

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: |
| Course Date: |
|  |

**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 | output = []  for i in range(1, 101):  if i % 3 == 0 and i % 5 == 0:  output.append("fizzbuzz")  elif i % 3 == 0:  output.append("fizz")  elif i % 5 == 0:  output.append("buzz")  else:  output.append(str(i))  print(output) |

# **Day 3: Task 1**

Using the ‘student.csv’ which can be downloaded [here](https://justit831-my.sharepoint.com/:x:/g/personal/danpe_justit_co_uk/ER92LoQB1PpNqWj07fnfO4EBh9HB7CiI-i4RH273HoqY6A?e=mVdIeY), 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.  import pandas as pd  df = pd.read\_csv('your\_file.csv')  2.  import pandas as pd  print(df.head())  3.  import pandas as pd  print(df.info())  4.  import pandas as pd  print(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.  import pandas as pd  name\_column = df['name']  print(name\_column)  2.  import pandas as pd  name\_mark\_columns = df[['name', 'mark']]  print(name\_mark\_columns)  3.  import pandas as pd  first\_three\_rows = df.head(3) print(first\_three\_rows)  4.  import pandas as pd  class\_four\_rows = df[df['class'] == 'Four']  print(class\_four\_rows) |

### **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."

|  |
| --- |
| 1.  df['passed'] = df['mark'] >= 60  print(df)  2.  df.rename(columns={'mark': 'score'}, inplace=True)  print(df)  3.  df.drop(columns=['passed'], inplace=True)  print(df) |

### **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.  mean\_mark\_by\_class = df.groupby('class')['mark'].mean()  print(mean\_mark\_by\_class)  2.  student\_count\_by\_class = df['class'].value\_counts()  print(student\_count\_by\_class)  3.  average\_mark\_by\_gender = df.groupby('gender')['mark'].mean()  print(average\_mark\_by\_gender) |

### **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."

|  |
| --- |
| 1.  pivot\_table = df.pivot\_table(values='mark', index='class', columns='gender', aggfunc='mean')  print(pivot\_table)  2.  def assign\_grade(mark):  if mark >= 85:  return 'A'  elif 70 <= mark < 85:  return 'B'  elif 60 <= mark < 70:  return 'C'  else:  return 'D'  df['grade'] = df['mark'].apply(assign\_grade)  print(df)  3.  sorted\_df = df.sort\_values(by='mark', ascending=False)  print(sorted\_df) |

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

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

|  |
| --- |
| 1.  df.to\_csv('new\_file.csv', index=False) |

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

|  |
| --- |
| ` |

# **Day 4: Task 1**

Using the ‘GDP (nominal) per Capita.csv’ which can be downloaded [here](https://justit831-my.sharepoint.com/:x:/g/personal/danpe_justit_co_uk/EV1Xzb5eNENHmOVMDssxyoMBqTCVcLg18U4qOLUDZZHSkw?e=PAbKfN), 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

|  |
| --- |
|  |

# **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 [here](https://justit831-my.sharepoint.com/:u:/g/personal/danpe_justit_co_uk/Ede5Pb1JwPNMj49hTDzeEUMB7GZWBP7SVidCo0Gt6tnP1w?e=D83SIR). 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)

|  |
| --- |
| INPUT:  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  df = pd.read\_csv("GDP (nominal) per Capita.csv",encoding= 'unicode\_escape', index\_col=0)  print(df.info())  print(df.head())  # Number of countries per region  countries\_per\_region = df['UN\_Region'].value\_counts()  print("Number of countries per region:")  print(countries\_per\_region)  # What is "European Union [n 1]  print("Details about 'European Union [n 1]':")  print(df[df['Country/Territory'] == 'European Union [n 1]'])  # Countries in Europe below the average GDP  europe\_data = df[df['UN\_Region'] == 'Europe']  average\_gdp = europe\_data['WorldBank\_Estimate'].mean()  below\_average = europe\_data[europe\_data['WorldBank\_Estimate'] < average\_gdp]  print("Countries in Europe below the average GDP:")  print(below\_average[['Country/Territory', 'WorldBank\_Estimate']])  # Countries in Europe with a higher GDP than the UK  uk\_gdp = df.loc[df['Country/Territory'] == 'United Kingdom', 'WorldBank\_Estimate'].values[0]  higher\_than\_uk = europe\_data[europe\_data['WorldBank\_Estimate'] > uk\_gdp]  print("Countries in Europe with higher GDP than the UK:")  print(higher\_than\_uk[['Country/Territory', 'WorldBank\_Estimate']])  # Countries below average by IMF world estimate  global\_average = df['IMF\_Estimate'].mean()  below\_average\_imf = df[df['IMF\_Estimate'] < global\_average]  print("Countries below average by IMF world estimate:")  print(below\_average\_imf[['Country/Territory', 'IMF\_Estimate']])  # Highest estimates  highest\_un = df[df['UN\_Estimate'] == df['UN\_Estimate'].max()]  highest\_imf = df[df['IMF\_Estimate'] == df['IMF\_Estimate'].max()]  print("Country with highest UN Estimate:", highest\_un['Country/Territory'].values[0])  print("Country with highest IMF Estimate:", highest\_imf['Country/Territory'].values[0])  # Highest estimates  highest\_un = df[df['UN\_Estimate'] == df['UN\_Estimate'].max()]  highest\_imf = df[df['IMF\_Estimate'] == df['IMF\_Estimate'].max()]  print("Country with highest UN Estimate:", highest\_un['Country/Territory'].values[0])  print("Country with highest IMF Estimate:", highest\_imf['Country/Territory'].values[0])  # Replace 0 with NaN  df.replace(0, np.nan, inplace=True)  # Calculate average of WorldBank and UN estimates  df['avg\_worldbank\_un'] = df[['WorldBank\_Estimate', 'UN\_Estimate']].mean(axis=1)  # Fill null values in 'IMF\_Estimate' with the calculated average  df['IMF\_Estimate'].fillna(df['avg\_worldbank\_un'], inplace=True)  # Drop temporary column if not needed  df.drop(columns=['avg\_worldbank\_un'], inplace=True)  # Check missing values  print("Missing values per column:")  print(df.isnull().sum())  ## VISUALATION  ## Histogram  # General histograms  df.hist(figsize=(10, 8))  plt.show()  # Specific columns  df[['IMF\_Estimate', 'UN\_Estimate', 'WorldBank\_Estimate']].hist(figsize=(12, 9), bins=10)  plt.show()  ## Correlation Heatmap  corr = df[['IMF\_Estimate', 'UN\_Estimate', 'WorldBank\_Estimate']].corr()  plt.figure(figsize=(9, 6))  sns.heatmap(corr, annot=True, cmap='GnBu', fmt='.2f', annot\_kws={'size': 12})  plt.title("Correlation Heatmap")  plt.show()  ## Bar Plot  sns.barplot(x="UN\_Region", y="WorldBank\_Estimate", data=df, errorbar=None)  plt.xticks(rotation=90)  plt.title("World Bank Estimate by Region")  plt.show()  ## Box Plot  sns.boxplot(x=df['UN\_Estimate'])  plt.title("Boxplot of UN Estimate")  plt.show()  ## Outlier Detection  lower\_q = df['UN\_Estimate'].quantile(0.25)  upper\_q = df['UN\_Estimate'].quantile(0.75)  iqr = upper\_q - lower\_q  upper\_boundary = upper\_q + 1.5 \* iqr  lower\_boundary = lower\_q - 1.5 \* iqr  df\_filtered = df[(df['UN\_Estimate'] > lower\_boundary) & (df['UN\_Estimate'] < upper\_boundary)]  print("Filtered dataset shape:", df\_filtered.shape)  ## Scatter Plot  df.plot(x='UN\_Region', y='UN\_Estimate', kind='scatter', figsize=(10, 6), title="Scatter Plot")  plt.show()  OUTPUT:  Data columns (total 8 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Country/Territory 223 non-null object  1 UN\_Region 223 non-null object  2 IMF\_Estimate 223 non-null int64  3 IMF\_Year 223 non-null int64  4 WorldBank\_Estimate 223 non-null int64  5 WorldBank\_Year 223 non-null int64  6 UN\_Estimate 223 non-null int64  7 UN\_Year 223 non-null object  Country/Territory UN\_Region IMF\_Estimate IMF\_Year WorldBank\_Estimate \  1 Monaco Europe 0 0 234316  2 Liechtenstein Europe 0 0 157755  3 Luxembourg Europe 132372 2023 133590  4 Ireland Europe 114581 2023 100172  5 Bermuda Americas 0 0 114090  WorldBank\_Year UN\_Estimate UN\_Year  1 2021 234317 2021  2 2020 169260 2021  3 2021 133745 2021  4 2021 101109 2021  5 2021 112653 2021  Number of countries per region:  UN\_Region  Africa 55  Asia 51  Europe 48  Americas 48  Oceania 20  World 1  Countries in Europe below the average GDP:  Country/Territory WorldBank\_Estimate  34 France 43659  35 Andorra 42137  36 European Union[n 1] 38411  40 Malta 33487  41 Italy 35658  51 Slovenia 29291  52 Czech Republic 26821  53 Spain 30104  54 Estonia 27944  57 Lithuania 23723  59 Portugal 24568  60 Latvia 21148  62 Slovakia 21392  63 Greece 20193  70 Croatia 17685  72 Poland 18000  75 Hungary 18728  78 Romania 14858  87 Bulgaria 12222  90 Russia 12195  103 Montenegro 9466  106 Serbia 9230  112 Bosnia and Herzegovina 7143  115 Belarus 7302  118 North Macedonia 6695  120 Albania 6493  127 Moldova 5231  133 Kosovo 5270  143 Ukraine 4836  Countries in Europe with higher GDP than the UK:  Country/Territory WorldBank\_Estimate  1 Monaco 234316  2 Liechtenstein 157755  3 Luxembourg 133590  4 Ireland 100172  6 Norway 89154  7 Switzerland 91992  9 Isle of Man 87158  13 Iceland 68728  14 Channel Islands 75153  15 Faroe Islands 69010  16 Denmark 68008  18 Netherlands 57768  20 Austria 53638  22 Sweden 61029  23 Finland 53655  24 Belgium 51247  28 Germany 51204  Countries below average by IMF world estimate:  Country/Territory IMF\_Estimate  1 Monaco 0  2 Liechtenstein 0  5 Bermuda 0  9 Isle of Man 0  10 Cayman Islands 0  .. ... ...  219 Malawi 496  220 South Sudan 467  221 Sierra Leone 415  222 Afghanistan 611  223 Burundi 249  Country with highest UN Estimate: Monaco  Country with highest IMF Estimate: Luxembourg  Country with highest UN Estimate: Monaco  Country with highest IMF Estimate: Luxembourg  Missing values per column:  Country/Territory 0  UN\_Region 0  IMF\_Estimate 0  IMF\_Year 26  WorldBank\_Estimate 7  WorldBank\_Year 7  UN\_Estimate 9  UN\_Year 0 |

|  |
| --- |
| **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.**