

Fundamental Libraries in Python

Introduction:

In this assignment, we focus on key operations in Python using the libraries NumPy and Pandas. These exercises aim to improve your data manipulation and analysis skills by working with arrays and data structures. The tasks cover various essential actions, such as reshaping arrays, slicing data, modifying series and dataframes, and saving data. Each exercise will help you gain practical experience to build efficient data pipelines and perform exploratory data analysis.

1. **NumPy:** This is a Python library used for numerical computations. It provides support for creating and manipulating arrays, enabling efficient mathematical operations on large datasets.
2. **Pandas:** This is a Python library used for data manipulation and analysis. It provides data structures like Series and DataFrame, which simplify handling and analysing structured data.
3. **Dataframe:** It is a two-dimensional, size-mutable, and heterogeneous data structure with labelled axes (rows and columns). It is like a spreadsheet or SQL table.
4. **CSV (Comma-Separated Values):** It is a plain text file that stores tabular data in a structured format, where each line represents a row and each value is separated by a comma.
5. **Reshaping:** It is the process of changing the structure or dimensions of a NumPy array, such as converting a 1D array into a 2D matrix.
6. **Slicing:** It is the process of selecting specific portions of an array or DataFrame based on index or conditions. It helps in extracting subsets of data for analysis.

These concepts are fundamental in data analysis and manipulation, forming the basis for working with structured and semi-structured data in Python.

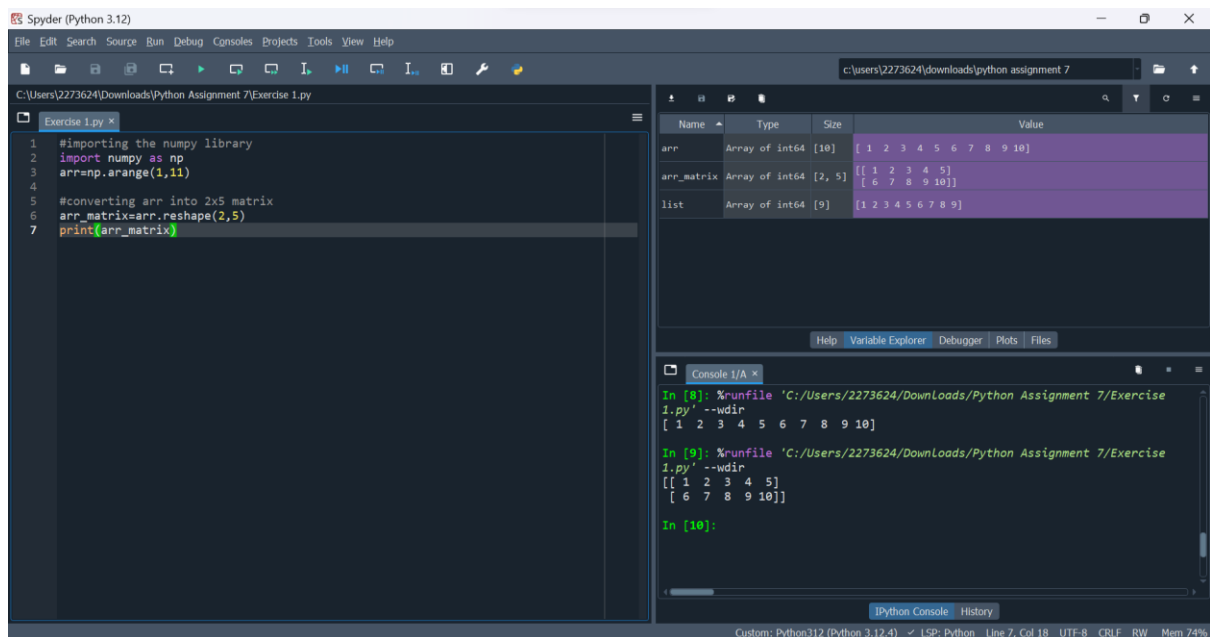
Exercise 1: (Score: 1)

Create a numpy array containing the numbers from 1 to 10, and then reshape it to a 2x5 matrix.

- Description: In this exercise, we created a NumPy array from 1 to 10 and reshaped it into a 2x5 matrix. This helps understand the basic structure and reshaping operations in NumPy.
- This exercise improves the understanding of array creation and reshaping, essential when handling multi-dimensional data for processing.
- Code:

```
#importing the numpy library
import numpy as np
arr=np.arange(1,11)
```

```
#converting arr into 2x5 matrix
arr_matrix=arr.reshape(2,5)
print(arr_matrix)
```



The screenshot shows the Spyder Python IDE interface. The left pane displays the code for Exercise 1. The right pane shows the variable explorer with the following data:

Name	Type	Size	Value
arr	Array of int64	[10]	[1 2 3 4 5 6 7 8 9 10]
arr_matrix	Array of int64	[2, 5]	[[1 2 3 4 5] [6 7 8 9 10]]
list	Array of int64	[9]	[1 2 3 4 5 6 7 8 9]

The bottom pane shows the IPython console output:

```
In [8]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/Exercise 1.py' --wdir
[1 2 3 4 5 6 7 8 9 10]

In [9]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/Exercise 1.py' --wdir
[[1 2 3 4 5]
 [6 7 8 9 10]]

In [10]:
```

Exercise 2: (Score: 1)

Create a numpy array containing the numbers from 1 to 20, and then extract the elements between the 5th and 15th index.

- Description: In this task, we created a NumPy array with values from 1 to 20 and extracted elements between the 5th and 15th index.
- Advantage: Slicing arrays is crucial for selecting subsets of data, which is commonly needed in data exploration and manipulation tasks.
- Code:

```
#importing the numpy library
import numpy as np
```

```
arr=np.arange(1,21)
```

```
#slicing the array from 5th to 15th index
slice_arr=arr[5:16]
print(slice_arr)
```

```
Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\2273624\Downloads\Python Assignment 7\Exercise 2.py

Exercise 1.py * Exercise 2.py *
1 #importing the numpy library
2 import numpy as np
3
4 arr=np.arange(1,21)
5
6 #slicing the array from 5th to 15th index
7 slice_arr=arr[5:16]
8 print(slice_arr)

Name Type Size Value
arr Array of int64 [20] [ 1  2  3 ... 18 19 20]
arr_matrix Array of int64 [2, 5] [[ 1  2  3  4  5]
[ 6  7  8  9 10]]
list Array of int64 [9] [ 1  2  3  4  5  6  7  8  9]
slice_arr Array of int64 [11] [ 6  7  8 ... 14 15 16]

Help Variable Explorer Debugger Plots Files

Console 1/A *
In [13]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/
untitled4.py' --wdir
[ 6  7  8  9 10 11 12 13 14 15 16]

In [14]:

Python Console History
Custom: Python312 (Python 3.12.4) LSP: Python Line 8, Col 17 UTF-8 CRLF RW Mem 70%
```

Exercise 3: (Score: 2)

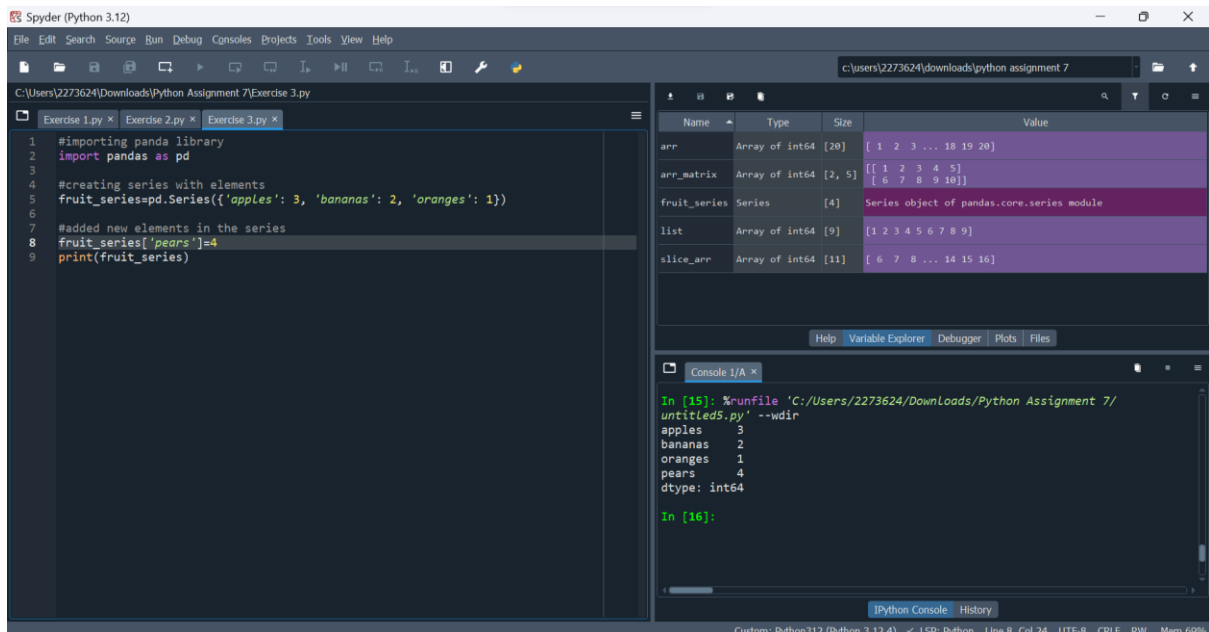
Create a Pandas series with the following data: {'apples': 3, 'bananas': 2, 'oranges': 1}. Then, add a new item to the series with the key 'pears' and the value 4.

- Description: Created a Pandas series with some fruit data and added a new item, demonstrating the ability to modify series dynamically.
- Advantage: This exercise teaches how to handle one-dimensional data and how to modify it, useful for managing simple datasets.
- Code:

```
#importing panda library
import pandas as pd

#creating series with elements
fruit_series=pd.Series({'apples': 3, 'bananas': 2, 'oranges': 1})

#added new elements in the series
fruit_series['pears']=4
print(fruit_series)
```



The screenshot shows the Spyder Python IDE interface. The left pane displays the code for Exercise 3, which imports pandas, creates a Series with fruit data, adds a new item 'pears', and prints the result. The right pane shows the Variable Explorer with a table of variables and their values. The bottom pane shows the IPython console with the execution output.

Name	Type	Size	Value
arr	Array of int64	[20]	[1 2 3 ... 18 19 20]
arr_matrix	Array of int64	[2, 5]	[[1 2 3 4 5] [6 7 8 9 10]]
fruit_series	Series	[4]	Series object of pandas.core.series module
list	Array of int64	[9]	[1 2 3 4 5 6 7 8 9]
slice_arr	Array of int64	[11]	[6 7 8 ... 14 15 16]

```
In [15]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/
untitled5.py' --wdir
apples      3
bananas     2
oranges     1
pears       4
dtype: int64

In [16]:
```

Exercise 4: (Score: 2)

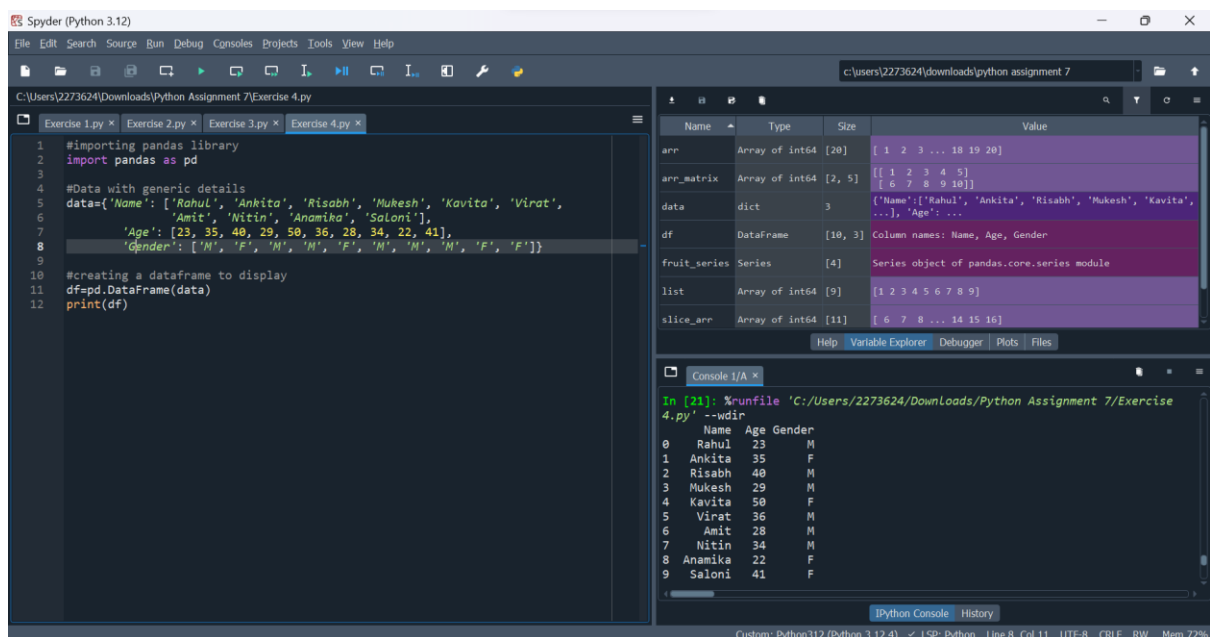
Create a dataframe with the following columns: name, age, and gender. The dataframe should have 10 rows of data.

- Description: Created a dataframe with columns name, age, and gender, adding 10 rows of data. This shows how to structure and organize tabular data in Pandas.
- Advantage: Creating a dataframe is key for managing and analysing tabular data, forming the basis of data operations in real-world applications.
- Code:

```
#importing pandas library
import pandas as pd
```

```
#Data with generic details
data= {'Name': ['Rahul', 'Ankita', 'Risabh', 'Mukesh', 'Kavita', 'Virat',
               'Amit', 'Nitin', 'Anamika', 'Saloni'],
       'Age': [23, 35, 40, 29, 50, 36, 28, 34, 22, 41],
       'Gender': ['M', 'F', 'M', 'M', 'F', 'M', 'M', 'M', 'F', 'F']}
```

```
#creating a dataframe to display
df=pd.DataFrame(data)
print(df)
```



The screenshot shows the Spyder Python IDE interface. The left pane displays the code for Exercise 4, which imports pandas, creates a dictionary 'data' with 10 rows of data, and creates a DataFrame 'df' from it. The right pane shows the Variable Explorer with a table of variables and their values. The console at the bottom shows the output of the code execution, displaying the DataFrame structure and the printed data.

Name	Age	Gender
Rahul	23	M
Ankita	35	F
Risabh	40	M
Mukesh	29	M
Kavita	50	F
Virat	36	M
Amit	28	M
Nitin	34	M
Anamika	22	F
Saloni	41	F

Exercise 5: (Score: 1)

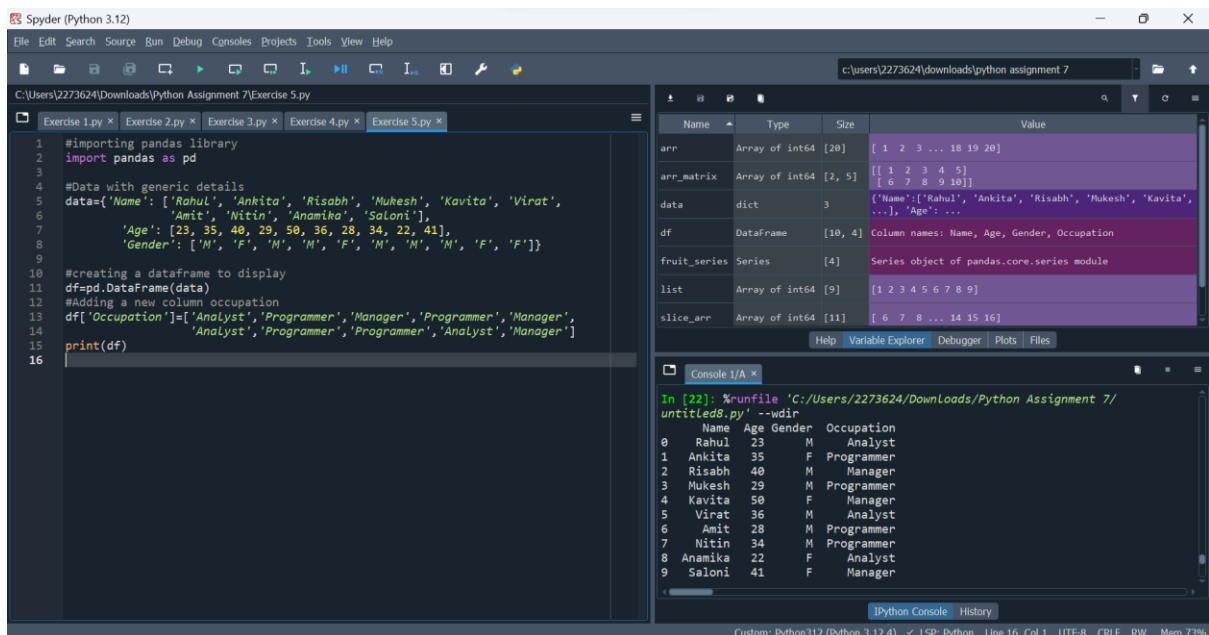
Add a new column to the data frame created in question 4, called occupation. The values for this column should be Programmer, Manager, and Analyst, corresponding to the rows in the dataframe.

- Description: Added a new column, occupation, to the dataframe. This helps in modifying existing dataframes to enrich data.
- Advantage: Adding columns is essential for augmenting datasets with new attributes, enabling more comprehensive analysis.
- Code:

```
#importing pandas library
import pandas as pd

#Data with generic details
data={'Name': ['Rahul', 'Ankita', 'Risabh', 'Mukesh', 'Kavita', 'Virat',
              'Amit', 'Nitin', 'Anamika', 'Saloni'],
      'Age': [23, 35, 40, 29, 50, 36, 28, 34, 22, 41],
      'Gender': ['M', 'F', 'M', 'M', 'F', 'M', 'M', 'M', 'F', 'F']}

#creating a dataframe to display
df=pd.DataFrame(data)
#Adding a new column occupation
df['Occupation']=['Analyst','Programmer','Manager','Programmer','Manager',
                 'Analyst','Programmer','Programmer','Analyst','Manager']
print(df)
```



The screenshot displays the Spyder Python IDE interface. The left pane shows the code for Exercise 5, which imports pandas, creates a DataFrame from a dictionary, adds a new 'Occupation' column, and prints the result. The right pane shows the Variable Explorer with a table of the DataFrame's contents. The bottom console shows the printed output of the DataFrame.

	Name	Age	Gender	Occupation
0	Rahul	23	M	Analyst
1	Ankita	35	F	Programmer
2	Risabh	40	M	Manager
3	Mukesh	29	M	Programmer
4	Kavita	50	F	Manager
5	Virat	36	M	Analyst
6	Amit	28	M	Programmer
7	Nitin	34	M	Programmer
8	Anamika	22	F	Analyst
9	Saloni	41	F	Manager

Exercise 6: (Score: 1)

Select the rows of the dataframe where the age is greater than or equal to 30.

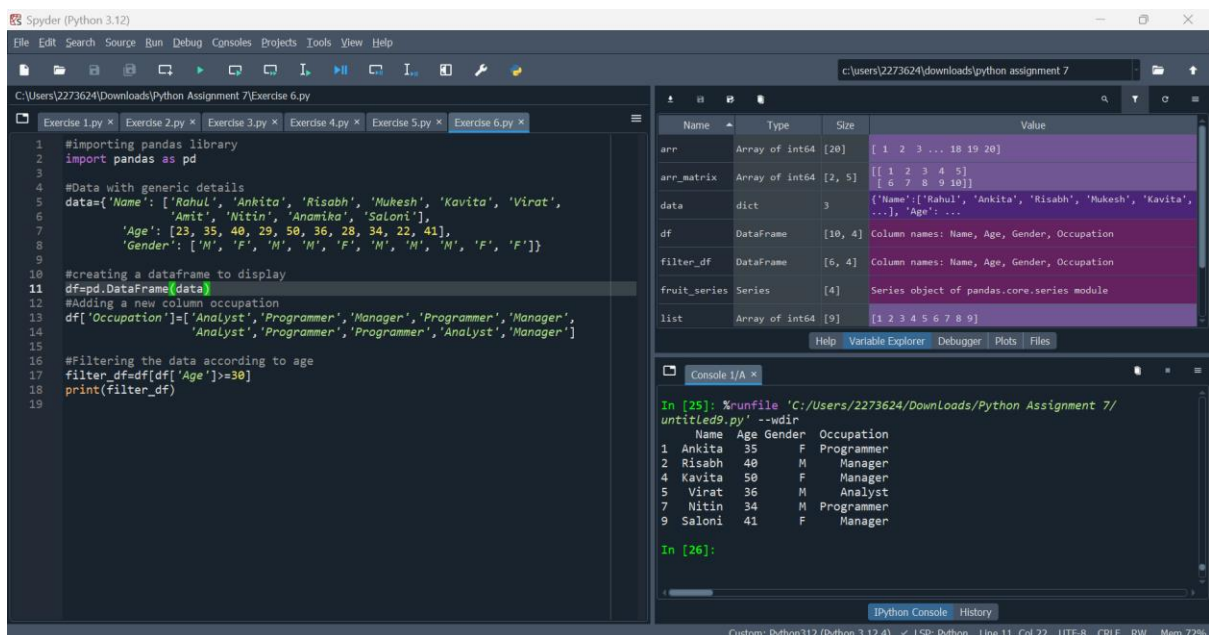
- Description: Filtered rows where age is greater than or equal to 30 from the dataframe. The data selection is based on conditions.
- Advantage: Filtering allows for targeted data extraction, which is vital for analysing specific groups within a larger dataset.
- Code:

```
#importing pandas library
import pandas as pd

#Data with generic details
data={'Name': ['Rahul', 'Ankita', 'Risabh', 'Mukesh', 'Kavita', 'Virat',
               'Amit', 'Nitin', 'Anamika', 'Saloni'],
      'Age': [23, 35, 40, 29, 50, 36, 28, 34, 22, 41],
      'Gender': ['M', 'F', 'M', 'M', 'F', 'M', 'M', 'M', 'F', 'F']}

#creating a dataframe to display
df=pd.DataFrame(data)
#Adding a new column occupation
df['Occupation'] = ['Analyst','Programmer','Manager','Programmer','Manager',
                   'Analyst','Programmer','Programmer','Analyst','Manager']

#Filtering the data according to age
filter_df=df[df['Age']>=30]
print(filter_df)
```



The screenshot shows the Spyder Python IDE interface. The editor window displays the code for Exercise 6. The variable explorer on the right shows the variables created: 'arr' (Array of int64 [20]), 'arr_matrix' (Array of int64 [2, 5]), 'data' (dict), 'df' (DataFrame with 10 rows and 4 columns), 'filter_df' (DataFrame with 6 rows and 4 columns), 'fruit_series' (Series), and 'list' (Array of int64 [9]). The console window at the bottom shows the output of the code, displaying the filtered DataFrame with 6 rows and 4 columns.

```
In [25]: %runfile 'C:/Users/2273624/Downloads/Python Assignment 7/untitled9.py' --wdir
Name Age Gender Occupation
1 Ankita 35 F Programmer
2 Risabh 40 M Manager
3 Kavita 50 F Manager
4 Virat 36 M Analyst
5 Nitin 34 M Programmer
6 Saloni 41 F Manager

In [26]:
```

Exercise 7: (Score: 2)

Convert this dataframe to a csv file and read that csv file, finally display the contents.

- Description: In this task, I converted the DataFrame to a CSV file, read it back, and displayed its contents. This shows how to save and reload data for persistence.
- Advantage: Saving and loading data is crucial for data sharing, storage, and reuse, especially in collaborative and production environments.
- Code:

```
#importing pandas library
import pandas as pd

#Data with generic details
data={'Name': ['Rahul', 'Ankita', 'Risabh', 'Mukesh', 'Kavita', 'Virat',
              'Amit', 'Nitin', 'Anamika', 'Saloni'],
      'Age': [23, 35, 40, 29, 50, 36, 28, 34, 22, 41],
      'Gender': ['M', 'F', 'M', 'M', 'F', 'M', 'M', 'M', 'F', 'F']}

#creating a dataframe to display
df=pd.DataFrame(data)
#Adding a new column occupation
df['Occupation']=['Analyst','Programmer','Manager','Programmer','Manager',
                 'Analyst','Programmer','Programmer','Analyst','Manager']

#convert the dataframe into a csv file
df.to_csv('employee.csv',index=False)
#reading the csv file again
read_df=pd.read_csv('employee.csv')
print(read_df)
```

The screenshot shows the Spyder Python IDE interface. The editor on the left contains the Python code for Exercise 7. The variable explorer on the right shows the variables created: 'data' (dict), 'df' (DataFrame), and 'read_df' (DataFrame). The console at the bottom displays the output of the code, showing the DataFrame with columns Name, Age, Gender, and Occupation.

```
1 #importing pandas library
2 import pandas as pd
3
4 #Data with generic details
5 data={'Name': ['Rahul', 'Ankita', 'Risabh', 'Mukesh', 'Kavita', 'Virat',
6               'Amit', 'Nitin', 'Anamika', 'Saloni'],
7       'Age': [23, 35, 40, 29, 50, 36, 28, 34, 22, 41],
8       'Gender': ['M', 'F', 'M', 'M', 'F', 'M', 'M', 'M', 'F', 'F']}
9
10 #creating a dataframe to display
11 df=pd.DataFrame(data)
12 #Adding a new column occupation
13 df['Occupation']=['Analyst','Programmer','Manager','Programmer','Manager',
14                 'Analyst','Programmer','Programmer','Analyst','Manager']
15
16 #convert the dataframe into a csv file
17 df.to_csv('employee.csv',index=False)
18 #reading the csv file again
19 read_df=pd.read_csv('employee.csv')
20 print(read_df)
```

	Name	Age	Gender	Occupation
0	Rahul	23	M	Analyst
1	Ankita	35	F	Programmer
2	Risabh	40	M	Manager
3	Mukesh	29	M	Programmer
4	Kavita	50	F	Manager
5	Virat	36	M	Analyst
6	Amit	28	M	Programmer
7	Nitin	34	M	Programmer
8	Anamika	22	F	Analyst
9	Saloni	41	F	Manager