

**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 i in range(1,101):    if i % 5 == 0 and i % 3 == 0:      print(f"{i} - fizzbuzz")    elif i % 3 == 0:      print(f"{i} - fizz")    elif i % 5 == 0:      print (f"{i} - buzz")    else:      print(i) |

# **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  # 1. Read the file into a table called df  df = pd.read\_csv('student.csv')  # 2. Peek at the first 5 rows to check your data  print("----------2. The first 5 rows--------")  print(df.head())  # 3. Print info (counts & data types)  print("\n----------3. Info (counts & data types)--------")  df.info()  # 4. Print summary statistics  print("\n----------4. Summary statistics:--------")  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'."

|  |
| --- |
| import pandas as pd  # Read the file into a table called df  df = pd.read\_csv('student.csv')  # 1. Select the 'name' column  print("----------1. Select the 'name' column--------")  col\_name = df['name'].head(10)  print("\n Name \n", col\_name)  # 2. Select the 'name' and 'mark' columns.  print("\n----------2. Select the 'name' and 'mark' columns.--------")  col\_name\_and\_mark = df[['name', 'mark']].head(10)  print(col\_name\_and\_mark)  # 3. Select the first 3 rows.  print("\n----------3. First 3 rows --------")  print(df.head(3))  #4. The 'class' is 'Four'.  print("\n----------4. The 'class' is 'Four' --------")  class\_four = df[df['class'] == 'Four']  print(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."

|  |
| --- |
| import pandas as pd  # Read the file into a table called df  df = pd.read\_csv('student.csv')  # 1.  Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."  df\_student\_greater\_60 = df.copy()  df\_student\_greater\_60['passed'] = df\_student\_greater\_60['mark'] >= 60  print("\n New column 'passed' that indicates whether the student passed \n", df\_student\_greater\_60.head(10))  # 2.  Question: "Write the code to rename the 'mark' column to 'score'."  print("\nColumns before renaming:", df\_student\_greater\_60.columns)  df\_renamed = df\_student\_greater\_60.copy()  df\_renamed.rename(columns={'mark': 'score'}, inplace=True)  print("Columns after renaming:", df\_renamed.columns)  # 3.  Question: "Write the code to drop the 'passed' column."  df\_dropped = df\_renamed.copy()  df\_dropped = df\_dropped.drop(columns=['passed'])  print("Columns after dropping:", df\_dropped.columns) |

### **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."

|  |
| --- |
| import pandas as pd  # Read the file into a table called df  df = pd.read\_csv('student.csv')  # 1.  Group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."  print("\nAverage mark per class:")  print(df.groupby('class')['mark'].mean())  # 2.  Count the number of students in each class."  print("\nNumber of students per class:")  print(df['class'].value\_counts())  # 3.  Average mark for each gender.  print("\nAverage mark per gender:")  print(df.groupby('gender')['mark'].mean()) |

### **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."

|  |
| --- |
| import pandas as pd  # Read the file into a table called df  df = pd.read\_csv('student.csv')  # 1.  A pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values.  print("Pivot Table:")  pivot = df.pivot\_table(index='class', columns='gender', values='mark')  print(pivot)  # 2.  New column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."  df\_grade = df.copy()  df\_grade['grade'] = df\_grade['mark'].apply(lambda x: 'A' if x >= 85 else ('B' if 70 <= x < 85 else ('C' if 60 <= x < 70 else 'D')))  print("\nNew DataFrame with 'grade' column:")  print(df\_grade)  # 3.  Question: "Write the code to sort the DataFrame by 'mark' in descending order."  df\_sorted = df\_grade.sort\_values(by='mark', ascending=False)  print("\nDataFrame sorted by 'mark' in descending order:")  print(df\_sorted) |

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

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

|  |
| --- |
|  |

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

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  # 1. Read the file into a table called df  df = pd.read\_csv('student.csv')  # -------------------------------  # Step 1: View the original shape and check for missing values  # -------------------------------  print("Original data shape (rows, columns):")  print(df.shape)  print("\nMissing values in each column:")  print(df.isnull().sum())  # -------------------------------  # Step 2: Remove rows with missing data  # -------------------------------  df\_no\_missing = df.copy()  df\_no\_missing = df\_no\_missing.dropna()  print("\nData shape after dropping rows with missing values:")  print(df\_no\_missing.shape)  print("Missing values after dropping:")  print(df\_no\_missing.isnull().sum())  # -------------------------------  # Step 3: Fill missing values  # -------------------------------  df\_filled = df.copy()  # Convert 'mark' to numeric, turn errors into NaN  df\_filled['mark'] = pd.to\_numeric(df\_filled['mark'], errors='coerce')  # Fill missing marks with the average (mean) value  mean\_mark = df\_filled['mark'].mean()  df\_filled['mark'] = df\_filled['mark'].fillna(mean\_mark)  print("\nMissing values Mark after filling:")  # Fill missing gender values with most\_common\_gender  most\_common\_gender = df\_filled['gender'].mode()[0]  df\_filled['gender'] = df\_filled['gender'].fillna(most\_common\_gender)  # Fill missing Name values with 'noName'  df\_filled['name'] = df\_filled['name'].fillna('noName')  # Fill missing Class values with 'noClass'  df\_filled = df\_filled.dropna(subset=['class'])  # -------------------------------  # Step 4: Remove duplicate rows  # -------------------------------  df\_no\_duplicates = df\_filled.copy()  # Drop duplicate rows  df\_no\_duplicates = df\_no\_duplicates.drop\_duplicates()  print("\nShape after removing duplicate rows:")  print(df\_no\_duplicates.shape)  print(df\_no\_duplicates.head())  # -------------------------------  # Step 5: Visualizing the results  # -------------------------------  # How many students are there in each 'class'?  plt.figure()  df\_no\_duplicates['class'].value\_counts().plot(kind='bar', title='Students per Class', color='skyblue')  plt.show()  # Line plot of student marks  plt.figure()  pd.to\_numeric(df\_no\_duplicates['mark'], errors='coerce').plot(title='Histogram of Student Marks', color='purple')  plt.show()  # The numbers of 'male' and 'female' students  plt.figure()  df\_no\_duplicates['gender'].value\_counts().plot(kind='bar', title='Gender Comparison', color='#FFFCA8',)  plt.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  # Read and save the ‘GDP (nominal) per Capita’ data to a data frame called “df” in Jyputer notebook  df = pd.read\_csv('GDP (nominal) per Capita.csv')  print(df.info())  print("=============================")  print("The first 10 rows")  print("=============================")  print(df.head(10))  print("\n=============================")  print("The last 5 rows:")  print("=============================")  print(df.tail(5))  print("\n=============================")  print("‘Country/Territory’ and ‘UN\_Region’ columns:")  print("=============================")  print(df[['Country/Territory', 'UN\_Region']]) |

# **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)

|  |
| --- |
|  |

|  |
| --- |
| **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:

name = input("What is your name? ")

glasses\_of\_water = int(input("How many glasses of water you drank today? "))

exercise\_hours = int(input("How many hours you exercised today? "))

healthy\_score = (glasses\_of\_water \* 2) + (exercise\_hours \* 5)

print(f"\nHello, {name}! Hope you are well! ")

print(f"Your healthy score is: {healthy\_score}.")

print("Good job!")

| **Concept** | **What it does** | | **Example Use** |
| --- | --- | --- | --- |
| **Loop** | Repeats code | Go through numbers 1–10 | |
| **Function** | Groups code into **reusable** blocks | Create a "greet" tool | |

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

<https://www.youtube.com/watch?v=iGFdh6_FePU>

### 🔥 Problem with Single Backslash

In strings, a backslash is used to **escape** characters like:

* \n = newline
* \t = tab
* \\ = literal backslash
* \" = double quote
* \' = single quote

#### ❌ Example of a mistake:

python

КопіюватиРедагувати

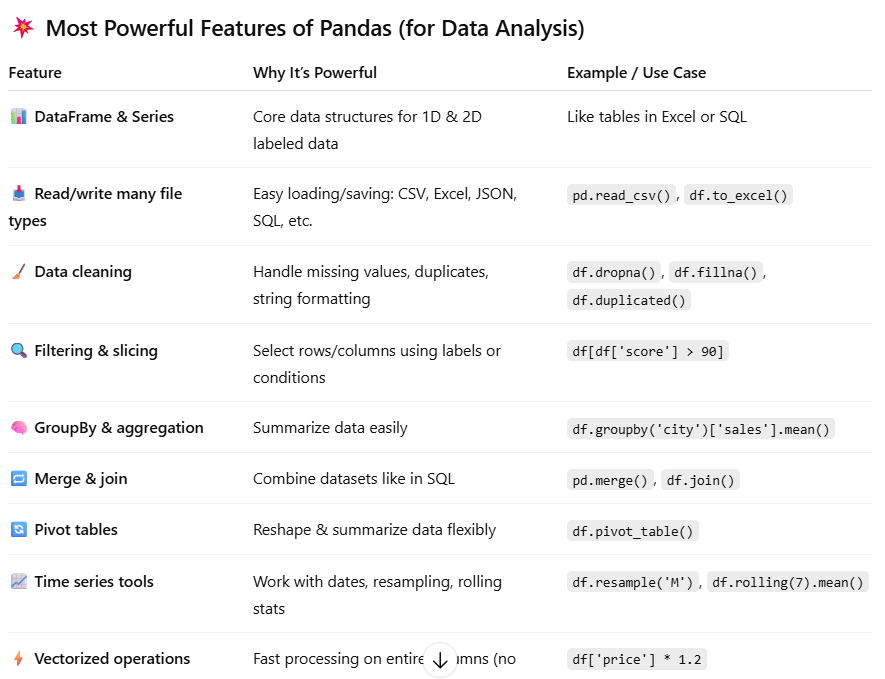
path = "C:\newfolder\test"

print(path)

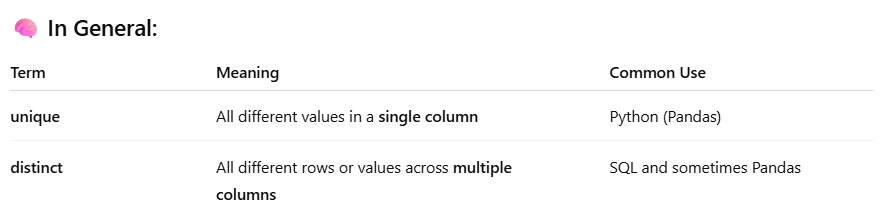
🔎 **Problem:**

* \n becomes a **newline**
* \t becomes a **tab**

<https://pandas.pydata.org/>



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.





<https://www.youtube.com/watch?v=-E7nMqPVmyQ>

<https://data.gov/>

**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.**