



Just IT

 B2Wgroup

Apprenticeships | Training | Recruitment

# Data Technician

Name: Sara Haider

Course Date: 22/05/2025

## Table of contents

Day 2: Task 1 .....	3
<b>Day 3: Task 1 .....</b>	<b>4</b>
<b>Exercise 1: Loading and Exploring the Data.....</b>	<b>4</b>
<b>Exercise 2: Indexing and Slicing .....</b>	<b>6</b>
<b>Exercise 3: Data Manipulation .....</b>	<b>8</b>
<b>Exercise 4: Aggregation and Grouping.....</b>	<b>9</b>
<b>Exercise 5: Advanced Operations .....</b>	<b>11</b>
<b>Exercise 6: Exporting Data .....</b>	<b>12</b>
<b>Exercise 7: If finished early try visualising the results.....</b>	<b>12</b>
<b>Day 4: Task 1 .....</b>	<b>13</b>
<b>4: Task 2 .....</b>	<b>15</b>
<b>Course Notes.....</b>	<b>19</b>
<b>Additional Information.....</b>	<b>19</b>



## 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

```
[2] # Go through the integers from 1 to 100.
    for x in range(1, 101):

        # If a number is both divisible by 3 and by 5, print "fizzbuzz"
        if x % 3 == 0 and x % 5 == 0:
            print("fizzbuzz")

        # If a number is divisible by 3, print "fizz"
        elif x % 3 == 0:
            print("fizz")

        # If a number is divisible by 5, print "buzz"
        elif x % 5 == 0:
            print("buzz")

        # Otherwise, print just the number
        else:
            print(x)
```

```
⇒ 1
   2
   fizz
   4
   buzz
   fizz
   7
   8
   fizz
   buzz
  11
  fizz
  13
  14
 fizzbuzz
```



## 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. Import by clicking files>upload>click on student file


```
import pandas as pd

df = pd.read_csv('student.csv')
print(df)
```

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
6	7	My John Rob	Fifth	78	male

- 2.

```
print (df.head(5))
```



	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

3.

```
print(df.info())
```

```
>>> <class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0    id      35 non-null    int64
 1   name    34 non-null    object
 2   class   34 non-null    object
 3   mark    35 non-null    int64
 4   gender  33 non-null    object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
None
```

4.

```
print(df.describe())
```

	id	mark
count	35.000000	35.000000
mean	18.000000	74.657143
std	10.246951	16.401117
min	1.000000	18.000000
25%	9.500000	62.500000
50%	18.000000	79.000000
75%	26.500000	88.000000
max	35.000000	96.000000

## 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.

```
df['name']
```



	name
0	John Deo
1	Max Ruin
2	Arnold
3	Krish Star
4	John Mike
5	Alex John
6	My John Rob
7	Asruid

2.

```
df[['name', 'mark']]
```



	name	mark
0	John Deo	75
1	Max Ruin	85
2	Arnold	55
3	Krish Star	60
4	John Mike	60
5	Alex John	55
6	My John Rob	78
7	Asruid	85
8	Tes Qry	78
9	Big John	55



3.

```
df.iloc[:3]
```

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male

4.

```
df[df['class'] == 'Four']
```



	id	name	class	mark	gender
0	1	John Deo	Four	75	female
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
9	10	Big John	Four	55	female
15	16	Gimmy	Four	88	male
20	21	Babby John	Four	69	female
30	31	Marry Toeey	Four	88	male

### 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)
```

	id	name	class	mark	gender	passed
0	1	John Deo	Four	75	female	True
1	2	Max Ruin	Three	85	male	True
2	3	Arnold	Three	55	male	False
3	4	Krish Star	Four	60	female	True
4	5	John Mike	Four	60	female	True
5	6	Alex John	Four	55	male	False
6	7	My John Rob	Fifth	78	male	True
7	8	Asruid	Five	85	male	True
8	9	Tes Qry	Six	78	NaN	True
9	10	Big John	Four	55	female	False
10	11	Ronald	Six	89	female	True
11	12	Recky	Six	94	female	True
12	13	Kty	Seven	88	female	True
13	14	Bigy	Seven	88	female	True

2.

```
df.rename(columns={'mark': 'score'}, inplace=True)  
print(df)
```

	id	name	class	score	gender	passed
0	1	John Deo	Four	75	female	True
1	2	Max Ruin	Three	85	male	True
2	3	Arnold	Three	55	male	False
3	4	Krish Star	Four	60	female	True
4	5	John Mike	Four	60	female	True
5	6	Alex John	Four	55	male	False
6	7	My John Rob	Fifth	78	male	True
7	8	Asruid	Five	85	male	True
8	9	Tes Qry	Six	78	NaN	True
9	10	Big John	Four	55	female	False
10	11	Ronald	Six	89	female	True
11	12	Recky	Six	94	female	True
12	13	Kty	Seven	88	female	True
13	14	Bigy	Seven	88	female	True



3.

```
df.drop('passed', axis=1, inplace=True)
print(df)
```

```

id      name  class  score  gender
0      1  John Deo   Four    75  female
1      2  Max Ruin   Three    85    male
2      3    Arnold   Three    55    male
3      4  Krish Star   Four    60  female
4      5  John Mike   Four    60  female
5      6  Alex John   Four    55    male
6      7  My John Rob Fifth    78    male
7      8    Asruid   Five    85    male
8      9    Tes Qry   Six    78     NaN
9     10  Big John   Four    55  female
10    11    Ronald   Six    89  female

```

## 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.

```
df.groupby('class')['mark'].mean()
```

```

class
Eight  79.000000
Fifth  78.000000
Five   80.000000
Four   68.750000
Nine   41.500000
Seven  77.600000
Six    82.571429
Three  73.666667

dtype: float64

```

2.

```
df['class'].value_counts()
```



count	
class	
Seven	10
Four	8
Six	7
Three	3
Nine	2
Five	2
Fifth	1
Eight	1

3.

```
df.groupby('gender')['mark'].mean()
```

mark	
gender	
female	77.312500
male	71.588235

## 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  $\geq 85$  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 = df.pivot_table(values='mark', index='class', columns='gender', aggfunc='mean')
print(pivot)
```

gender	female	male
class		
Eight	NaN	79.0
Fifth	NaN	78.0
Five	NaN	80.0
Four	63.8	77.0
Nine	65.0	18.0
Seven	81.4	73.8
Six	89.2	54.0
Three	NaN	70.0

2.

```
df['grade'] = pd.cut(
    df['mark'],
    bins=[0, 59, 69, 84, 100],
    labels=['D', 'C', 'B', 'A']
)
print(df[['name', 'mark', 'grade']])
```

	name	mark	grade
0	John Deo	75	B
1	Max Ruin	85	A
2	Arnold	55	D
3	Krish Star	60	C
4	John Mike	60	C
5	Alex John	55	D
6	My John Rob	78	B
7	Asruid	85	A
8	Tes Qry	78	B
9	Big John	55	D
10	Ronald	89	A
11	Recky	94	A
12	Kty	88	A
13	Bigy	88	A
14	Tade Row	88	A
15	Gimmy	88	A

3.

```
df.sort_values(by='mark', ascending=False, inplace=True)
print(df[['name', 'mark']])
```

	name	mark
32	Kenn Rein	96
11	Recky	94
31	Binn Rott	90
10	Ronald	89
30	Marry Toeey	88
34	Rows Noup	88
24	Giff Tow	88
14	Tade Row	88
15	Gimmy	88
12	Kty	88
13	Bigy	88
27	Rojj Base	86
7	Asruid	85
1	Max Ruin	85
26	NaN	81

## 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('students_with_grades.csv', index=False)
print(df)
```

	id	name	class	mark	gender
32	33	Kenn Rein	Six	96	female
11	12	Recky	Six	94	female
31	32	Binn Rott	Seven	90	female
10	11	Ronald	Six	89	female
30	31	Marry Toeey	Four	88	male
34	35	Rows Noump	Six	88	female
		...			

..

▶ drive

▶ sample\_data

student.csv

students\_with\_grades.csv

## 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 Jupyter 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')

print(df.head(10))
```

	Unnamed: 0	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	\
0	1	Monaco	Europe	0	0	
1	2	Liechtenstein	Europe	0	0	
2	3	Luxembourg	Europe	132372	2023	
3	4	Ireland	Europe	114581	2023	
4	5	Bermuda	Americas	0	0	
5	6	Norway	Europe	101103	2023	
6	7	Switzerland	Europe	98767	2023	
7	8	Singapore	Asia	91100	2023	
8	9	Isle of Man	Europe	0	0	
9	10	Cayman Islands	Americas	0	0	

	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
0	234316	2021	234317	2021
1	157755	2020	169260	2021
2	133590	2021	133745	2021
3	100172	2021	101109	2021
4	114090	2021	112653	2021
5	89154	2021	89242	2021
6	91992	2021	93525	2021
7	72794	2021	66822	2021
8	87158	2019	0	0
9	86569	2021	85250	2021

```
print(df.tail(5))
```

	Unnamed: 0	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	\
218	219	Malawi	Africa	496	2023	
219	220	South Sudan	Africa	467	2023	
220	221	Sierra Leone	Africa	415	2023	
221	222	Afghanistan	Asia	611	2020	
222	223	Burundi	Africa	249	2023	

	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
218	635	2021	613	2021
219	1072	2015	400	2021
220	480	2021	505	2021
221	369	2021	373	2021
222	222	2021	311	2021

```
print(df[['Country/Territory', 'UN_Region']])
```

	Country/Territory	UN_Region
0	Monaco	Europe
1	Liechtenstein	Europe
2	Luxembourg	Europe
3	Ireland	Europe
4	Bermuda	Americas
..	...	...
218	Malawi	Africa
219	South Sudan	Africa
220	Sierra Leone	Africa
221	Afghanistan	Asia
222	Burundi	Africa

[223 rows x 2 columns]



## 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.](#)

### # Number of countries per region

```
region = df['UN_Region'].value_counts()
print(region)
```

```
UN_Region
Africa      55
Asia        51
Americas    48
Europe      48
Oceania     20
World        1
Name: count, dtype: int64
```

### #What is European Union [n 1]?

```
eu = df[df["Country/Territory"] == "European Union[n 1]"]
print(eu)
```

```
Unnamed: 0   Country/Territory  UN_Region  IMF_Estimate  IMF_Year  \
35          36  European Union[n 1]    Europe         39940        2023

WorldBank_Estimate  WorldBank_Year  UN_Estimate  UN_Year
35                38411            2021         31875    2021
```

### # Countries in Europe below average

```
europe_ba = df[(df["UN_Region"] == "Europe") & (df["WorldBank_Estimate"] < df["WorldBank_Estimate"].mean())]
print(europe_ba[["Country/Territory", "WorldBank_Estimate"]])
```

```
Country/Territory  WorldBank_Estimate
69      Croatia         17685
71      Poland         18000
74      Hungary         18728
77      Romania         14858
86      Bulgaria        12222
89      Russia         12195
102     Montenegro         9466
105     Serbia          9230
111  Bosnia and Herzegovina        7143
114     Belarus          7302
117   North Macedonia         6695
119     Albania         6493
126     Moldova         5231
132     Kosovo         5270
142     Ukraine         4836
```



## #Which countries in Europe has higher GDP than UK?

```
# Get UK's GDP
gdp = df.loc[df["Country/Territory"] == "United Kingdom", "WorldBank_Estimate"].values[0]

# Find European countries with higher GDP
higher_gdp= df[(df["UN_Region"] == "Europe") & (df["WorldBank_Estimate"] > gdp)][["Country/Territory", "WorldBank_Estimate"]]

print(higher_gdp)
```

	Country/Territory	WorldBank_Estimate
0	Monaco	234316
1	Liechtenstein	157755
2	Luxembourg	133590
3	Ireland	100172
5	Norway	89154
6	Switzerland	91992
8	Isle of Man	87158
12	Iceland	68728
13	Channel Islands	75153
14	Faroe Islands	69010
15	Denmark	68008
17	Netherlands	57768
19	Austria	53638
21	Sweden	61029
22	Finland	53655
23	Belgium	51247
27	Germany	51204

## #What is the average GDP for each region?

```
# Group by region and calculate average GDP
avg_gdp_reg = df.groupby("UN_Region")["WorldBank_Estimate"].mean()

print(avg_gdp_reg)
```

```
UN_Region
Africa      2470.836364
Americas    18565.125000
Asia        13921.313725
Europe      45193.687500
Oceania     15113.650000
World       12235.000000
Name: WorldBank_Estimate, dtype: float64
```

## #Which countries below average by IMF world estimate?

```
# Calculate the average world IMF
avg_imf = df["IMF_Estimate"].mean()

# Filter and display countries below the average
print(df[df["IMF_Estimate"] < avg_imf][["Country/Territory", "IMF_Estimate"]])
```

	Country/Territory	IMF_Estimate
0	Monaco	0
1	Liechtenstein	0
4	Bermuda	0
8	Isle of Man	0
9	Cayman Islands	0
...	...	...
218	Malawi	496
219	South Sudan	467
220	Sierra Leone	415
221	Afghanistan	611
222	Burundi	249

[159 rows x 2 columns]



### #IMF estimate 0 values

```
# Countries where IMF estimate is 0
print(df[df["IMF_Estimate"] == 0][["Country/Territory"]])
```

```
Country/Territory
0      Monaco
1    Liechtenstein
4      Bermuda
8      Isle of Man
9    Cayman Islands
13   Channel Islands
14   Faroe Islands
18   Greenland
30  British Virgin Islands
36   US Virgin Islands
38   New Caledonia
41      Guam
57  Sint Maarten (Dutch part)
60  Northern Mariana Islands
64  Saint Martin (French part)
67  Turks and Caicos Islands
70   French Polynesia
75     Cook Islands
76     Anguilla
81     Curaçao
84     Montserrat
85   American Samoa
103     Cuba
195     Zanzibar
203     Syria
211   North Korea
```

### #Which country has highest UN Estimate?

```
print(df.loc[df["UN_Estimate"].idxmax(), ["Country/Territory", "UN_Estimate"]])
```

```
Country/Territory    Monaco
UN_Estimate          234317
Name: 0, dtype: object
```

### #Which country has highest Worldbank Estimate?

```
print(df.loc[df["WorldBank_Estimate"].idxmax(), ["Country/Territory", "WorldBank_Estimate"]])
```

```
Country/Territory    Monaco
WorldBank_Estimate    234316
Name: 0, dtype: object
```

### #Which country has highest IMF Estimate?

```
print(df.loc[df["IMF_Estimate"].idxmax(), ["Country/Territory", "IMF_Estimate"]])
```

```
Country/Territory    Luxembourg
IMF_Estimate          132372
Name: 2, dtype: object
```

## #Filling 0 Values by average

```
import numpy as np
```

```
# replace 0 with null values
```

```
df.replace(0, np.nan, inplace=True)
print(df.head())
```

```
Unnamed: 0  Country/Territory  UN_Region  IMF_Estimate  IMF_Year  \
0          1          Monaco      Europe           NaN          NaN
1          2  Liechtenstein      Europe           NaN          NaN
2          3      Luxembourg      Europe      132372.0      2023.0
3          4          Ireland      Europe      114581.0      2023.0
4          5          Bermuda  Americas           NaN          NaN
```

```
WorldBank_Estimate  WorldBank_Year  UN_Estimate  UN_Year
0          234316.0          2021.0      234317.0      2021
1          157755.0          2020.0      169260.0      2021
2          133590.0          2021.0      133745.0      2021
3          100172.0          2021.0      101109.0      2021
4          114090.0          2021.0      112653.0      2021
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
```

```
df["Average_Estimate"] = df[["WorldBank_Estimate", "UN_Estimate"]].mean(axis=1)

print(df[["Country/Territory", "Average_Estimate"]])
```

```
Country/Territory  Average_Estimate
0          Monaco      234316.5
1  Liechtenstein      163507.5
2    Luxembourg      133667.5
3      Ireland      100640.5
4      Bermuda      113371.5
..          ...          ...
218        Malawi        624.0
219   South Sudan        736.0
220   Sierra Leone        492.5
221   Afghanistan        371.0
222     Burundi        266.5
```

```
[223 rows x 2 columns]
```

```
# Fill null values with the average
```

```
df["IMF_Estimate"].fillna(df["IMF_Estimate"].mean(), inplace=True)

print(df[["Country/Territory", "IMF_Estimate"]].head())
```

```
Country/Territory  IMF_Estimate
0          Monaco      17377.736041
1  Liechtenstein      17377.736041
2    Luxembourg      132372.000000
3      Ireland      114581.000000
4      Bermuda      17377.736041
```

## #Missing Values

```
missing = df.isnull().sum()
print(missing)
```

```
Unnamed: 0      0
Country/Territory  0
UN_Region        0
IMF_Estimate     0
IMF_Year        26
WorldBank_Estimate  7
WorldBank_Year    7
UN_Estimate      9
UN_Year          0
Average_Estimate  1
dtype: int64
```

## 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:

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.**

