Alliance School of Advanced Computing

Department of Computer Science and Engineering

Class Assignment-1

Course Code: 5CS1025
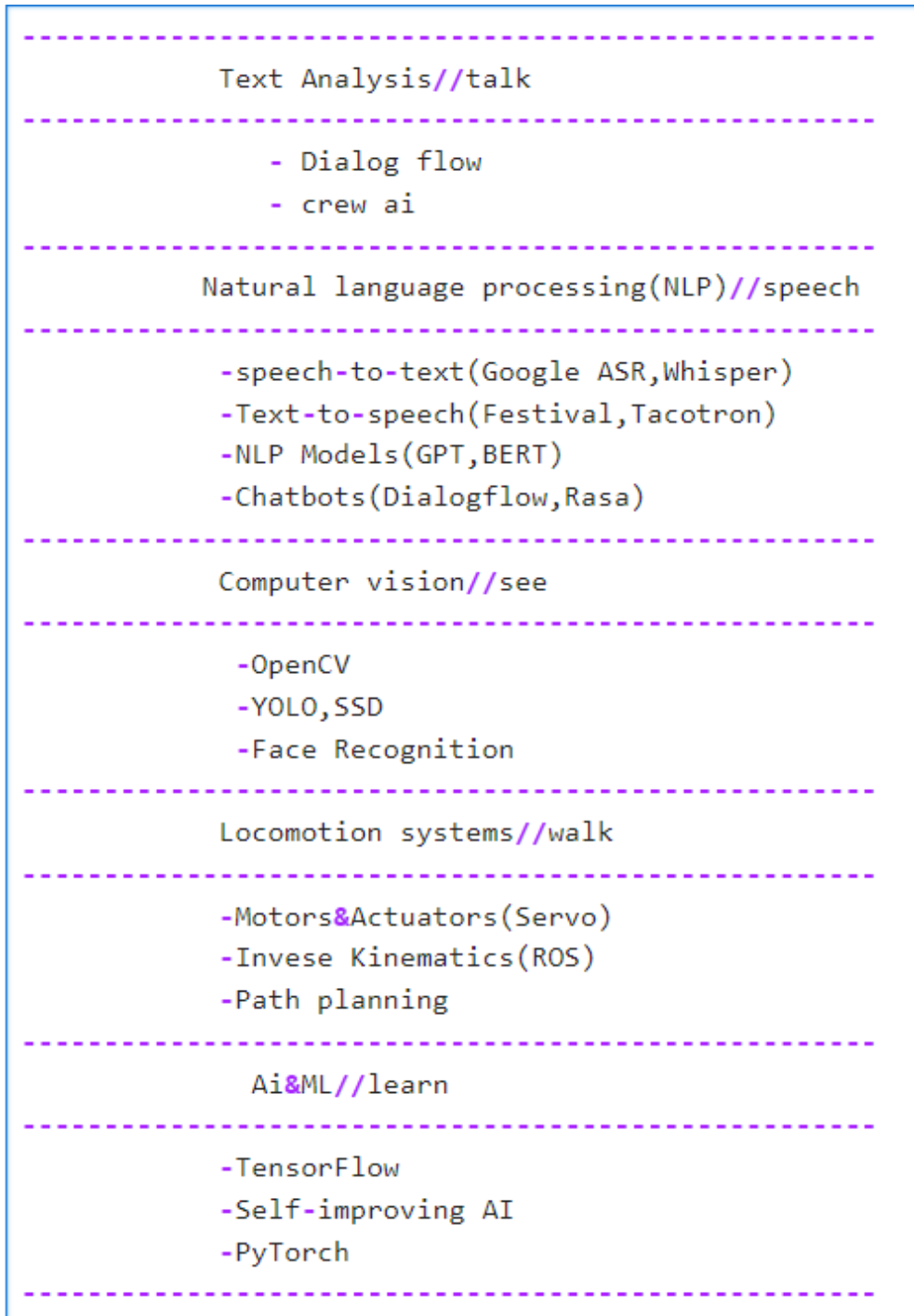**Course Title:** Artificial Intelligence
Semester: 04
Class : AIML

Summitted by: Kamili Reddy Rushitha

Reg no:2023BCSE07AED313

1. Imagine you are tasked with designing a humanoid robot to assist in a home or office environment. The robot must be capable of interacting with people by talking and listening, walking to different locations, seeing and recognizing objects, and learning from its surroundings to adapt its behavior. What technologies, tools, and frameworks would you need to build such a robot? Give as flow chart.

```
-------------------------------------------------
        Text Analysis//talk
-------------------------------------------------
            - Dialog flow
            - crew ai
-------------------------------------------------
    Natural language processing(NLP)//speech
-------------------------------------------------
        -speech-to-text(Google ASR,Whisper)
        -Text-to-speech(Festival,Tacotron)
        -NLP Models(GPT,BERT)
        -Chatbots(Dialogflow,Rasa)
-------------------------------------------------
        Computer vision//see
-------------------------------------------------
         -OpenCV
         -YOLO,SSD
         -Face Recognition
-------------------------------------------------
    Locomotion systems//walk
-------------------------------------------------
        -Motors&Actuators(Servo)
        -Invese Kinematics(ROS)
        -Path planning
-------------------------------------------------
         Ai&ML//learn
-------------------------------------------------
        -TensorFlow
        -Self-improving AI
        -PyTorch
-------------------------------------------------
```

2. Calculate and interpret mean, median, mode, variance and standard deviation for a given dataset.
Data =[15,21,29,21,15,24,32,21,15,30]

```python
[1]: import numpy as np
     data=[15,21,29,21,15,24,32,21,15,30]
     mean=np.mean(data)
     median=np.median(data)
     variance=np.var(data)
     std_dev=np.std(data)
     print("mean is ",mean)
     print("median is",median)
     print("variance is ",variance)
     print("standard deviation ",std_dev)

     mean is  22.3
     median is 21.0
     variance is  36.61
     standard deviation  6.050619802962338
```

3. You are analyzing a dataset that captures the daily performance and activity of a humanoid robot in a simulated environment. The dataset link robot_dataset(robot_dataset)_1.csv includes the following attributes

What is the average (mean) number of conversations the robot has daily?
2) Find the total steps walked by the robot over a given period.
3) Determine the maximum and minimum energy consumption in the dataset.
4) Calculate the correlation between the number of steps walked and energy consumption.
5) Analyze the distribution of objects recognized daily (e.g., histogram or box plot).
6) What is the variance in the number of learning sessions completed?.

```python
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv("robot_dataset(robot_dataset)_1(in).csv")

# Compute required statistics
mean_interactions = df["Interaction_Count"].mean()
total_steps_walked = df["Steps_Walked"].sum()
max_energy = df["Energy_Consumption (kWh)"].max()
min_energy = df["Energy_Consumption (kWh)"].min()
correlation = df["Steps_Walked"].corr(df["Energy_Consumption (kWh)"])
variance = df["Learning_Sessions"].var()

# Print results
print(f"The average number of conversations the robot has daily is {mean_interactions}")
print(f"The total steps walked by the robot over a given period is {total_steps_walked}")
print(f"The maximum energy consumption in the dataset is {max_energy}")
print(f"The minimum energy consumption in the dataset is {min_energy}")
print(f"Correlation between the number of steps walked and energy consumption is {correlation}")
print(f"The variance in the number of learning sessions completed is {variance}")

The average number of conversations the robot has daily is 5.51
The total steps walked by the robot over a given period is 14379
The maximum energy consumption in the dataset is 3.0
The minimum energy consumption in the dataset is 1.0
Correlation between the number of steps walked and energy consumption is 0.0015478137393314497
The variance in the number of learning sessions completed is 391.9422845691382
```

4. Write a Python program that declares variables of different data types (e.g., string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.

```python
student_name = "Aarav"  # String
age = 20  # Integer
cgpa = 8.9  # Float
is_graduated = False  # Boolean
# Printing the variables in a sentence format
print(f"The student's name is",student_name)
print(f"He is age years old.",age)
print(f"His current CGPA is",cgpa)
print(f"Has he graduated? is_graduated",is_graduated)
```

```
The student's name is Aarav
He is age years old. 20
His current CGPA is 8.9
Has he graduated? is_graduated False
```

5. Write a Python program that takes an integer input and checks whether the number is positive, negative, or zero using conditional statements (if-else)
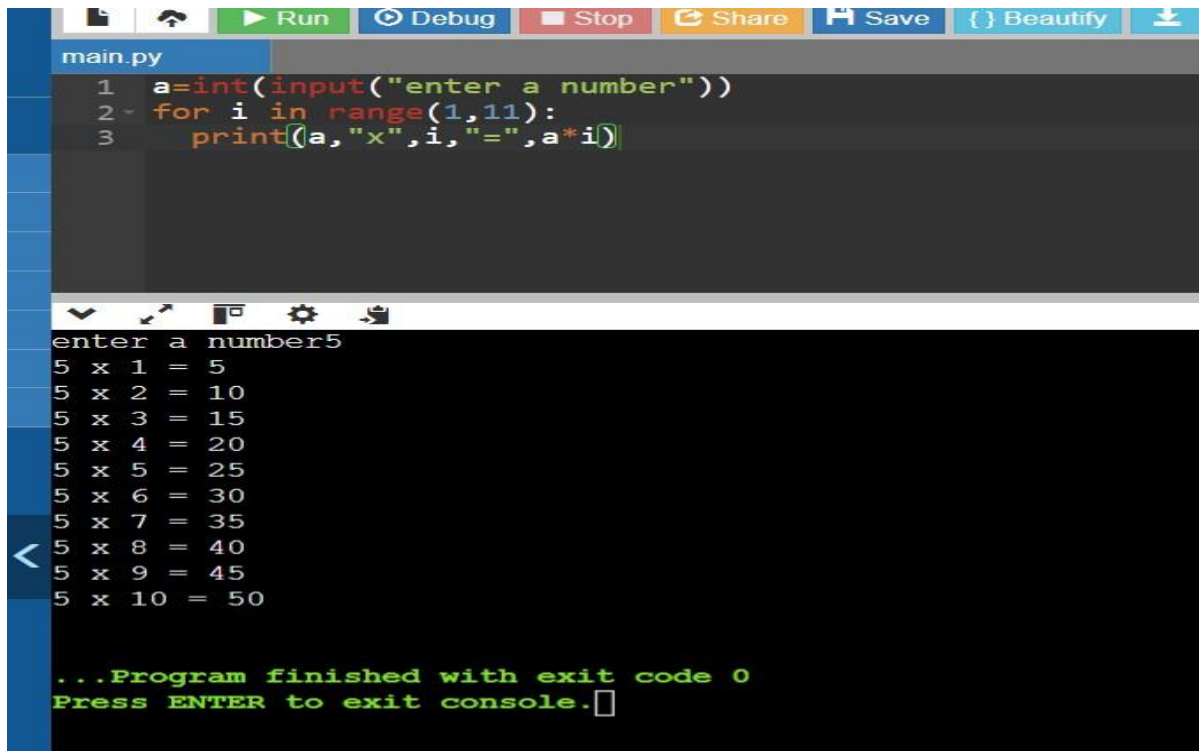
```python
[1]: num = int(input("Enter an integer: "))
     if num > 0:
         print("The number is positive.")
     elif num < 0:
         print("The number is negative.")
     else:
         print("The number is zero.")
```

```
Enter an integer:  12
The number is positive.
```

```python
[2]: num = int(input("Enter an integer: "))
     if num > 0:
         print("The number is positive.")
     elif num < 0:
         print("The number is negative.")
     else:
         print("The number is zero.")
```

```
Enter an integer:  -34
The number is negative.
```

6. Write a Python program that takes a number as input and prints the multiplication table for that number (from 1 to 10).

```
main.py
1   a=int(input("enter a number"))
2 - for i in range(1,11):
3       print(a,"x",i,"=",a*i)
```

```
enter a number5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50


...Program finished with exit code 0
Press ENTER to exit console.
```

7. Create a Python list that contains the names of 5 different fruits. Perform the given operations on the list.

```python
[2]: fruits = ["Apple", "Banana", "Cherry", "Mango", "Grapes"]
# Operations on the list
# 1. Add a fruit to the list
fruits.append("Orange")
print("List after adding Orange:", fruits)
# 2. Remove a fruit from the list
fruits.remove("Banana")
print("List after removing Banana:", fruits)
# 3. Access a fruit by index
print("Fruit at index 2:", fruits[2])
# 4. Sort the list alphabetically
fruits.sort()
print("List after sorting:", fruits)
# 5. Reverse the list
fruits.reverse()
print("List after reversing:", fruits)
# 6. Count the number of fruits in the list
print("Total number of fruits in the list:", len(fruits))
```

```
List after adding Orange: ['Apple', 'Banana', 'Cherry', 'Mango', 'Grapes', 'Orange']
List after removing Banana: ['Apple', 'Cherry', 'Mango', 'Grapes', 'Orange']
Fruit at index 2: Mango
List after sorting: ['Apple', 'Cherry', 'Grapes', 'Mango', 'Orange']
List after reversing: ['Orange', 'Mango', 'Grapes', 'Cherry', 'Apple']
Total number of fruits in the list: 5
```

8. Write a Python program that creates a tuple containing 5 numbers. Perform the given operations on the tuple.

```
[4]: numbers = (5, 10, 15, 20, 25)
     # Operations on the tuple
     # 1. Access an element by index
     print("Element at index 2:", numbers[2])
     # 2. Count the occurrences of a number in the tuple
     count_15 = numbers.count(15)
     print("Count of 15 in the tuple:", count_15)
     # 3. Find the index of a number in the tuple
     index_of_20 = numbers.index(20)
     print("Index of 20 in the tuple:", index_of_20)
     # 4. Length of the tuple
     print("Length of the tuple:", len(numbers))
     # 5. Slicing the tuple (extracting a portion of the tuple)
     sliced_tuple = numbers[1:4]
     print("Sliced portion of the tuple (from index 1 to 3):", sliced_tuple)
```

```
Element at index 2: 15
Count of 15 in the tuple: 1
Index of 20 in the tuple: 3
Length of the tuple: 5
Sliced portion of the tuple (from index 1 to 3): (10, 15, 20)
```

9. Create a dictionary that stores the names of 3 students as keys and their marks in mathematics as values. Perform the given operations.

```
]: students_marks = {"Alice": 85,"Bob": 92,"Charlie": 78}
   # Operations on the dictionary
   # 1. Accessing the marks of a specific student
   print("Marks of Bob:", students_marks["Bob"])
   # 2. Adding a new student with marks
   students_marks["David"] = 88
   print("Dictionary after adding David:", students_marks)
   # 3. Updating the marks of an existing student
   students_marks["Alice"] = 90
   print("Dictionary after updating Alice's marks:", students_marks)
   # 4. Removing a student from the dictionary
   del students_marks["Charlie"]
   print("Dictionary after removing Charlie:", students_marks)
   # 5. Checking if a student exists in the dictionary
   if "Bob" in students_marks:
       print("Bob is in the dictionary.")
   else:
       print("Bob is not in the dictionary.")
   # 6. Getting the number of students (length of the dictionary)
   print("Number of students:", len(students_marks))
```

```
Marks of Bob: 92
Dictionary after adding David: {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 88}
Dictionary after updating Alice's marks: {'Alice': 90, 'Bob': 92, 'Charlie': 78, 'David': 88}
Dictionary after removing Charlie: {'Alice': 90, 'Bob': 92, 'David': 88}
Bob is in the dictionary.
Number of students: 3
```

10. Create two sets of integers. Perform the given set operations.

```python
set_1 = {10, 20, 30, 40, 50}
set_2 = {30, 40, 50, 60, 70}
# Operations on the sets
# 1. Union of the two sets (all elements from both sets)
union_set = set_1 | set_2
print("Union of set_1 and set_2:", union_set)
# 2. Intersection of the two sets (common elements)
intersection_set = set_1 & set_2
print("Intersection of set_1 and set_2:", intersection_set)
# 3. Difference of the two sets (elements in set_1 but not in set_2)
difference_set = set_1 - set_2
print("Difference of set_1 and set_2 (set_1 - set_2):", difference_set)
# 4. Symmetric Difference (elements in either set_1 or set_2, but not in both)
symmetric_difference_set = set_1 ^ set_2
print("Symmetric Difference of set_1 and set_2:", symmetric_difference_set)
# 5. Subset check (is set_1 a subset of set_2?)
is_subset = set_1 <= set_2
print("Is set_1 a subset of set_2?", is_subset)
# 6. Superset check (is set_1 a superset of set_2?)
is_superset = set_1 >= set_2
print("Is set_1 a superset of set_2?", is_superset)
# 7. Checking if an element exists in a set
element_check = 30 in set_1
print("Does element 30 exist in set_1?", element_check)
```

```
Union of set_1 and set_2: {70, 40, 10, 50, 20, 60, 30}
Intersection of set_1 and set_2: {40, 50, 30}
Difference of set_1 and set_2 (set_1 - set_2): {10, 20}
Symmetric Difference of set_1 and set_2: {20, 70, 10, 60}
Is set_1 a subset of set_2? False
Is set_1 a superset of set_2? False
Does element 30 exist in set_1? True
```

11. Write a Python function called find_largest() that takes a list of numbers as input and returns the largest number from the list. Test the function with a sample list.

```python
def find_largest(numbers):
    if not numbers:
        return None
    largest = max(numbers)
    return largest
sample_list = [12, 45, 7, 89, 34, 56, 23]
largest_number = find_largest(sample_list)
print("The largest number in the list is:", largest_number)
```
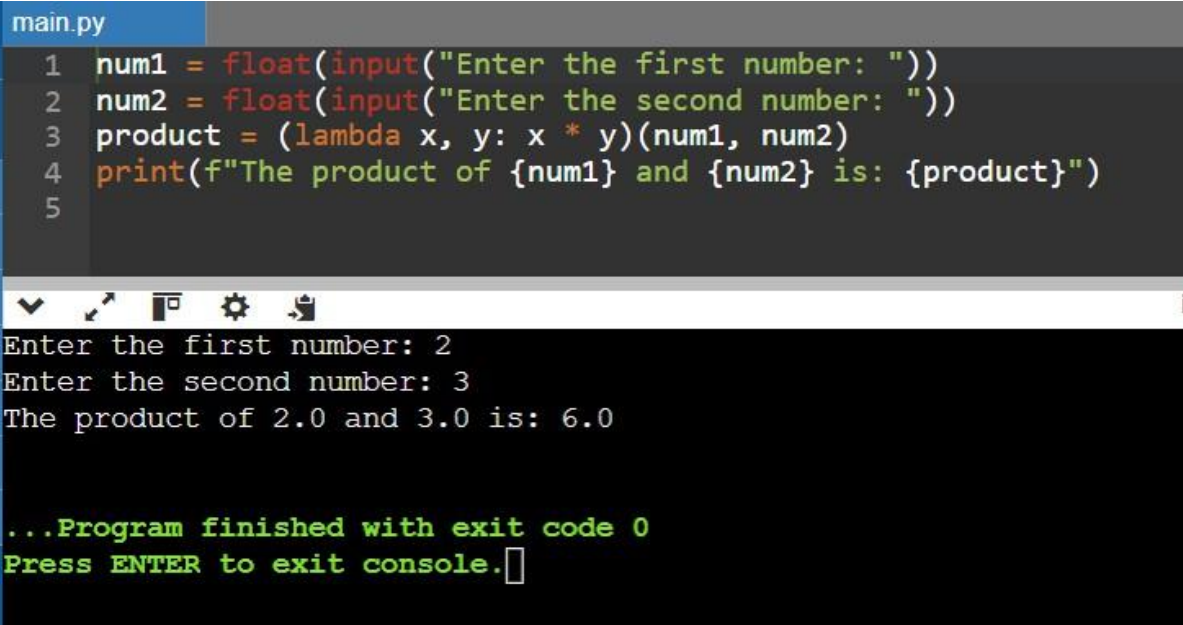
```
The largest number in the list is: 89
```

12. Use list comprehension to create a list of squares of all even numbers between 1 and 20.

```
even_squares = [x**2 for x in range(1, 21) if x % 2 == 0]
# Print the resulting list
print("List of squares of even numbers between 1 and 20:", even_squares)
```

List of squares of even numbers between 1 and 20: [4, 16, 36, 64, 100, 144, 196, 256, 324, 400]

13. Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user.

```
main.py
1  num1 = float(input("Enter the first number: "))
2  num2 = float(input("Enter the second number: "))
3  product = (lambda x, y: x * y)(num1, num2)
4  print(f"The product of {num1} and {num2} is: {product}")
5
```

```
Enter the first number: 2
Enter the second number: 3
The product of 2.0 and 3.0 is: 6.0


...Program finished with exit code 0
Press ENTER to exit console.
```

14. Write a Python program to create a one-dimensional, two-dimensional, and three-dimensional NumPy array. Print the shape and dimensions of each array.

```python
import numpy as np
one_d_array = np.array([1, 2, 3, 4, 5])
two_d_array = np.array([[1, 2, 3], [4, 5, 6]])
three_d_array = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("One-dimensional array:")
print("Array:", one_d_array)
print("Shape:", one_d_array.shape)
print("Dimensions:", one_d_array.ndim)
print("\nTwo-dimensional array:")
print("Array:\n", two_d_array)
print("Shape:", two_d_array.shape)
print("Dimensions:", two_d_array.ndim)
print("\nThree-dimensional array:")
print("Array:\n", three_d_array)
print("Shape:", three_d_array.shape)
print("Dimensions:", three_d_array.ndim)
```

o/p

```
One-dimensional array:
Array: [1 2 3 4 5]
Shape: (5,)
Dimensions: 1

Two-dimensional array:
Array:
 [[1 2 3]
 [4 5 6]]
Shape: (2, 3)
Dimensions: 2

Three-dimensional array:
Array:
 [[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
Shape: (2, 2, 2)
Dimensions: 3
```

15. Write a Python program to create a 5x5 NumPy array of random integers and Perform array indexing as given.

```python
import numpy as np
array = np.random.randint(0, 100, size=(5, 5))
print("Generated 5x5 NumPy Array of Random Integers:")
print(array)
element = array[1, 2]
print("\nElement at row 2, column 3:", element)
third_row = array[2]
print("\n3rd Row of the Array:", third_row)
fourth_column = array[:, 3]
print("\n4th Column of the Array:", fourth_column)
subarray = array[1:3, 2:4]
print("\nSubarray (rows 1 to 2 and columns 2 to 3):")
print(subarray)
last_row = array[-1]
last_column = array[:, -1]
print("\nLast Row of the Array:", last_row)
print("Last Column of the Array:", last_column)
```

o/p

```
Generated 5x5 NumPy Array of Random Integers:
[[78 32 87 11 89]
 [ 1 74 33 99 81]
 [86 63 90 90 42]
 [16 23 71 37 22]
 [22 82 58 92 72]]

33ment at row 2, column 3:

[86 63 90 90 42]ray:

[11 99 90 37 92] Array:

Subarray (rows 1 to 2 and columns 2 to 3):
[[33 99]
 [90 90]]

[22 82 58 92 72]rray:
Last Column of the Array: [89 81 42 22 72]
```

16. create a NumPy array of shape (4, 4) containing numbers from 1 to 16. Use slicing to extract for the given conditions

```
6]:  import numpy as np
     array_4x4 = np.arange(1, 17).reshape(4, 4)
     print("Original 4x4 Array:")
     print(array_4x4)
     sliced_array = array_4x4[:2, :2]
     print("\nSliced Array (first 2 rows and 2 columns):")
     print(sliced_array)
```

```
Original 4x4 Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Sliced Array (first 2 rows and 2 columns):
[[1 2]
 [5 6]]
```

17. Write a Python program that creates a 2D array of shape (6, 2) using np.arange() and then reshapes it into a 3D array of shape (2, 3, 2). Flatten the reshaped array and print the result.

```
3]:  array_2d = np.arange(1, 13).reshape(6, 2)
     print("Original 2D Array (6, 2):")
     print(array_2d)
     array_3d = array_2d.reshape(2, 3, 2)
     print("\nReshaped 3D Array (2, 3, 2):")
     print(array_3d)
     flattened_array = array_3d.flatten()
     print("\nFlattened Array:")
     print(flattened_array)
```

```
Original 2D Array (6, 2):
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]

Reshaped 3D Array (2, 3, 2):
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]

Flattened Array:
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

18. Write a Python program to demonstrate broadcasting. Create an array of shape (3, 3) and add a one-dimensional array of shape (1, 3) to it using broadcasting.

```
array_3x3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Original 3x3 Array:")
print(array_3x3)
array_1d = np.array([10, 20, 30])
print("\n1D Array to be added:")
print(array_1d)
broadcasted_result = array_3x3 + array_1d
print("\nResult after broadcasting:")
print(broadcasted_result)
```

```
Original 3x3 Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

1D Array to be added:
[10 20 30]

Result after broadcasting:
[[11 22 33]
 [14 25 36]
 [17 28 39]]
```

19. Create two NumPy arrays of the same shape, A and B. Perform the following arithmetic operations:
Element-wise addition.
Element-wise subtraction.
Element-wise multiplication.
Element-wise division.

```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[7, 8, 9], [10, 11, 12]])
print("Array A:")
print(A)
print("\nArray B:")
print(B)
add_result = A + B
print("\nElement-wise Addition:")
print(add_result)
sub_result = A - B
print("\nElement-wise Subtraction:")
print(sub_result)
mul_result = A * B
print("\nElement-wise Multiplication:")
print(mul_result)
div_result = A / B
print("\nElement-wise Division:")
print(div_result)

Array A:
```

o/p:

```
Array A:
[[1 2 3]
 [4 5 6]]

Array B:
[[ 7  8  9]
 [10 11 12]]

Element-wise Addition:
[[ 8 10 12]
 [14 16 18]]

Element-wise Subtraction:
[[-6 -6 -6]
 [-6 -6 -6]]

Element-wise Multiplication:
[[ 7 16 27]
 [40 55 72]]

Element-wise Division:
[[0.14285714 0.25       0.33333333]
 [0.4        0.45454545 0.5       ]]
```

20. Create a Pandas DataFrame with the given Name and marks of 3 courses:
Add a new column named 'Total' that represents the sum of all the courses. Add 'Grade' based on the values of the 'Total'. Print the updated DataFrame with the new 'Total' and 'Grade' column.

```python
import pandas as pd
data = {
    'Name': ['John', 'Alice', 'Bob'],
    'Math': [85, 92, 78],
    'Science': [88, 94, 82],
    'English': [90, 89, 75]
}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
df['Total'] = df['Math'] + df['Science'] + df['English']
def assign_grade(total):
    if total >= 270:
        return 'A'
    elif total >= 240:
        return 'B'
    elif total >= 210:
        return 'C'
    else:
        return 'D'
df['Grade'] = df['Total'].apply(assign_grade)
print("\nUpdated DataFrame with 'Total' and 'Grade':")
print(df)
```

o/p

```
Original DataFrame:
    Name  Math  Science  English
0   John    85       88       90
1  Alice    92       94       89
2    Bob    78       82       75

Updated DataFrame with 'Total' and 'Grade':
    Name  Math  Science  English  Total Grade
0   John    85       88       90    263     B
1  Alice    92       94       89    275     A
2    Bob    78       82       75    235     C
```