



Module Title

Fundamentals of Data Science

Assessment Weightage & Type

Weekly Assignment 7 and 8 - Coursework & Regular

Year

2025

Student Name: NIRVIK K.C.

UWE ID: 25024649

Assignment Due Date: July 7, 2025

Assignment Submission Date: July 7, 2025



Bi-weekly assignment

Module Details

Module Code	UFCFK1-15-0
Module Title	Fundamentals of Data Science
Module Tutors	Saurav Gautam
Year	2024-2025
Component/Element Number	PSA/Bi-weekly assignment/Regular
Weighting	10%

Dates

Submission Date	07-July-2025
Submission Place	Backboard
Submission Time	23:59
Submission Notes	Submit Gitlab URL

Assignment 1

This assignment consists of the programming questions related to the topics of week 3 and week 4. The main topics of questions are: Python Basics, Operators, and Conditional Statements.

All the students are required to follow the format of the program as specified in the guideline below.

1. All the programs should have initial **doc string** comment (‘’ description of program’’) mentioning what your program will do.
2. Try to maintain single/multi-line comments in the place where needed to make the program understandable.
3. Maintain proper indention and newline spaces to increase the readability of the program.
4. The deliverable are 2 type of files (a single word file and multiple python program files):
 - a) Separate python program files with **.py** extension (e.g. program_name.py). Provide a relevant name to your program file on the basis of functionality of the program.
 - b) A word file describing the working of all the programs according to their number. The details required in this is the description of program, screenshot of the testing (input given and output obtained in the execution environment such as IDLE or Command prompt or terminal whichever you prefer.). It is preferred that you work with multiple inputs and outputs.

Questions

1. Write a program to generate a numpy array of numbers (e.g. [1, 2, 3, 4, 5]).

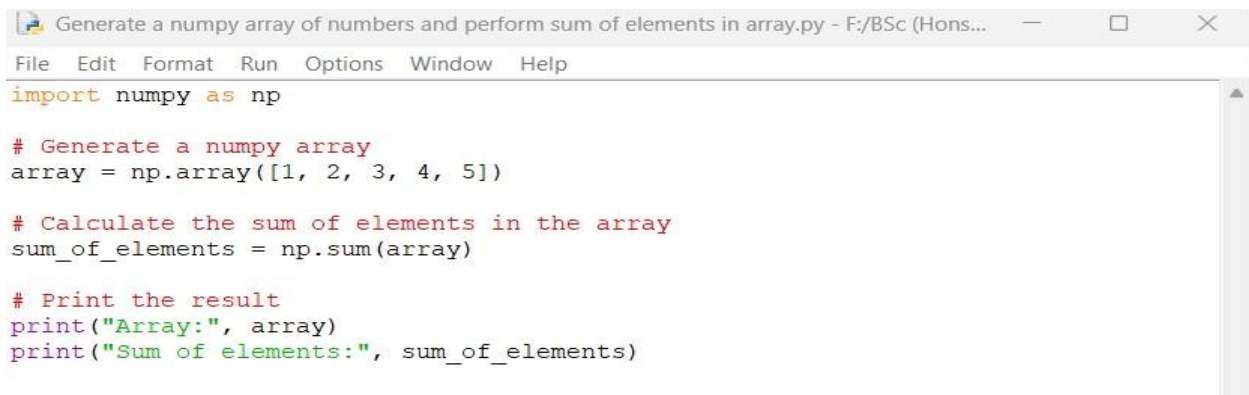
Perform the numpy array operations on it such as:

a) Sum of elements in array

Answer:

The given python program below uses the Numpy library to create an array and perform operation like finding the sum of elements in array.

Following code for input:



```

Generate a numpy array of numbers and perform sum of elements in array.py - F:/BSc (Hons...
File Edit Format Run Options Window Help
import numpy as np

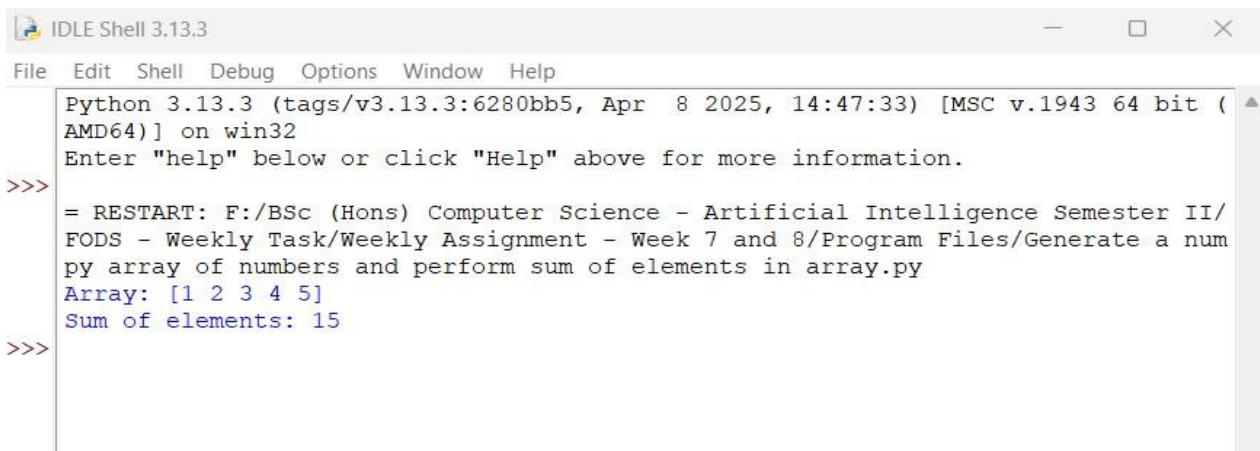
# Generate a numpy array
array = np.array([1, 2, 3, 4, 5])

# Calculate the sum of elements in the array
sum_of_elements = np.sum(array)

# Print the result
print("Array:", array)
print("Sum of elements:", sum_of_elements)

```

Output obtained in execution:



```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Generate a num
py array of numbers and perform sum of elements in array.py
Array: [1 2 3 4 5]
Sum of elements: 15
>>>

```

Python Program File: “Generate a numpy array of numbers and perform sum of elements in array.py.”

Explanation of code:

Importing Numpy:

This line imports the Numpy library and allows you to use it with the alias ‘np’. NumPy is a powerful library for numerical computations in python.

Creating a Numpy Array:

Here, a NumPy array named ‘array’ is created using the ‘np.array()’ function. The array contains the integers 1, 2, 3, 4, and 5.

Calculating the Sum of Elements:

```
sum_of_elements = np.sum(array)
```

The ‘np.sum()’ function is called with the ‘array’ as an argument. This function calculates the sum of all the elements in the array. In this case, it adds $1 + 2 + 3 + 4 + 5$, resulting in 15 as sum.

Printing the Results:

The ‘print()’ function is used to display the contents of the array and the sum is calculated. The first print statement outputs the array, and the second print statement outputs the sum of its elements.

Conclusion of the Program:

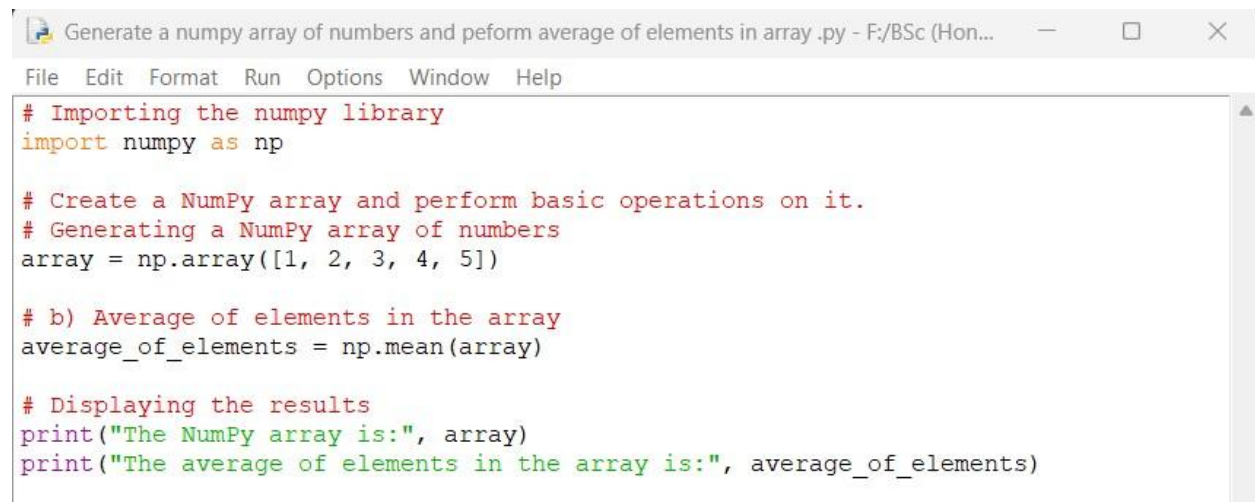
The given python program is an example of how to create a NumPy array and perform basic operation (sum) on its elements. The output shows the original array and the result of the sum. This shows an example of using NumPy for numerical computations.

b) Average of elements in array

Answer:

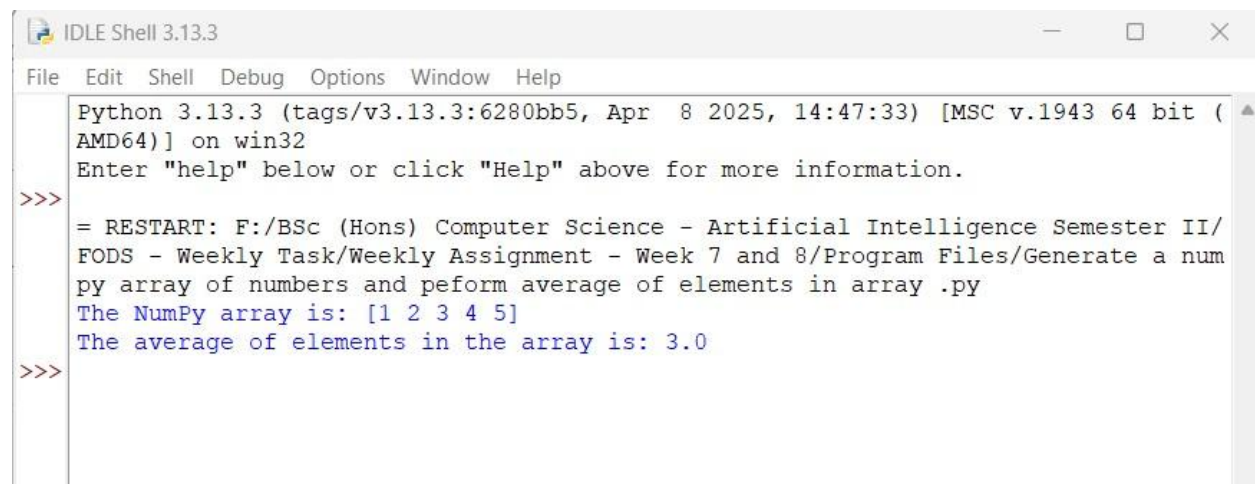
The given python program below generates a Numpy array of numbers and calculates the average of its elements.

Following code for input:



```
Generate a numpy array of numbers and perform average of elements in array .py - F:/BSc (Hon...  
File Edit Format Run Options Window Help  
# Importing the numpy library  
import numpy as np  
  
# Create a NumPy array and perform basic operations on it.  
# Generating a NumPy array of numbers  
array = np.array([1, 2, 3, 4, 5])  
  
# b) Average of elements in the array  
average_of_elements = np.mean(array)  
  
# Displaying the results  
print("The NumPy array is:", array)  
print("The average of elements in the array is:", average_of_elements)
```

Output obtained in execution:



```
IDLE Shell 3.13.3  
File Edit Shell Debug Options Window Help  
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32  
Enter "help" below or click "Help" above for more information.  
>>>  
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/  
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Generate a num  
py array of numbers and perform average of elements in array .py  
The NumPy array is: [1 2 3 4 5]  
The average of elements in the array is: 3.0  
>>>
```

Python Program File: “Generate a numpy array of numbers and perform average of elements in array.py.”

Explanation of code:

Importing Numpy:

The program begins by importing the Numpy library, which is essential for creating and manipulating arrays.

Creating a NumPy Array:

The ‘np.array()’ function is used to create a NumPy array containing the numbers 1 through 5.

Calculating the Average:

The ‘np.mean()’ function calculates the average (mean) of all elements in the array.

Displaying the Results:

The program prints the original array and the average of all elements in array.

Conclusion of the Program:

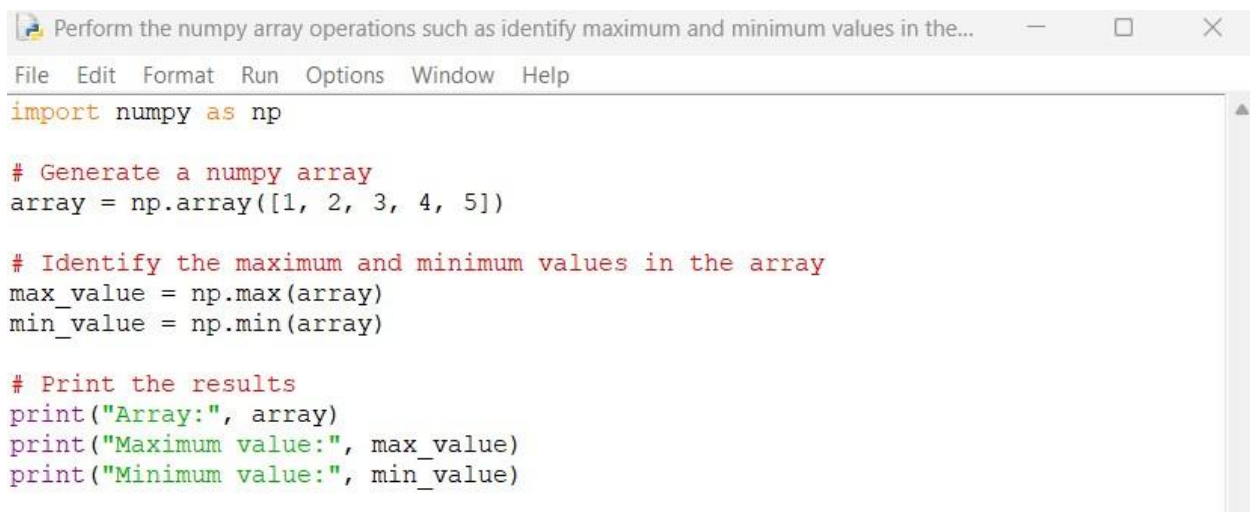
The given python program is an example of how to create a NumPy array and perform a basic operation – calculating the average of its elements. The ‘np.mean()’ function provides a way to calculate the average of all elements in array, showcasing NumPy’s capabilities for efficient numerical calculations.

c) Identify maximum and minimum values in the array

Answer:

The given python program below identifies the maximum and minimum values in a NumPy array. It generates a NumPy array of numbers and finds both the maximum and minimum values.

Following code for input:



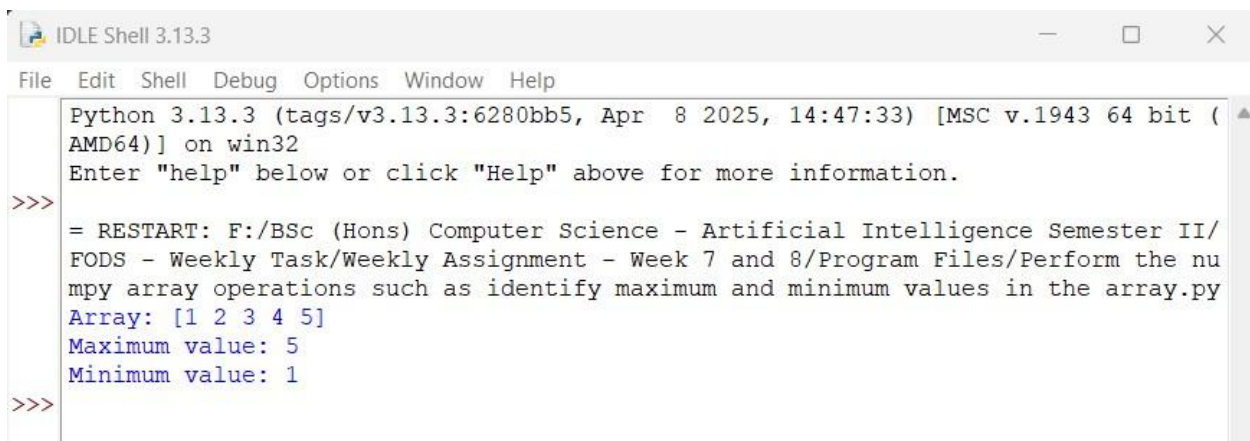
```
import numpy as np

# Generate a numpy array
array = np.array([1, 2, 3, 4, 5])

# Identify the maximum and minimum values in the array
max_value = np.max(array)
min_value = np.min(array)

# Print the results
print("Array:", array)
print("Maximum value:", max_value)
print("Minimum value:", min_value)
```

Output obtained in execution:



```
IDLE Shell 3.13.3

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Perform the numpy array operations such as identify maximum and minimum values in the array.py
Array: [1 2 3 4 5]
Maximum value: 5
Minimum value: 1
>>>
```


Python Program File: “Perform the numpy array operations such as identify maximum and minimum values in the array.py.”

Explanation of code:

Import NumPy:

The program starts by importing the NumPy library.

Create an Array:

A NumPy array is created with the numbers 1 through 5.

Find Maximum Value:

The ‘np.max()’ function is used to find the maximum value in the given array.

Find the Minimum Value:

The ‘np.min()’ function is used to find the minimum value in the array.

Output of the Program:

Finally, the array, its maximum value, and minimum value are printed to the console.

Conclusion of the Program:

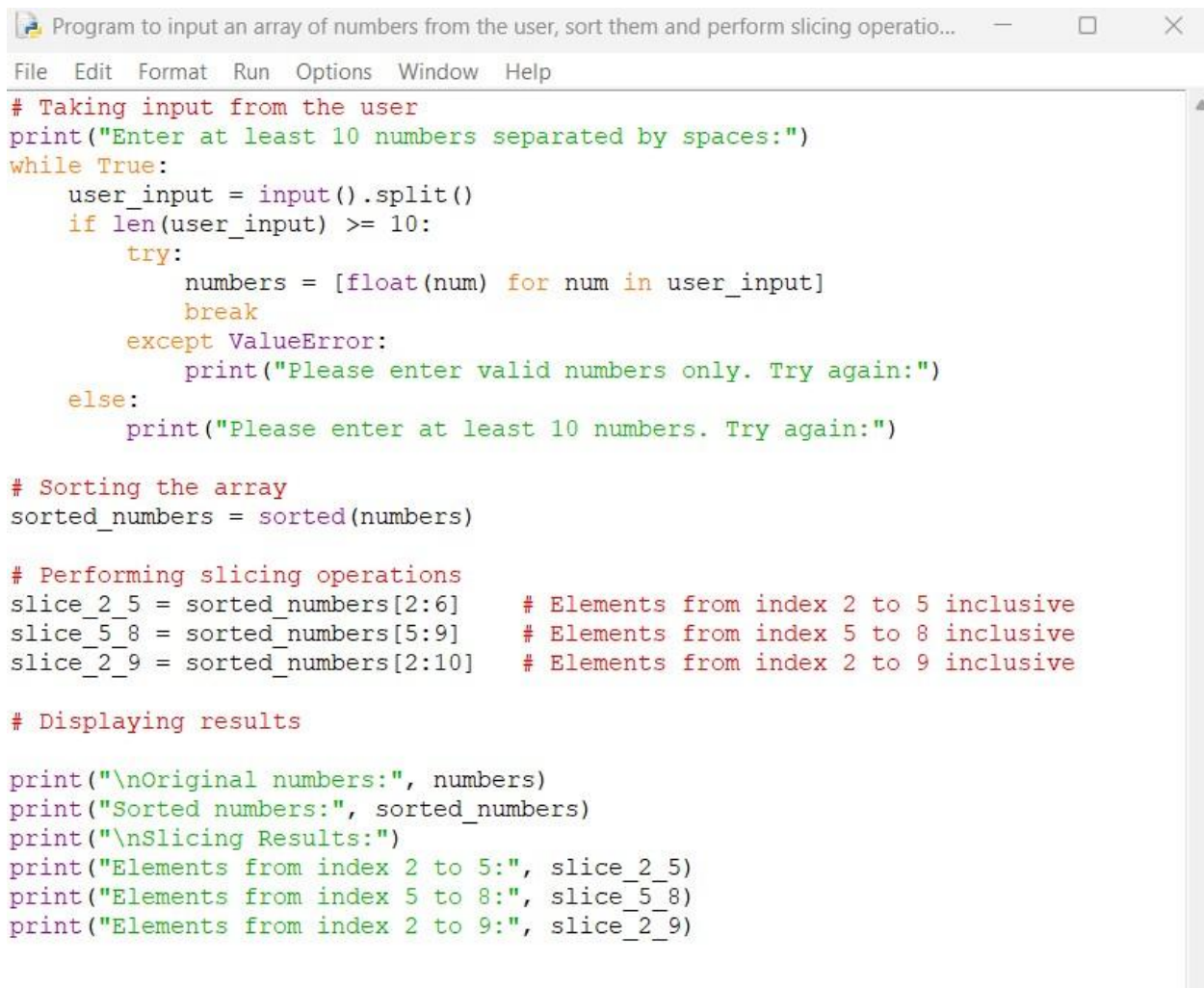
The given program demonstrates how to create a NumPy array and perform operations to find the maximum and minimum values within that given array. The output shows the original array along with the resulted maximum and minimum values.

2. Write a program to input an array of numbers from the user (at least 10 elements in list), sort them and perform slicing operations to get elements between indexes such as 2-5, 5-8, 2-9.

Answer:

The given python program below takes a list of numbers from the user (minimum 10 elements), sort them, and performs slicing operations.

Following code for input:



```

Program to input an array of numbers from the user, sort them and perform slicing operatio...
File Edit Format Run Options Window Help
# Taking input from the user
print("Enter at least 10 numbers separated by spaces:")
while True:
    user_input = input().split()
    if len(user_input) >= 10:
        try:
            numbers = [float(num) for num in user_input]
            break
        except ValueError:
            print("Please enter valid numbers only. Try again:")
    else:
        print("Please enter at least 10 numbers. Try again:")

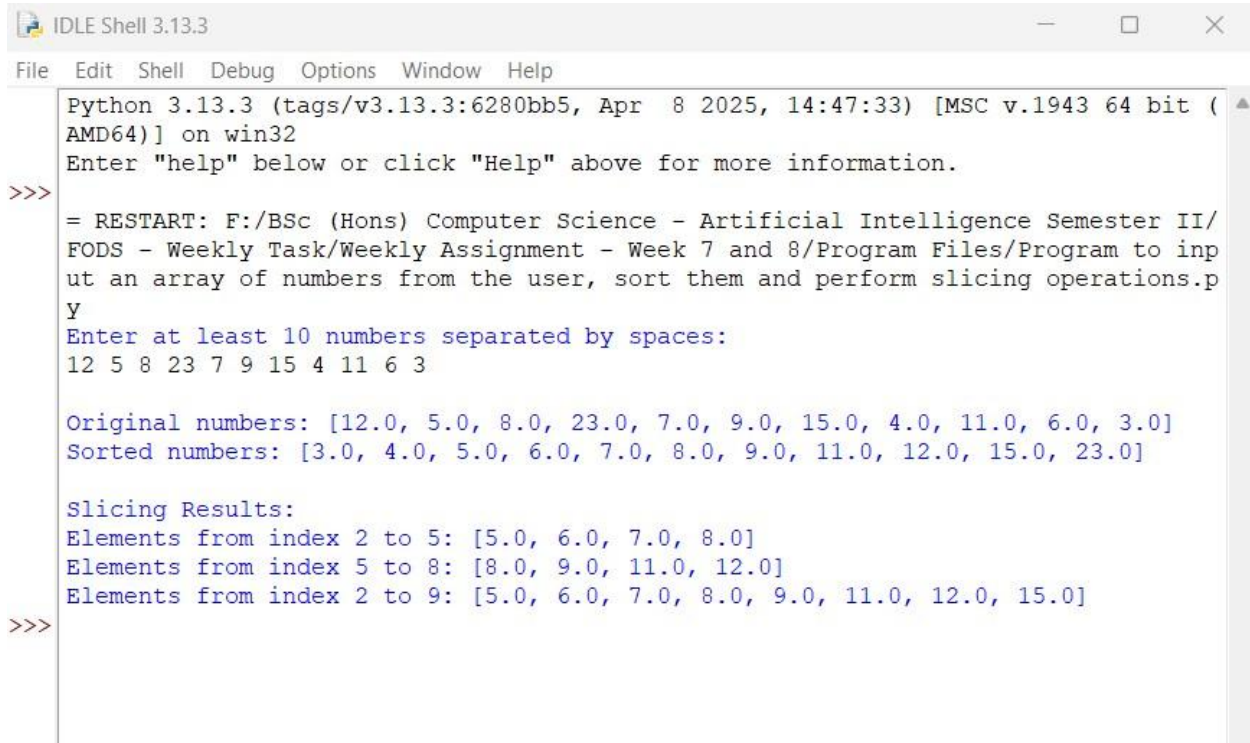
# Sorting the array
sorted_numbers = sorted(numbers)

# Performing slicing operations
slice_2_5 = sorted_numbers[2:6]      # Elements from index 2 to 5 inclusive
slice_5_8 = sorted_numbers[5:9]      # Elements from index 5 to 8 inclusive
slice_2_9 = sorted_numbers[2:10]     # Elements from index 2 to 9 inclusive

# Displaying results
print("\nOriginal numbers:", numbers)
print("Sorted numbers:", sorted_numbers)
print("\nSlicing Results:")
print("Elements from index 2 to 5:", slice_2_5)
print("Elements from index 5 to 8:", slice_5_8)
print("Elements from index 2 to 9:", slice_2_9)

```

Output obtained in execution:



```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Program to input an array of numbers from the user, sort them and perform slicing operations.py
Enter at least 10 numbers separated by spaces:
12 5 8 23 7 9 15 4 11 6 3

Original numbers: [12.0, 5.0, 8.0, 23.0, 7.0, 9.0, 15.0, 4.0, 11.0, 6.0, 3.0]
Sorted numbers: [3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 11.0, 12.0, 15.0, 23.0]

Slicing Results:
Elements from index 2 to 5: [5.0, 6.0, 7.0, 8.0]
Elements from index 5 to 8: [8.0, 9.0, 11.0, 12.0]
Elements from index 2 to 9: [5.0, 6.0, 7.0, 8.0, 9.0, 11.0, 12.0, 15.0]
>>>

```

Python Program File: “Program to input an array numbers from the user, sort them and perform slicing operations.py.”

Explanation of code:

Input Handling:

The program prompts the user to enter at least 10 numbers separated by spaces. It validates the input for the user to ensure at least 10 numbers are entered. It can convert the input strings to float numbers.

Sorting:

The entered numbers are sorted in ascending order using the built-in function ‘sorted()’.

Slicing:

Slicing uses 'start:stop' notation where 'start' is inclusive and 'stop' is exclusive. For index 2-5, we '[2:6]' to get elements at positions 2, 3, 4, 5. Similarly we use slicing for other ranges.

Output of the Program:

It shows the original input, sorted list, and the three requested slices.

Conclusion of the Program:

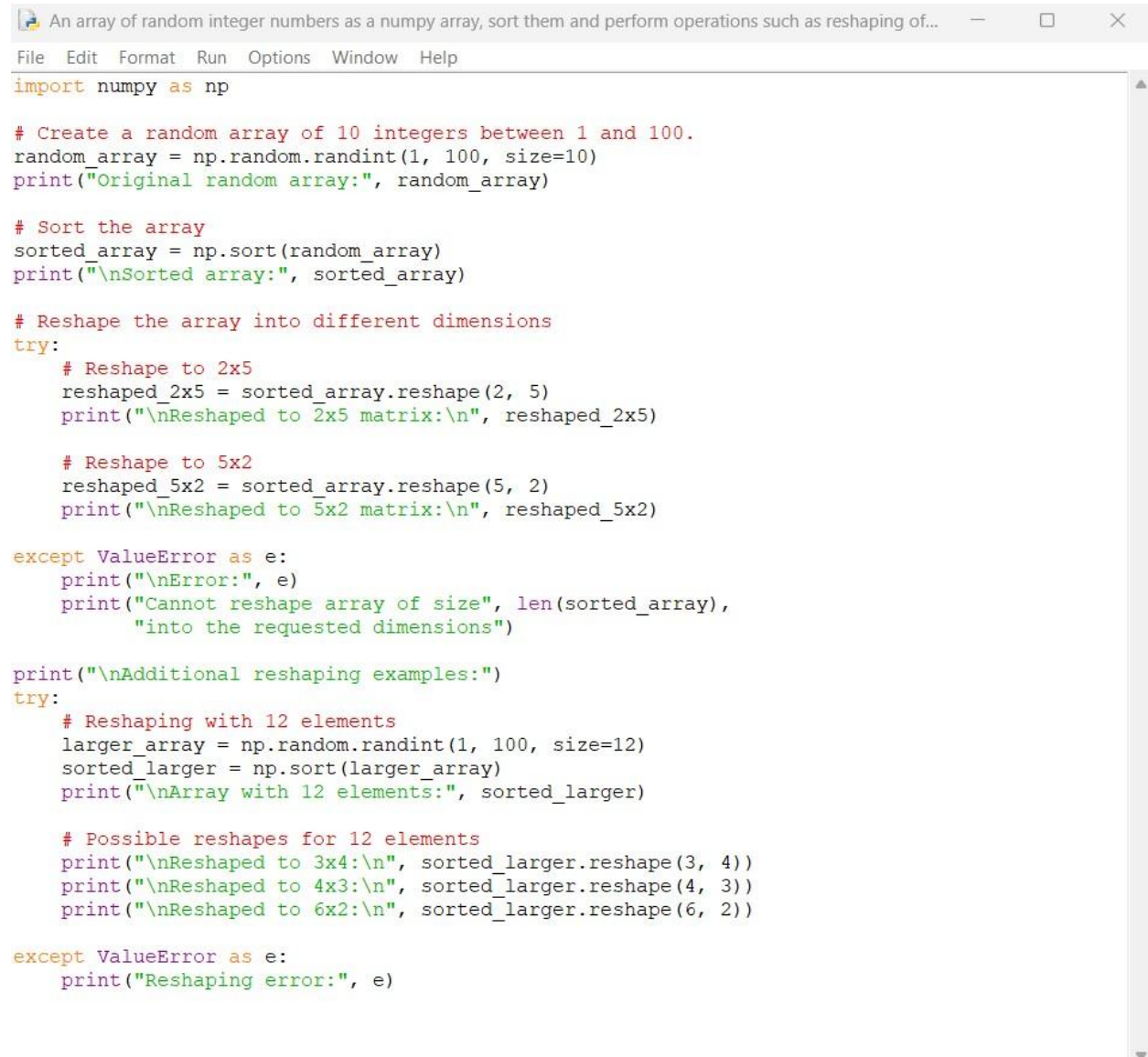
The given python program is an example of data handling, showcasing concepts such as input validation, sorting algorithms, and data manipulation through slicing.

3. Create an array of random integer numbers as a numpy array, sort them and perform operations such as reshaping of the array into matrix of feasible dimensions. (e.g., if we have an array of 1 * 10, then we can reshape it into 2 * 5 or 5 * 2 matrix.) [Hint: Use the array of reshape (row * column)].

Answer:

The given python program below creates a random array, sorts it, and performs reshaping of array operation.

Following code for input:



```
An array of random integer numbers as a numpy array, sort them and perform operations such as reshaping of...
File Edit Format Run Options Window Help

import numpy as np

# Create a random array of 10 integers between 1 and 100.
random_array = np.random.randint(1, 100, size=10)
print("Original random array:", random_array)

# Sort the array
sorted_array = np.sort(random_array)
print("\nSorted array:", sorted_array)

# Reshape the array into different dimensions
try:
    # Reshape to 2x5
    reshaped_2x5 = sorted_array.reshape(2, 5)
    print("\nReshaped to 2x5 matrix:\n", reshaped_2x5)

    # Reshape to 5x2
    reshaped_5x2 = sorted_array.reshape(5, 2)
    print("\nReshaped to 5x2 matrix:\n", reshaped_5x2)

except ValueError as e:
    print("\nError:", e)
    print("Cannot reshape array of size", len(sorted_array),
          "into the requested dimensions")

print("\nAdditional reshaping examples:")
try:
    # Reshaping with 12 elements
    larger_array = np.random.randint(1, 100, size=12)
    sorted_larger = np.sort(larger_array)
    print("\nArray with 12 elements:", sorted_larger)

    # Possible reshapes for 12 elements
    print("\nReshaped to 3x4:\n", sorted_larger.reshape(3, 4))
    print("\nReshaped to 4x3:\n", sorted_larger.reshape(4, 3))
    print("\nReshaped to 6x2:\n", sorted_larger.reshape(6, 2))

except ValueError as e:
    print("Reshaping error:", e)
```

Output obtained in execution:

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/An array of random integer numbers as a numpy array, sort them and perform operations such as reshaping of the array.py
Original random array: [50 11 2 53 30 47 98 42 96 41]

Sorted array: [2 11 30 41 42 47 50 53 96 98]

Reshaped to 2x5 matrix:
[[2 11 30 41 42]
 [47 50 53 96 98]]

Reshaped to 5x2 matrix:
[[2 11]
 [30 41]
 [42 47]
 [50 53]
 [96 98]]

Additional reshaping examples:

Array with 12 elements: [14 43 46 58 67 69 73 85 89 92 97 99]

Reshaped to 3x4:
[[14 43 46 58]
 [67 69 73 85]
 [89 92 97 99]]

Reshaped to 4x3:
[[14 43 46]
 [58 67 69]
 [73 85 89]
 [92 97 99]]

Reshaped to 6x2:
[[14 43]
 [46 58]
 [67 69]
 [73 85]
 [89 92]
 [97 99]]
>>>

```

Python Program File: “An array of random integer number as a numpy array, sort them and perform operations such as reshaping of the array.py.”

Explanation of code:**Random Array Generation:**

The `'np.random.randint(1, 100, size=10)'` creates an array of 10 random integers between 1 and 100. The size of the parameter determines how many elements are generated in the array.

Sorting:

`'np.sort()'` sorts the array in ascending order which creates a new sorted array without modifying the original one.

Reshaping:

The method `'reshape(2, 5)'` converts the 1D array into a 2×5 matrix, another method, `'reshape(5, 2)'` converts it into a 5×2 matrix, and the product of dimensions must equal the original array size ($2 \times 5 = 10$, $2 \times 5 = 10$).

Error Handling:

The try-except block catches cases where the reshaping isn't possible in the program. For example, if user could not reshape a 10-element array into a 3×3 matrix (9 elements).

Additional Reshaping with Elements:

The program demonstrates reshaping with 12 elements and shows the possible reshapes like 2×5 , 2×5 , and 2×5 matrix.

Conclusion of the Program:

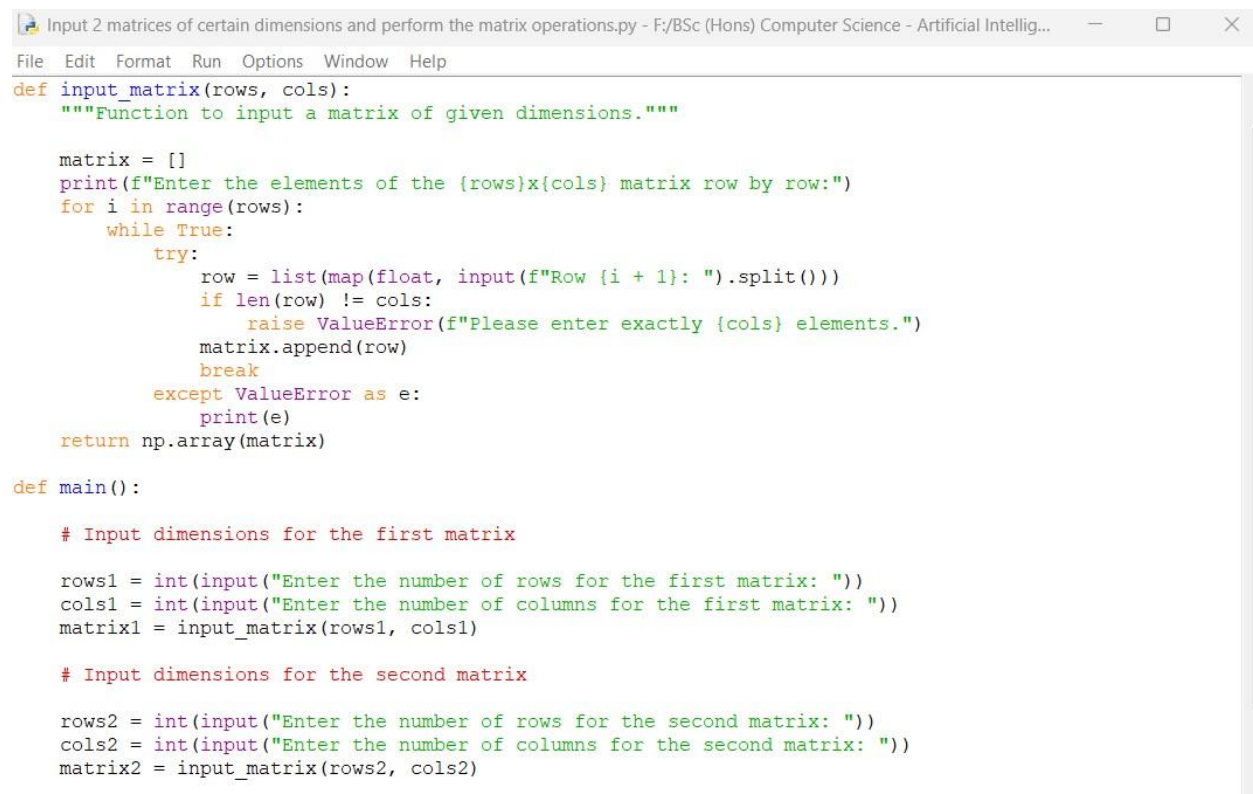
The given python program is an example of NumPy's capabilities for array and data manipulation. It showcases aspects such as random array generation, sorting, reshaping arrays, and error handling.

4. Write a program to input 2 matrices of certain dimensions and perform the matrix operations such as additions, subtraction, multiplication using numpy. Validation of matrix size should be done before the operations are performed. Mismatch of size for operations should raise the exception.

Answer:

The given python program below allows the user to input two matrices, validates their sizes, and performs matrix operations such as addition, subtraction, and multiplication using NumPy.

Following code for input:



```

Input 2 matrices of certain dimensions and perform the matrix operations.py - F:/BSc (Hons) Computer Science - Artificial Intellig...
File Edit Format Run Options Window Help
def input_matrix(rows, cols):
    """Function to input a matrix of given dimensions."""
    matrix = []
    print(f"Enter the elements of the {rows}x{cols} matrix row by row:")
    for i in range(rows):
        while True:
            try:
                row = list(map(float, input(f"Row {i + 1}: ").split()))
                if len(row) != cols:
                    raise ValueError(f"Please enter exactly {cols} elements.")
                matrix.append(row)
                break
            except ValueError as e:
                print(e)
    return np.array(matrix)

def main():
    # Input dimensions for the first matrix
    rows1 = int(input("Enter the number of rows for the first matrix: "))
    cols1 = int(input("Enter the number of columns for the first matrix: "))
    matrix1 = input_matrix(rows1, cols1)

    # Input dimensions for the second matrix
    rows2 = int(input("Enter the number of rows for the second matrix: "))
    cols2 = int(input("Enter the number of columns for the second matrix: "))
    matrix2 = input_matrix(rows2, cols2)
  
```


Continue:

```
# Perform matrix operations with validation
try:
    # Addition
    if matrix1.shape == matrix2.shape:
        addition_result = matrix1 + matrix2
        print("\nAddition Result:\n", addition_result)
    else:
        raise ValueError("Matrix addition requires both matrices to have the same dimensions.")

    # Subtraction
    if matrix1.shape == matrix2.shape:
        subtraction_result = matrix1 - matrix2
        print("\nSubtraction Result:\n", subtraction_result)
    else:
        raise ValueError("Matrix subtraction requires both matrices to have the same dimensions.")

    # Multiplication
    if cols1 == rows2:
        multiplication_result = np.dot(matrix1, matrix2)
        print("\nMultiplication Result:\n", multiplication_result)
    else:
        raise ValueError("Matrix multiplication requires the number of columns in the first matrix to be equal to the number of rows in the second matrix.")

except ValueError as e:
    print("Error:", e)

if __name__ == "__main__":
    main()
```

Output obtained in execution:

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Input 2 matrices of certain dimensions and perform the matrix operations.py
Enter the number of rows for the first matrix: 2
Enter the number of columns for the first matrix: 2
Enter the elements of the 2x2 matrix row by row:
Row 1: 1 2
Row 2: 3 4
Enter the number of rows for the second matrix: 2
Enter the number of columns for the second matrix: 2
Enter the elements of the 2x2 matrix row by row:
Row 1: 5 6
Row 2: 7 8

Addition Result:
[[ 6.  8.]
 [10. 12.]]

Subtraction Result:
[[-4. -4.]
 [-4. -4.]]

Multiplication Result:
[[19. 22.]
 [43. 50.]]
>>>

```

Python Program File: “Input 2 matrices of certain dimensions and perform the matrix operations.py.”

Explanation of code:

Input Function:

The ‘input_matrix’ function prompts the user to enter the elements of a matrix row by row. It validates that the correct number of elements is entered for each row.

Main Function:

The program starts by asking the dimensions of the first and second matrices from the user. It then calls the 'input_matrix' function to create the matrices.

Matrix Operations:

For addition, it checks if the shapes of the two matrices are the same before performing addition. For subtraction, it checks for shape compatibility. For multiplication, it checks if the number of columns in the first matrix matches the number of rows in the second matrix before performing the multiplication using 'np.dot()'.

Error Handling:

The program could raise exceptions with the messages if the matrices are not compatible for the requested operations.

Conclusion of the Program:

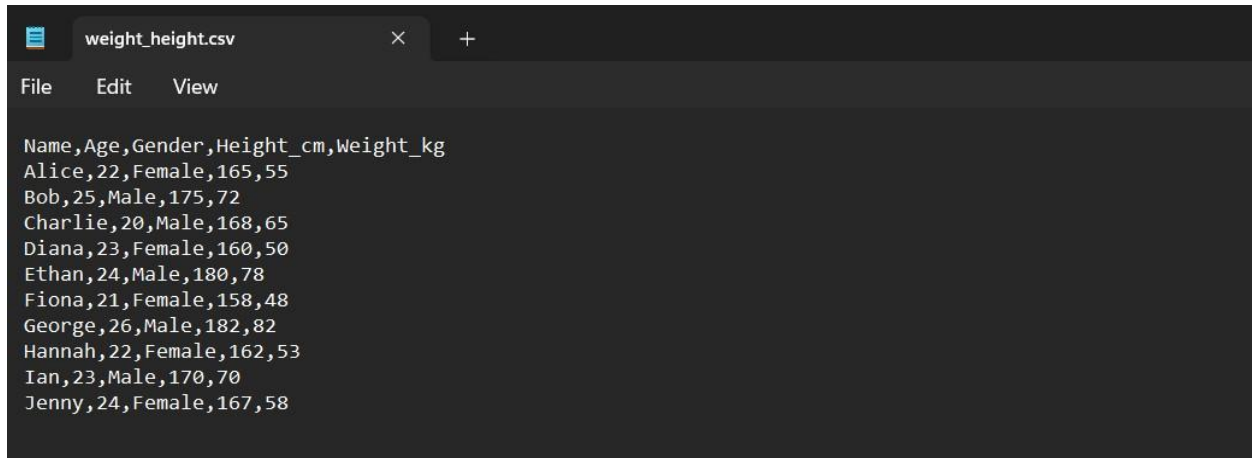
The given python program is an example of how to handle matrix operations in python using NumPy. It shows user input validation, error handling, and the use of dimensions in mathematical operations such addition, subtraction, and multiplication.

5. Write a program to read the csv file "weight_height.csv" using Pandas. Plot the data as a scatterplot (weight vs height, age vs weight, height vs age, gender vs height, gender vs weight) using Matplotlib library.

Answer:

The given python program below reads the CSV file “weight_height.csv” and plots the specified scatterplots using Pandas and Matplotlib. The program uses necessary libraries installed (‘pandas’ and ‘matplotlib’).

Read CSV file weight_height.csv:

A screenshot of a text editor window with a dark background. The title bar at the top shows a file icon, the name 'weight_height.csv', and window control buttons (close, maximize, and a plus sign). Below the title bar is a menu bar with 'File', 'Edit', and 'View'. The main text area contains the following CSV data:

```
Name,Age,Gender,Height_cm,Weight_kg
Alice,22,Female,165,55
Bob,25,Male,175,72
Charlie,20,Male,168,65
Diana,23,Female,160,50
Ethan,24,Male,180,78
Fiona,21,Female,158,48
George,26,Male,182,82
Hannah,22,Female,162,53
Ian,23,Male,170,70
Jenny,24,Female,167,58
```

Following code for input:

```

Read the csv file "weight_height.csv" using Pandas and plot the data as a scatterplot.py - F:/BSc (Hons) Computer Science - Artificial...
File Edit Format Run Options Window Help
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv("weight_height.csv")

# Display the first few rows of the dataframe
print(data.head())

# Create scatter plots
plt.figure(figsize=(15, 10))

# Scatter plot: Weight vs Height

plt.subplot(2, 2, 1)
plt.scatter(data['Height_cm'], data['Weight_kg'], color='blue')
plt.title('Weight vs Height')
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')

# Scatter plot: Age vs Weight

plt.subplot(2, 2, 2)
plt.scatter(data['Age'], data['Weight_kg'], color='green')
plt.title('Age vs Weight')
plt.xlabel('Age (years)')
plt.ylabel('Weight (kg)')

# Scatter plot: Height vs Age

plt.subplot(2, 2, 3)
plt.scatter(data['Height_cm'], data['Age'], color='red')
plt.title('Height vs Age')
plt.xlabel('Height (cm)')
plt.ylabel('Age (years)')

# Scatter plot: Gender vs Height

plt.subplot(2, 2, 4)
gender_colors = {'Male': 'blue', 'Female': 'pink'}
plt.scatter(data['Height_cm'], data['Gender'].map(gender_colors), color=data['Gender'].map(gender_colors))
plt.title('Gender vs Height')
plt.xlabel('Height (cm)')
plt.ylabel('Gender')

# Adjust layout
plt.tight_layout()
plt.show()

```

Output obtained in execution:

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)]
on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - W
eekly Task/Weekly Assignment - Week 7 and 8/Program Files/Read the csv file "weight_heig
ht.csv" using Pandas and plot the data as a scatterplot.py
Matplotlib is building the font cache; this may take a moment.
   Name  Age  Gender  Height_cm  Weight_kg
0  Alice   22  Female      165         55
1   Bob    25   Male      175         72
2  Charlie 20   Male      168         65
3  Diana   23  Female      160         50
4  Ethan   24   Male      180         78
Traceback (most recent call last):
  File "C:\Users\ASUS\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\indexes\base.py", line 3812, in get_loc
    return self._engine.get_loc(casted_key)
  File "pandas/_libs/index.pyx", line 167, in pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/index.pyx", line 196, in pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/hashtable_class_helper.pxi", line 7088, in pandas._libs.hashtable.P
yObjectHashTable.get_item
  File "pandas/_libs/hashtable_class_helper.pxi", line 7096, in pandas._libs.hashtable.P
yObjectHashTable.get_item
KeyError: 'Weight_kg'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Week
ly Task/Weekly Assignment - Week 7 and 8/Program Files/Read the csv file "weight_height.
csv" using Pandas and plot the data as a scatterplot.py", line 16, in <module>
    plt.scatter(data['Height_cm'], data['Weight_kg'], color='blue')
  File "C:\Users\ASUS\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\frame.py", line 4107, in __getitem__
    indexer = self.columns.get_loc(key)
  File "C:\Users\ASUS\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\indexes\base.py", line 3819, in get_loc
    raise KeyError(key) from err
KeyError: 'Weight_kg'
>>>

```

Python Program File: “Read the csv file “weight_height.csv” using Pandas and plot the data as a scatterplot.py.”

Explanation of code:**Import Libraries:**

The program imports the necessary libraries, 'pandas' for data manipulation and 'matplotlib.pyplot' for plotting.

Read CSV File:

The program reads the CSV file "weight_height.csv" into a Pandas DataFrame using 'pd.read_csv()'.

Display the Data:

It prints the first few rows of the DataFrame to verify that the data has been loaded correctly.

Create Scatter Plots:

The program creates a figure with a specified size using 'plt.figure()' and creates four scatter plots in 2×2 grids. There is weight vs height, where the plots height on the x – axis and weight on the y – axis, age vs weight, where the plots age on the x – axis and age on the y – axis, height vs age, where the plots height on the x -axis and age on the y – axis, and gender vs height, where plots height on the x-axis and uses color to represent gender.

Adjust Layout:

The 'plt.tight_layout()' function is called to adjust the spacing between subplots for better visibility.

Show plots:

Finally, 'plt.show()' is called to display the plots.

Output of the Program:

After you run the program, the window displaying scatter plots based on the data from the CSV file is visible. Each plot is shown to have a relationship between the specified variables.

Conclusion of the Program:

The given python program is an example to execute a script to read a CSV file and plot a scatterplot using libraries such as Pandas and Matplotlib.

6. Read the data from csv file “weight_height.csv” in a data frame using Pandas. Add 2 additional columns (BMI and Risk) in the existing DataFrame. Add the data according to the calculations given below.

BMI = Weight / Height

Risk values vary according to the conditions given below:

BMI less than 18.5 : Nutrient deficient

BMI between 18.5 and 24.9: lower risk

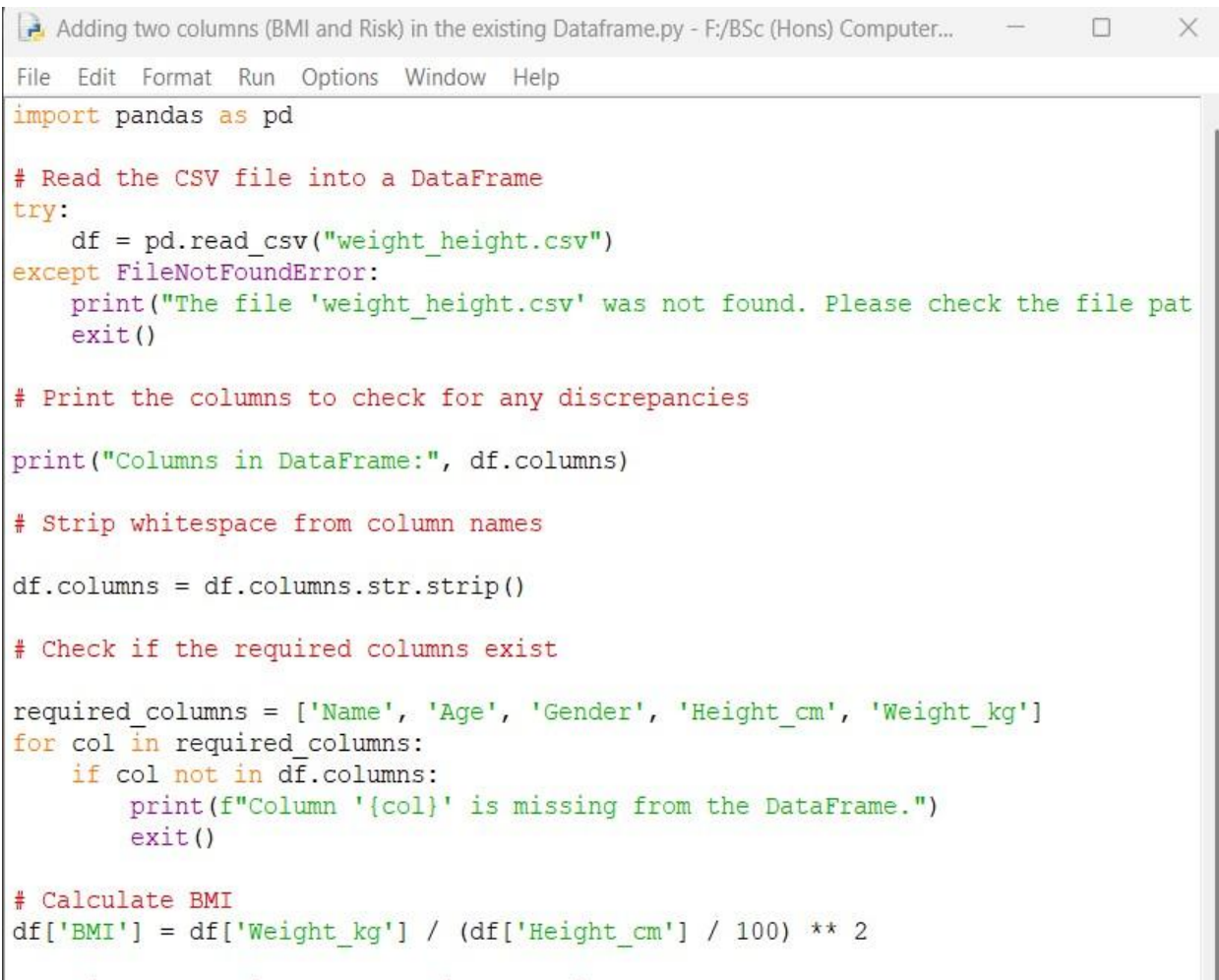
BMI between 25 and 29.9: Heart disease risk

BMI between 30 and 34.9: Higher risk of diabetes, heart disease

BMI 40 or higher: Serious health condition risk

Answer:

The given python program below to read the data from the CSV file “weight_height.csv” and add the BMI and Risk columns to the Dataframe using Pandas.

Following code for input:

```
import pandas as pd

# Read the CSV file into a DataFrame
try:
    df = pd.read_csv("weight_height.csv")
except FileNotFoundError:
    print("The file 'weight_height.csv' was not found. Please check the file path")
    exit()

# Print the columns to check for any discrepancies
print("Columns in DataFrame:", df.columns)

# Strip whitespace from column names
df.columns = df.columns.str.strip()

# Check if the required columns exist
required_columns = ['Name', 'Age', 'Gender', 'Height_cm', 'Weight_kg']
for col in required_columns:
    if col not in df.columns:
        print(f"Column '{col}' is missing from the DataFrame.")
        exit()

# Calculate BMI
df['BMI'] = df['Weight_kg'] / (df['Height_cm'] / 100) ** 2
```

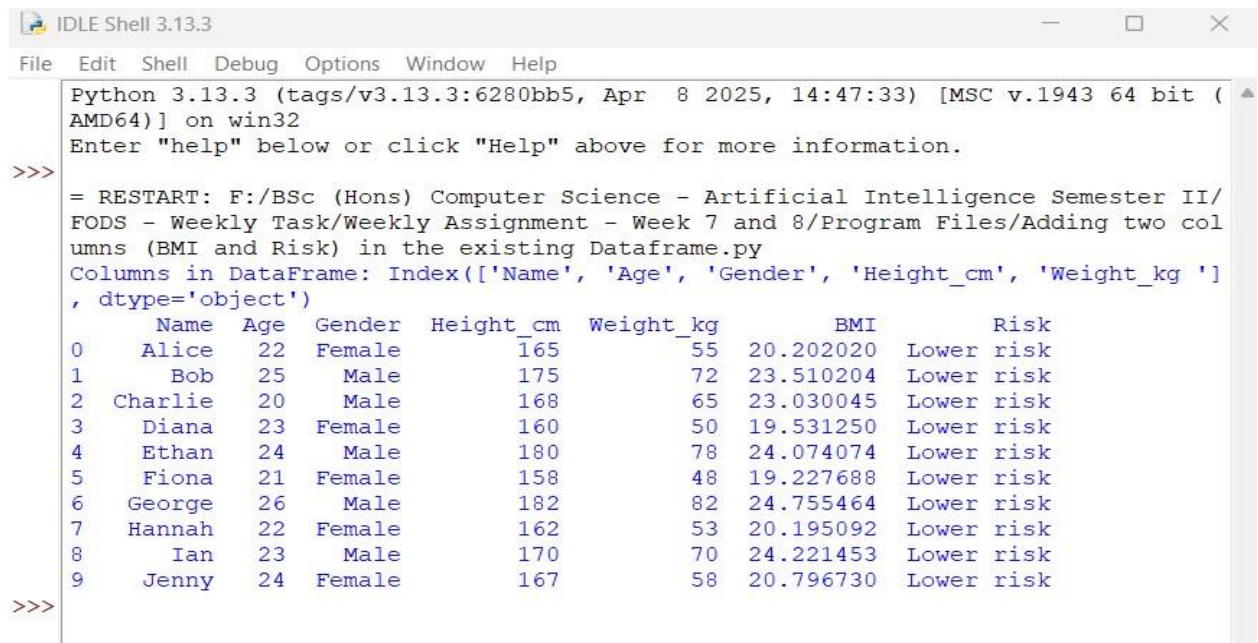
Continue:

```
# Define a function to determine the risk category based on BMI
def determine_risk(bmi):
    if bmi < 18.5:
        return 'Nutrient deficient'
    elif 18.5 <= bmi < 24.9:
        return 'Lower risk'
    elif 25 <= bmi < 29.9:
        return 'Heart disease risk'
    elif 30 <= bmi < 34.9:
        return 'Higher risk of diabetes, heart disease'
    elif bmi >= 40:
        return 'Serious health condition risk'
    else:
        return 'Unknown'

# function to create the Risk column
df['Risk'] = df['BMI'].apply(determine_risk)

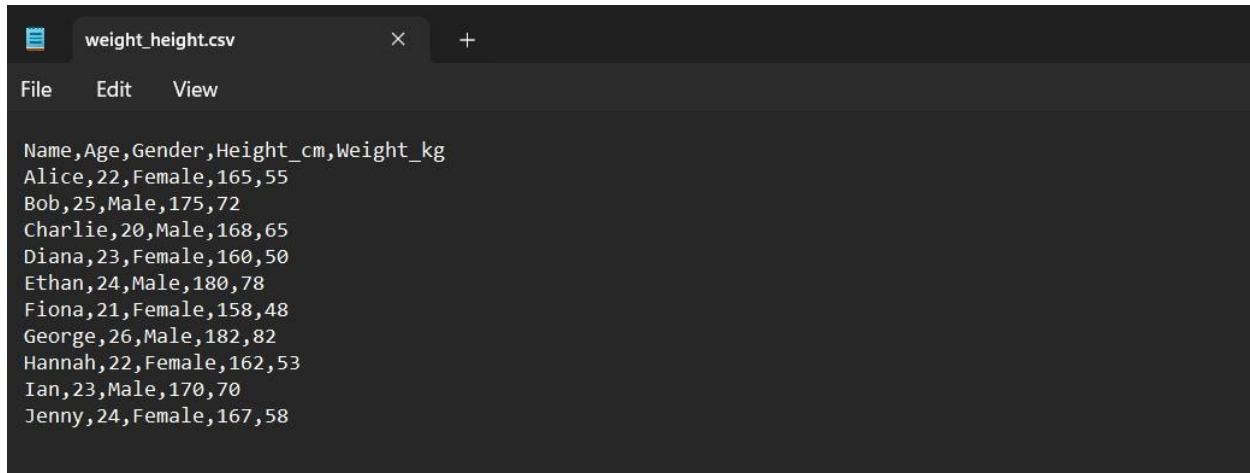
# Display the updated DataFrame
print(df)

# Save the updated DataFrame to a new CSV file
df.to_csv("updated_weight_height.csv", index=False)
```

Output obtained in execution:


```
IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 7 and 8/Program Files/Adding two col
umns (BMI and Risk) in the existing Dataframe.py
Columns in DataFrame: Index(['Name', 'Age', 'Gender', 'Height_cm', 'Weight_kg '],
dtype='object')
  Name  Age  Gender  Height_cm  Weight_kg      BMI      Risk
0  Alice   22  Female      165         55  20.202020  Lower risk
1   Bob   25   Male      175         72  23.510204  Lower risk
2  Charlie  20   Male      168         65  23.030045  Lower risk
3  Diana   23  Female      160         50  19.531250  Lower risk
4  Ethan   24   Male      180         78  24.074074  Lower risk
5  Fiona   21  Female      158         48  19.227688  Lower risk
6  George  26   Male      182         82  24.755464  Lower risk
7  Hannah  22  Female      162         53  20.195092  Lower risk
8   Ian   23   Male      170         70  24.221453  Lower risk
9  Jenny   24  Female      167         58  20.796730  Lower risk
>>>
```

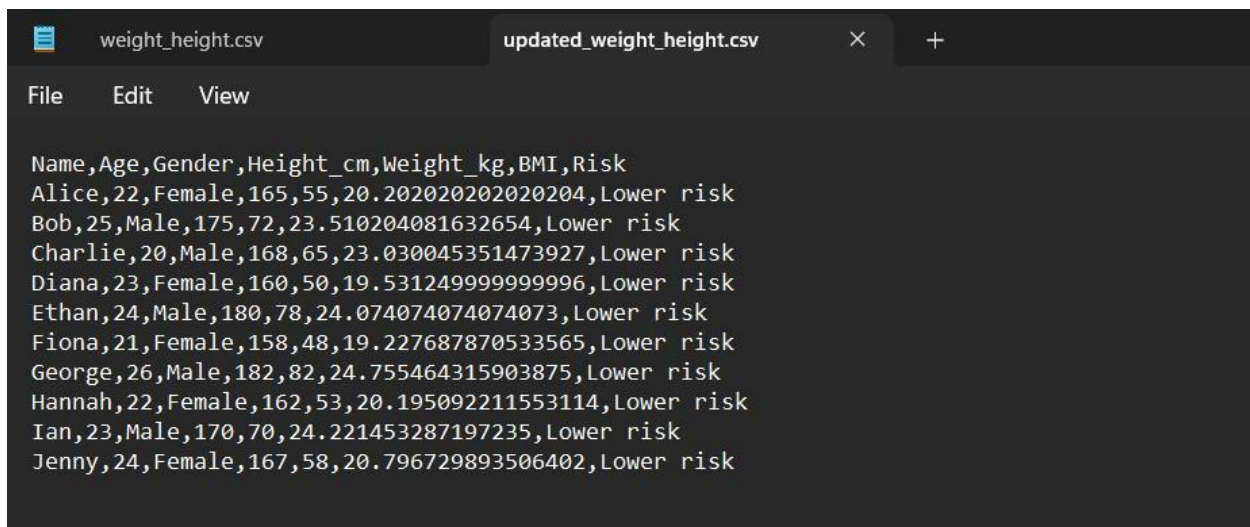
Original CSV File “weight_height.csv”:



The screenshot shows a text editor window with the file 'weight_height.csv' open. The editor has a menu bar with 'File', 'Edit', and 'View'. The content of the file is as follows:

```
Name,Age,Gender,Height_cm,Weight_kg
Alice,22,Female,165,55
Bob,25,Male,175,72
Charlie,20,Male,168,65
Diana,23,Female,160,50
Ethan,24,Male,180,78
Fiona,21,Female,158,48
George,26,Male,182,82
Hannah,22,Female,162,53
Ian,23,Male,170,70
Jenny,24,Female,167,58
```

Updated CSV File “updated_weight_height.csv”:



The screenshot shows a text editor window with two files open: 'weight_height.csv' and 'updated_weight_height.csv'. The 'updated_weight_height.csv' file is the active one. The editor has a menu bar with 'File', 'Edit', and 'View'. The content of the file is as follows:

```
Name,Age,Gender,Height_cm,Weight_kg,BMI,Risk
Alice,22,Female,165,55,20.202020202020204,Lower risk
Bob,25,Male,175,72,23.510204081632654,Lower risk
Charlie,20,Male,168,65,23.030045351473927,Lower risk
Diana,23,Female,160,50,19.531249999999996,Lower risk
Ethan,24,Male,180,78,24.074074074074073,Lower risk
Fiona,21,Female,158,48,19.227687870533565,Lower risk
George,26,Male,182,82,24.755464315903875,Lower risk
Hannah,22,Female,162,53,20.195092211553114,Lower risk
Ian,23,Male,170,70,24.221453287197235,Lower risk
Jenny,24,Female,167,58,20.796729893506402,Lower risk
```

Python Program File: “Adding two columns (BMI and Risk) in the existing Dataframe.py.”

Explanation of code:**Importing Pandas:**

The code begins by importing the Pandas library, which is used for data manipulation and analysis.

Reading the CSV File:

The `'pd.read_csv'` function reads the `weight_height.csv` file into a DataFrame named `'df'`.

Error Handling:

A `'try-except'` block is used to catch a `'FileNotFoundError'`. If the file is not found, an error message is printed, and the program exits.

Printing Column Names:

The current column names of the DataFrame are printed to the console. This helps in identifying any inconsistency in the column names section.

Stripping Whitespace:

The `'str.strip()'` method is applied to the column names to remove any leading or trailing whitespace, which can cause issues when accessing the columns.

Required Columns Check:

A list of required column names is defined. The code iterates through the list to check if each column exists in the DataFrame.

Error Handling for Missing Columns:

An error message is printed and programs exits if any required column is missing.

BMI Calculation:

The Body Mass Index (BMI) is calculated using the formula $BMI = \frac{Weight}{Height^2}$. The height is converted from centimeters to meters by dividing by 100. The result is stored in a new column named 'BMI'.

Risk Determination Function:

A function 'determine_risk' is defined to categorize the health risk based on the BMI value. The use of conditional statements to return the appropriate risk category.

Creating the Risk Column:

The 'apply' method is used to apply the 'determine_risk' function to each value in the 'BMI' column. The results are stored in a new resulting column named 'Risk'.

Displaying the Dataframe:

The updated DataFrame, which now includes the columns 'BMI' and 'Risk', which is printed to the console.

Saving the DataFrame:

The method 'to_csv' is used to save the updated DataFrame to a new CSV file named "updated_weight_height.csv". The 'index=False' argument prevents Pandas from writing the row indices to the CSV file, resulting in a output.

Output of the Program:

The program first prints the column names of the DataFrame, which helps the user to verify that the data has been read correctly. After the calculating the BMI and the risk determination categories, the program prints the updated DataFrame. The program saves the updated DataFrame to a new CSV file named "updated_weight_height.csv". The file will contain all the original data along with the newly calculated BMI and Risk columns.

Conclusion of the Program:

The given python program reads a CSV file containing individuals' weight and height data, it calculates the Body Mass Index (BMI), and their category wise health risk based on the BMI values.