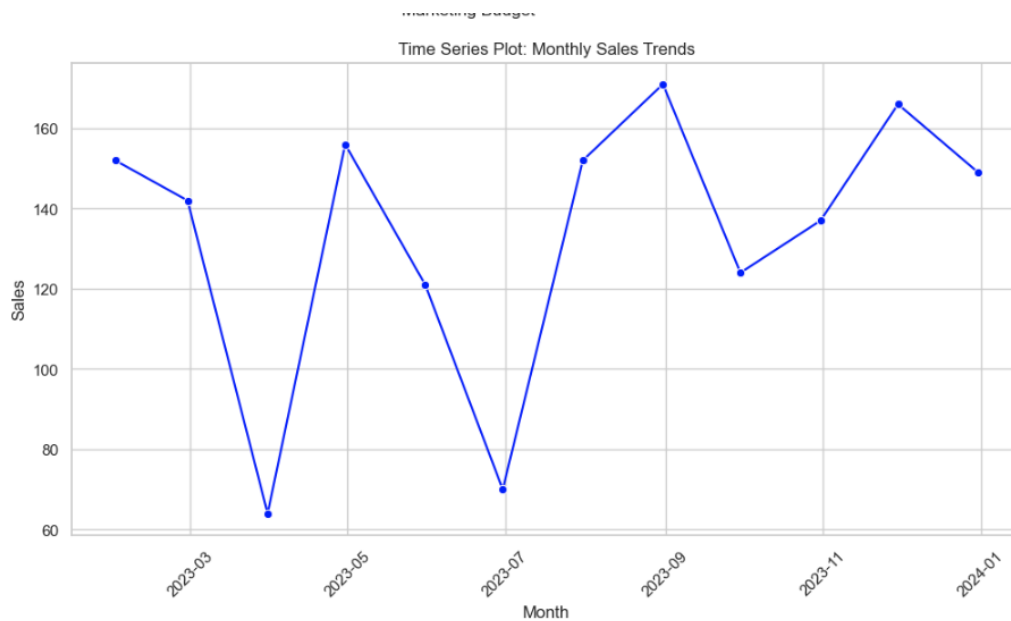
Task: A fashion brand would like identify trends and patterns throughout the year for each month. To understand how the marketing budget could be utilised to drive more sales for the following year.

Telling a story:

- To visualize sales trends
- To identify trend in the brand growth and decline
- To observe any consistency within the patterns to provide data driven insights and inform the marketing strategies.

Creating a business presentations & implementation:

- X-axis will display monthly time series from January through to December .
- Y-axis will display the sales figures for each month.
- Each line will represent the sales performance over time.



Question 5 [50 points] Each sub-question is worth 10 points.

Using the titanic dataset which you can read into your notebook using the following code, import pandas as pd

```
titanic=pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv')
```

answer the following questions:

(PLEASE SEE Jupyter Notebook)

a. How many columns and rows does the data have?

- 891 Rows
- 15 Columns

891 rows x 15 columns

1:

b. Get a sense of your data and find the min, max, and count/mean depending on the data type.

c. Give an overview (code and an explanation) of all missing values in the data.

d. Delete the rows where you do not have information about the age of the person. Then group the passengers in a 10 year age range (for example, you can do something like 0 – 10, 11 – 20, 21 – 30, etc).

e. For each age category created in d), find out how many passengers are female/male, and how many travelled in each class.