

**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 | for num in range (1,101):# Declaring range with 101 because 100 included  if num %3==0 and num %5!=0: # IF the number is both divisible by 5 and 3  print("Fizz")  elif num %5==0 and num %3!=0:# If the number is divisible by only 3  print("Buzz")  elif num %5==0 and num %3==0:# If the number is only divisible by 5  print("FizzBuzz")  else:  print(num) # Printing the numbers that aren't divisible by either  Output: |

# **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."

|  |
| --- |
| import pandas as pd # Importing the pandas library  df = pd.read\_csv('student(in).csv') # Reading the student file  print(df.head())# Printing the first few records  df.info()# Information about the dataframe  df.describe()# Describing the dataframe    Output |

### **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'."

|  |
| --- |
| print(df.name)# Printing name column  print(df[['name', 'mark']])# Printing the combination of the name and the marks column  print(df.head(3)) # printing the first 3 records  print(df[df['class'] == 'Four'])# printing the records where the class is four  Output:    Had to limit the number of records to get all 4 to fit in the same screenshot |

### **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."

|  |
| --- |
| df['passed'] = df['mark'] >= 60)# Code to filter the marks  df = df.rename(columns={'mark': 'score'}))# Code to rename the column  print(df.head(3)) )# Code to print the first 3 rows  df = df.drop('passed', axis=1)# Code to drop the passed column  Output |

### **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."

|  |
| --- |
| print(df.groupby('class')['mark'].mean())# the code for the mean mark  print(df['class'].value\_counts())# The code to for the number of students in each class  print(df.groupby('gender')['mark'].mean())# The code to for the 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."

|  |
| --- |
| pivot\_table = pd.pivot\_table(df, # Creating th epivot table with class as the row gender as thec column and mark as the value                              index='class',                              columns='gender',                              values='mark')  print(pivot\_table)# Printing the pivot table  def assign\_grade(mark):# Function to assign a grade to each number      if mark >= 85:# code to assign grade A to any mark greater or equal to 85          return 'A'      elif mark >= 70:#  Code to assign grade b if any mark is greater than or equal to 70          return 'B'      elif mark >= 60: # code to assign grade c if grade hits the 60 threshold          return 'C'      else:          return 'D'# code to return d if none of the other conditions are met  df['grade'] = df['mark'].apply(assign\_grade)# Applying the grading system to the dataframe  df\_sorted = df.sort\_values('mark', ascending=False)# Sorting the dataframe  print(df\_sorted.head(5))# The top 5 records  print(df\_sorted.tail(5))# the bottom 5 records  Output: |

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

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

|  |
| --- |
| df.to\_csv('student\_grades.csv', index=False)# Code to export the csv file  Output: |

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

|  |
| --- |
|  |

# **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('capital.csv')# Renamed the file to make it easier to read  print(df.head(10))# Code for the first 10 rows  print(df.tail(5))# Code for the last 5 rows  print(df[["Country/Territory","UN\_Region"]])# Code for the columns country/territory and un\_region  Output: |

# **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)

|  |
| --- |
| Display the number of countries in each un region:  region\_counts = df["UN\_Region"].value\_counts()# Counting the amount of countries in each region  print(region\_counts)# Printing the output  Output:    Where is the European union:  df[df["Country/Territory"] == "European Union[n 1]"]# Checking the location of the european union within the dataset  Output:    Below average Europe gdp:  europe\_avg\_gdp = df[df["UN\_Region"] == "Europe"]["WorldBank\_Estimate"].mean()# The average gdp for europe  below\_avg\_europe = df[(df["UN\_Region"] == "Europe") & (df['WorldBank\_Estimate']< europe\_avg\_gdp)]# The code for for countries that are below average  europe\_df = df[df['UN\_Region'] == 'Europe']# Dataframe filtered for europe  avg\_eu\_gdp = europe\_df['WorldBank\_Estimate'].mean() # Average gdp for europe  below\_avg\_gdp = europe\_df[europe\_df['WorldBank\_Estimate'] < avg\_eu\_gdp] # The countries with below average gdp  result = below\_avg\_gdp[['Country/Territory', 'WorldBank\_Estimate']].sort\_values(by='WorldBank\_Estimate') # The result  print(result)  Output:    Which countries higher gdp than uk:  country\_uk\_gdp = df[df["Country/Territory"] == "United Kingdom"]["IMF\_Estimate"].values[0]# Code to find the gdp of the uk  print(country\_uk\_gdp)  higher\_than\_uk = df[(df["UN\_Region"] == "Europe") & (df["IMF\_Estimate"] > country\_uk\_gdp)]# Code to find countries with a higher gdp than the uk  print(higher\_than\_uk  Output:    Group by:  group\_unyear = df.groupby("UN\_Year")["Country/Territory"].count()# Grouping by un year  print(group\_unyear)  group\_countries = df.groupby("UN\_Region")["Country/Territory"].count()# Grouping by un region  print(group\_countries)    Output:    Below imf world:  df\_nonnull = df[df['IMF\_Estimate'] > 0] # Removing the null values from the imf estimate column  global\_average\_imf = df\_nonnull['IMF\_Estimate'].mean() # Calculating the average imf estimate  below\_average\_imf = df\_nonnull[df\_nonnull['IMF\_Estimate'] < global\_average\_imf]# Seperating the countries with a imf estimate below the average  result = below\_average\_imf[['Country/Territory', 'IMF\_Estimate', 'UN\_Region']].sort\_values(by='IMF\_Estimate') # Sorting the countries with a imf estimate below the average  print(result)  Output:    Imf estimate 0 values  df\_nonnull = df[df['IMF\_Estimate'] < 1]# The countries with an estimate smaller than one  print(df\_nonnull)  Output:    Highest un estimate:  max\_df= df['UN\_Estimate'].max()# Finding the max un estimate  result = df[df['UN\_Estimate'] == max\_df][['Country/Territory', 'UN\_Estimate']]# Finding the country the max estimate belongs too  print(result)# Printing both  Output:    Highest imf estimate  max\_df= df['IMF\_Estimate'].max()# Finding the max imf estimate  result = df[df['IMF\_Estimate'] == max\_df][['Country/Territory', 'IMF\_Estimate']]# Finding the country the max estimate belongs too  print(result)# Printing both  Output:    Missing values:  null\_columns = ['IMF\_Estimate', 'WorldBank\_Estimate', 'UN\_Estimate', 'IMF\_Year', 'WorldBank\_Year', 'UN\_Year']# The colummns with null values  df[null\_columns] = df[null\_columns].replace(0, pd.NA)# Replacing the null values  print(null\_columns)  print(df.head())# Printing the first few rows    Output:    Had problems doing the visualisations on the jupyter for some reason. |

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