

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: Rash Paul Wadhwa |
| Course Date: 21/08/2025 |
|  |

**Table of contents**

[Day 2: Task 1 3](#_Toc1765240926)

[Day 3: Task 1 3](#_Toc873857632)

[Exercise 1: Loading and Exploring the Data 4](#_Toc1363711086)

[Exercise 2: Indexing and Slicing 4](#_Toc1679584900)

[Exercise 3: Data Manipulation 4](#_Toc998449442)

[Exercise 4: Aggregation and Grouping 5](#_Toc267294297)

[Exercise 5: Advanced Operations 5](#_Toc1232452443)

[Exercise 6: Exporting Data 6](#_Toc1251390377)

[Exercise 7: If finished early try visualising the results 6](#_Toc66584577)

[Day 4: Task 1 7](#_Toc1452452084)

[Day 4: Task 2 8](#_Toc934594145)

[Course Notes 8](#_Toc1227711066)

[Additional Information 9](#_Toc1836543406)

# Day 2: Task 1

It is a common software development interview question to create the below with a certain programming language. Create the below using Python syntax, test it and past the completed syntax and output below.

FizzBuzz:

Go through the integers from 1 to 100.

If a number is divisible by 3, print "fizz."

If a number is divisible by 5, print "buzz."

If a number is both divisible by 3 and by 5, print "fizzbuzz."

Otherwise, print just the number.

|  |  |
| --- | --- |
| Paste your completed work to the right |  |

# **Day 3: Task 1**

Download the ‘student.csv’, complete the below exercises as a group and paste your input and output. Although this is a group activity, everyone should have the below answered so it supports your portfolio:

### **Exercise 1: Loading and Exploring the Data**

1. Question: "Write the code to read a CSV file into a Pandas DataFrame."
2. Question: "Write the code to display the first 5 rows of the DataFrame."
3. Question: "Write the code to get the information about the DataFrame."
4. Question: "Write the code to get summary statistics for the DataFrame."

|  |
| --- |
| 1) df = pd.read\_csv('student.csv')  2) df.head()  3) df.info()  4) df.describe() |

### **Exercise 2: Indexing and Slicing**

1. Question: "Write the code to select the 'name' column."
2. Question: "Write the code to select the 'name' and 'mark' columns."
3. Question: "Write the code to select the first 3 rows."
4. Question: "Write the code to select all rows where the 'class' is 'Four'."

|  |
| --- |
| 1. one\_col   = df.loc[:, 'name']   print(one\_col.head())   1. two\_cols  = df.loc[:, ['name', 'mark']]   print(two\_cols.head())   1. rows = df.iloc[0:4]   print(rows\_0\_2.head())   1. class\_four = df[df["class"] == ‘Four’] |
|  |

### **Exercise 3: Data Manipulation**

1. Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."
2. Question: "Write the code to rename the 'mark' column to 'score'."
3. Question: "Write the code to drop the 'passed' column."

|  |
| --- |
| # Make a copy of the original DataFrame  df\_transformed = df.copy()   1. # Create a new column: 'pass\_fail' (Pass if mark >= 60)   df\_transformed['passed'] = df\_transformed['mark'].apply(lambda x: 'Pass' if x >= 60 else 'Fail')  print(df\_transformed[‘passed’].head())   1. # Rename the columns 'mark' to 'score'   df\_renamed.rename(columns={'mark': 'score'}, inplace=True)  # Print the new column names  print("Columns after renaming:", df\_renamed.columns)   1. # Drop the 'passed’ column   df\_transformed = df\_transformed.drop('pass\_fail', axis=1)  print(df\_transformed.head()) |
|  |

### **Exercise 4: Aggregation and Grouping**

1. Question: "Write the code to group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
2. Question: "Write the code to count the number of students in each class."
3. Question: "Write the code to calculate the average mark for each gender."

|  |
| --- |
| 1. # Group by 'gender' and calculate the average mark   gender\_avg\_mark = df.groupby('class')['mark'].mean().sort\_values(ascending=False)  print("Average mark grouped by gender:")  display(gender\_avg\_mark)   1. # “count the number of students in each class."   Number\_of\_students = df.groupby('gender')['mark'].agg(['count'])  print("\Number of students in each class:")  display(Number\_of\_students )   1. # ‘Calculate the average mark for each gender.’   gender\_summary = df.groupby('gender')['mark'].agg(['count', 'mean'])  print("\nSummary statistics for score grouped by gender:")  display(gender\_summary) |

### **Exercise 5: Advanced Operations**

1. Question: "Write the code to create a pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values."
2. Question: "Write the code to create a new column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."
3. Question: "Write the code to sort the DataFrame by 'mark' in descending order."

|  |
| --- |
| # Create the pivot table  pivot\_table = df.pivot\_table(values="score", index="class", columns="gender", aggfunc="mean")  print(pivot\_table) |

### **Exercise 6: Exporting Data**

1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."

|  |
| --- |
| # Save the DataFrame with the new 'grade' column to a new CSV file  df\_transformed.to\_csv('students\_with\_grade.csv', index=False) |

### **Exercise 7: If finished early try visualising the results**

|  |
| --- |
| import matplotlib.pyplot as plt  # Bar graph of student counts per class using only pandas  df['class'].value\_counts().plot.bar().figure.show()    # Line plot of student marks using only pandas  df['mark'].plot(title='Line Plot of Student Marks').figure.show()    import numpy as np  # Create a DataFrame with random data  df = pd.DataFrame({      'x': np.random.randint(1, 100, size=50),      'y': np.random.randint(1, 100, size=50)  })  # Create scatter plot using only pandas  df.plot.scatter(x='x', y='y', title='Scatter Plot of Random Values').figure.show() |

# **Day 4: Task 1**

Using the ‘GDP (nominal) per Capita.csv’ which can be downloaded from the shared Folder, complete the below exercises and paste your input and output. Work individually, but we will work and support each other in the room.

* Read and save the ‘GDP (nominal) per Capita’ data to a data frame called “df” in Jyputer notebook
* Print the first 10 rows
* Print the last 5 rows
* Print ‘Country/Territory’ and ‘UN\_Region’ columns

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| import pandas as pd  df = pd.read\_csv("GDP (nominal) per Capita.csv",encoding= 'unicode\_escape',  index\_col=0)  df.head(10)  df.tail()  display(df[['Country/Territory', 'UN\_Region']])  I converted this to a table.   | **index** | **Country/Territory** | **UN\_Region** | | --- | --- | --- | | **1** | Monaco | Europe | | **2** | Liechtenstein | Europe | | **3** | Luxembourg | Europe | | **4** | Ireland | Europe | | **5** | Bermuda | Americas | | **6** | Norway | Europe | | **7** | Switzerland | Europe | | **8** | Singapore | Asia | | **9** | Isle of Man | Europe | | **10** | Cayman Islands | Americas | | **11** | Qatar | Asia | | **12** | United States | Americas | | **13** | Iceland | Europe | | **14** | Channel Islands | Europe | | **15** | Faroe Islands | Europe | | **16** | Denmark | Europe | | **17** | Australia | Oceania | | **18** | Netherlands | Europe | | **19** | Greenland | Americas | | **20** | Austria | Europe | | **21** | Israel | Asia | | **22** | Sweden | Europe | | **23** | Finland | Europe | | **24** | Belgium | Europe | | **25** | San Marino | Europe |   Show 25 per page |

# **Day 4: Task 2**

Back with ‘GDP (nominal) per Capita’. As a group, import and work your way through the Day\_4\_Python\_Activity.ipynb notebook which can be found on the shared Folder. There are questions to answer, but also opportunities to have fun with the data – paste your input and output below.

Once complete, and again as a group, work with some more data and have some fun –there is no set agenda for this section, other than to embed the skills developed this week. Paste your input and output below and upon return we’ll discuss progress made.

[Additional data found here.](https://justit831-my.sharepoint.com/:f:/g/personal/danpe_justit_co_uk/Er0ybU9i0AZKiuGaCWZyj2ABoqKD23zwLGdJf3WlaixpRA?e=QVj2Bs)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Which countries below average by IMF world estimate?  # Calculate the average IMF Estimate  average\_imf\_estimate = df['IMF\_Estimate'].mean()  # Filter for countries with IMF Estimate below the average  below\_average\_imf = df[df['IMF\_Estimate'] < average\_imf\_estimate]  # Print the result  print("Countries with IMF Estimate below the average", below\_average\_imf)  Which country has highest UN Estimate?  highest\_un\_estimate = df.loc[df['UN\_Estimate'].idxmax()]  display(highest\_un\_estimate)  Which country has highest Worlbank Estimate?  highest\_worldbank\_estimate = df.loc[df['WorldBank\_Estimate'].idxmax()]  display(highest\_worldbank\_estimate)   |  |  | | --- | --- | | **Country/Territory** | Monaco | | **UN\_Region** | Europe | | **IMF\_Estimate** | 0 | | **IMF\_Year** | 0 | | **WorldBank\_Estimate** | 234316 | | **WorldBank\_Year** | 2021 | | **UN\_Estimate** | 234317 | | **UN\_Year** | 2021 |   Which country has highest IMF Estimate?  highest\_imf\_estimate = df.loc[df['IMF\_Estimate'].idxmax()]  display(highest\_imf\_estimate)   |  |  | | --- | --- | | **Country/Territory** | Luxembourg | | **UN\_Region** | Europe | | **IMF\_Estimate** | 132372 | | **IMF\_Year** | 2023 | | **WorldBank\_Estimate** | 133590 | | **WorldBank\_Year** | 2021 | | **UN\_Estimate** | 133745 | | **UN\_Year** | 2021 |   sns.barplot(x="UN\_Region", y="WorldBank\_Estimate", data=df, errorbar=None)  plt.title("Average World Bank Estimate by UN Region")  plt.xticks(rotation=45, ha='right')  plt.show()    Filling 0 Values by average  # Calculate the average of 'Worldbank\_Estimate' and 'UN\_Estimate' columns  # Calculate the mean of each estimate column, ignoring NaN values  mean\_imf = df['IMF\_Estimate'].mean()  mean\_worldbank = df['WorldBank\_Estimate'].mean()  mean\_un = df['UN\_Estimate'].mean()   | **Country/Territory** | **UN\_Region** | **IMF\_Estimate** | **IMF\_Year** | **WorldBank\_Estimate** | **WorldBank\_Year** | **UN\_Estimate** | **UN\_Year** | | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | Monaco | Europe | 17377.736041 | 0 | 234316.0 | 2021 | 234317.0 | 2021 | | **2** | Liechtenstein | Europe | 17377.736041 | 0 | 157755.0 | 2020 | 169260.0 | 2021 | | **3** | Luxembourg | Europe | 132372.000000 | 2023 | 133590.0 | 2021 | 133745.0 | 2021 | | **4** | Ireland | Europe | 114581.000000 | 2023 | 100172.0 | 2021 | 101109.0 | 2021 | | **5** | Bermuda | Americas | 17377.736041 | 0 | 114090.0 | 2021 | 112653.0 | 2021 |   # Print the first few rows to show the changes  print(df.head())  # Drop the temporary 'avg\_worldbank\_un' column if not needed  if 'avg\_worldbank\_un' in df.columns:      df = df.drop('avg\_worldbank\_un', axis=1)      print("'avg\_worldbank\_un' column dropped.")  else:      print("'avg\_worldbank\_un' column does not exist.")  display(df.head())  **Correlation Heatmap**  df[["IMF\_Estimate", "UN\_Estimate", "WorldBank\_Estimate"]].corr()   | **IMF\_Estimate** | **UN\_Estimate** | **WorldBank\_Estimate** | | --- | --- | --- | | **IMF\_Estimate** | 1.000000 | 0.708599 | 0.695033 | | **UN\_Estimate** | 0.708599 | 1.000000 | 0.963986 | | **WorldBank\_Estimate** | 0.695033 | 0.963986 | 1.000000 | |  |  |  |  |   corr = df[["IMF\_Estimate", "UN\_Estimate", "WorldBank\_Estimate"]].corr()  plt.figure(figsize=(9,6))  sns.heatmap(corr, annot=True)  plt.show() |

**Boxplot and Outliers**

sns.boxplot(x=df["UN\_Estimate"])

plt.show()

A graph of a bar graph

AI-generated content may be incorrect.

df[df["UN\_Estimate"]>50000].head()

| **Country/Territory** | **UN\_Region** | **IMF\_Estimate** | **IMF\_Year** | **WorldBank\_Estimate** | **WorldBank\_Year** | **UN\_Estimate** | **UN\_Year** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | Monaco | Europe | 17377.736041 | 0 | 234316.0 | 2021 | 234317.0 | 2021 |
| **2** | Liechtenstein | Europe | 17377.736041 | 0 | 157755.0 | 2020 | 169260.0 | 2021 |
| **3** | Luxembourg | Europe | 132372.000000 | 2023 | 133590.0 | 2021 | 133745.0 | 2021 |
| **4** | Ireland | Europe | 114581.000000 | 2023 | 100172.0 | 2021 | 101109.0 | 2021 |
| **5** | Bermuda | Americas | 17377.736041 | 0 | 114090.0 | 2021 | 112653.0 | 2021 |

sns.boxplot(x=df["WorldBank\_Estimate"])

plt.show()

A graph of a bar graph

AI-generated content may be incorrect.

|  |
| --- |
| **Course Notes** |

It is recommended to take notes from the course, use the space below to do so, or use the revision guide shared with the class:

|  |
| --- |
| **Additional Information** |

We have included a range of additional links to further resources and information that you may find useful, these can be found within your revision guide.

**END OF WORKBOOK**

**Please check through your work thoroughly before submitting and update the table of contents if required.**

**Please send your completed work booklet to your trainer.**