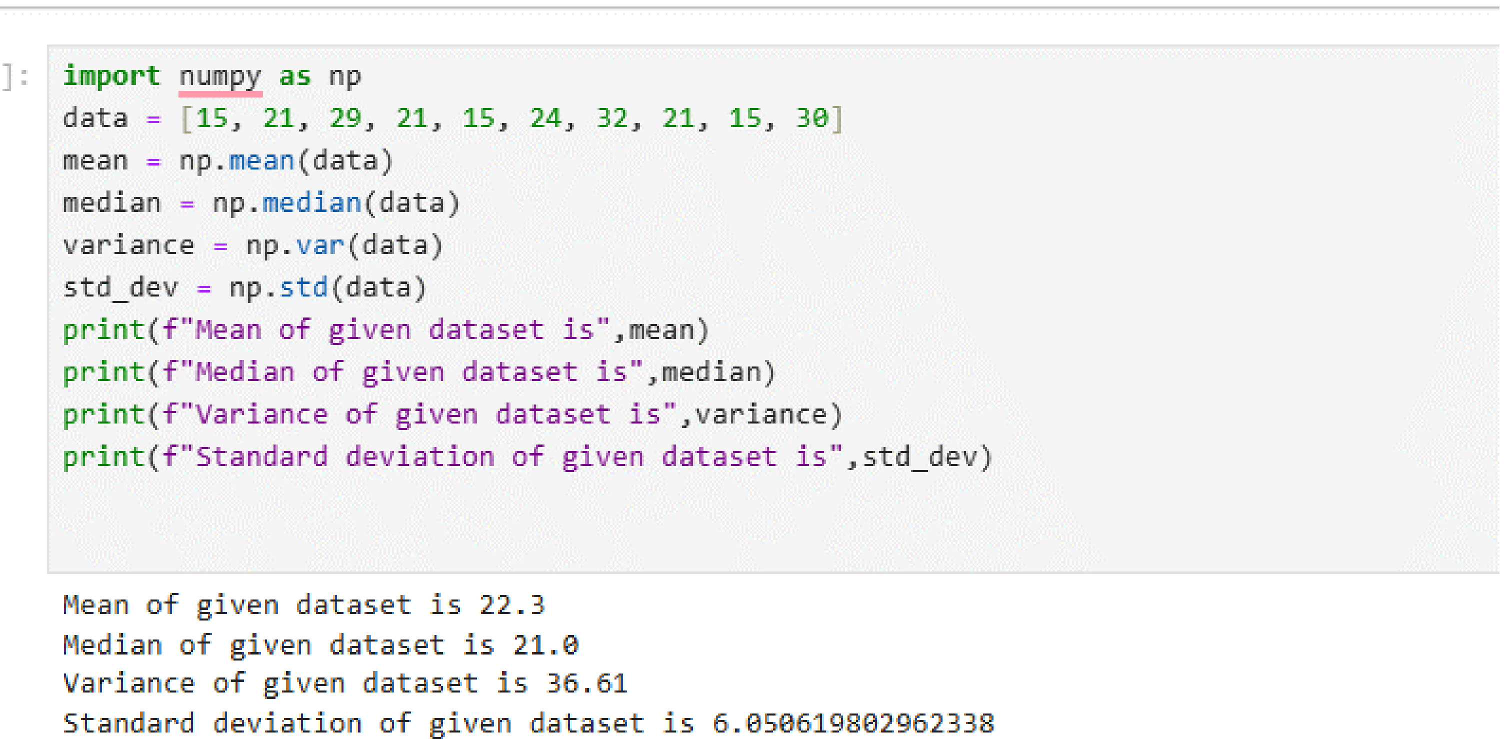
1.Imagine you are tasked with designing a humanoid robot to assist in a home or office environment. The robot must be capable of interacting with people by talking and listening, walking to different locations, seeing and recognizing objects, and learning from its surroundings to adapt its behaviour.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

Technologies, Tools, and Frameworks Required

1. **Speech Processing**
    - **Automatic Speech Recognition (ASR):**
        - Google Speech-to-Text
        - Mozilla DeepSpeech
    - **Text-to-Speech (TTS):**
        - Google Text-to-Speech
2. **Computer Vision**
    - **Object & Face Recognition:**
        - OpenCV
        - TensorFlow/Keras for deep learning
        - YOLO (You Only Look Once) for real-time object detection
3. **Decision-Making & AI Learning**
    - **Reinforcement Learning & AI Models:**
        - TensorFlow, PyTorch
        - OpenAI Gym (for training robotic behaviors)
4. **Motion Control & Robotics Frameworks**
    - **Robotics Middleware:**
        - ROS (Robot Operating System)
    - **Embedded Systems & Sensors:**
        - Arduino / Raspberry Pi (Prototyping)
        - NVIDIA Jetson (AI Edge Computing)
        - Motor Controllers, IMUs, Servo Motor

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]

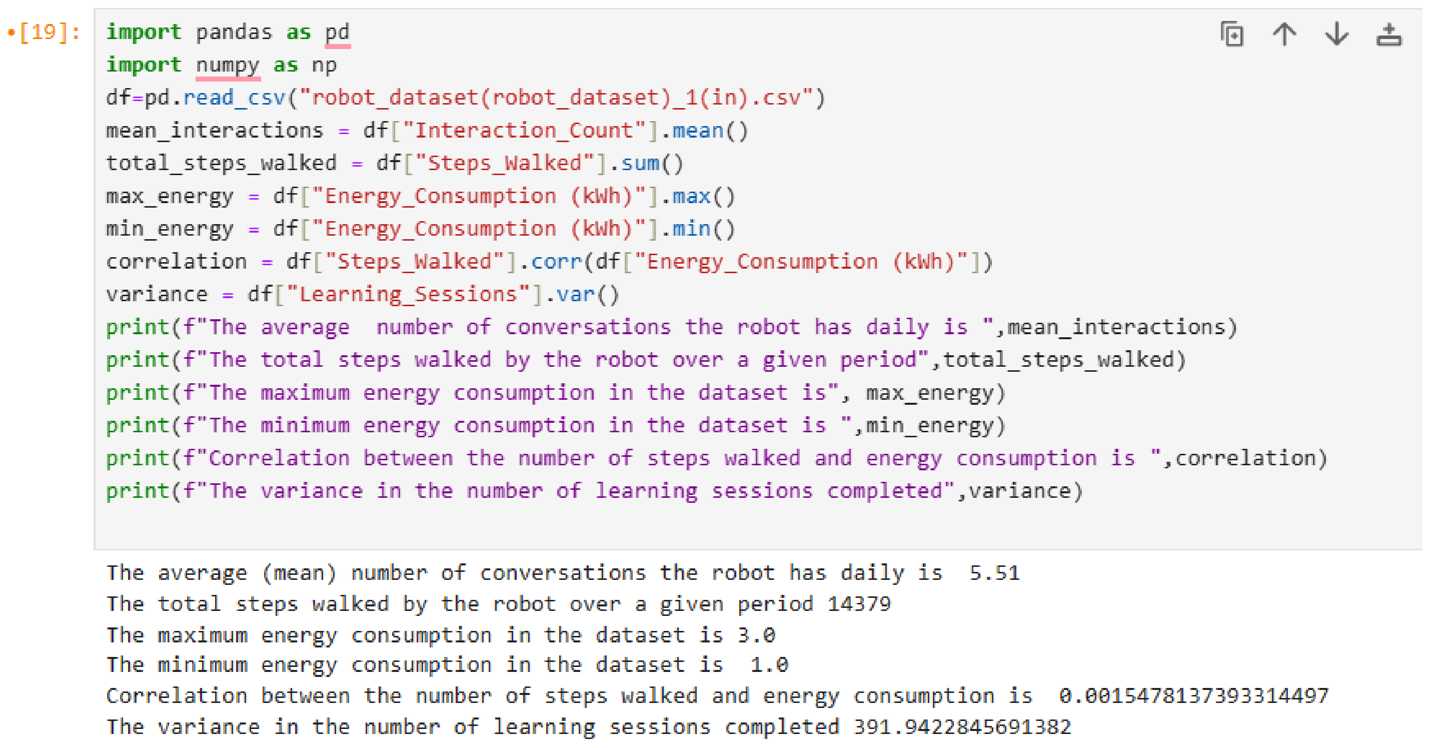


```
import numpy as np
data = [15, 21, 29, 21, 15, 24, 32, 21, 15, 30]
mean = np.mean(data)
median = np.median(data)
variance = np.var(data)
std_dev = np.std(data)
print(f"Mean of given dataset is",mean)
print(f"Median of given dataset is",median)
print(f"Variance of given dataset is",variance)
print(f"Standard deviation of given dataset is",std_dev)
```

```
Mean of given dataset is 22.3
Median of given dataset is 21.0
Variance of given dataset is 36.61
Standard deviation of given dataset is 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

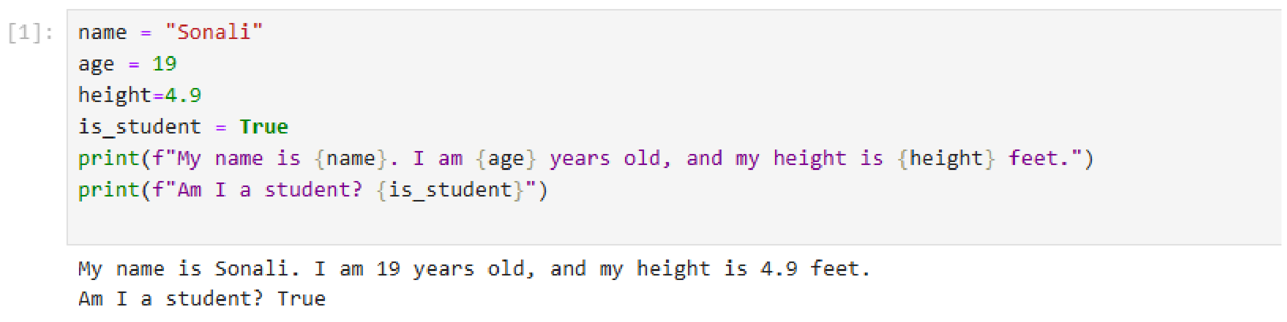
1) What is the average (mean) number of conversations the robot has daily?
2) Find the total steps walked by the robot over a given period.
3) Determine the maximum and minimum energy consumption in the dataset.
4) Calculate the correlation between the number of steps walked and energy consumption.
5)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



```
•[19]: import pandas as pd
       import numpy as np
       df=pd.read_csv("robot_dataset(robot_dataset)_1(in).csv")
       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(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",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",variance)
```

```
The average (mean) number of conversations the robot has daily is  5.51
The total steps walked by the robot over a given period 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 391.9422845691382
```

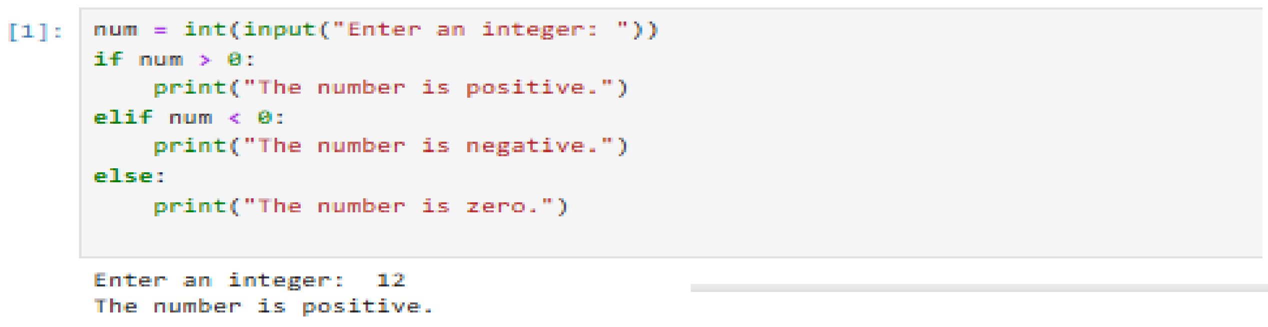
4.Write a Python program that declares variables of different datatypes(e.g.string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.



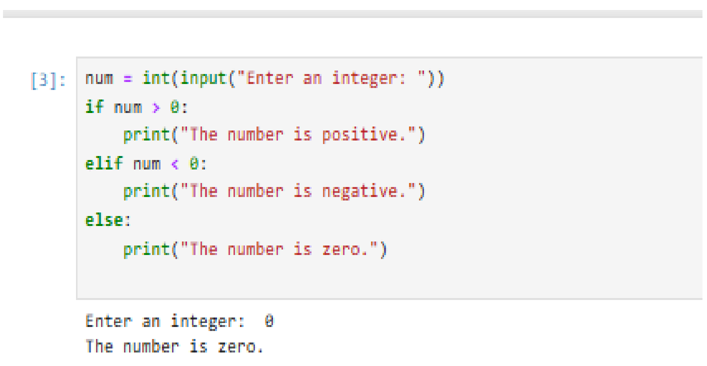
```
[1]: name = "Sonali"
     age = 19
     height=4.9
     is_student = True
     print(f"My name is {name}. I am {age} years old, and my height is {height} feet.")
     print(f"Am I a student? {is_student}")
```

```
My name is Sonali. I am 19 years old, and my height is 4.9 feet.
Am I a student? True
```

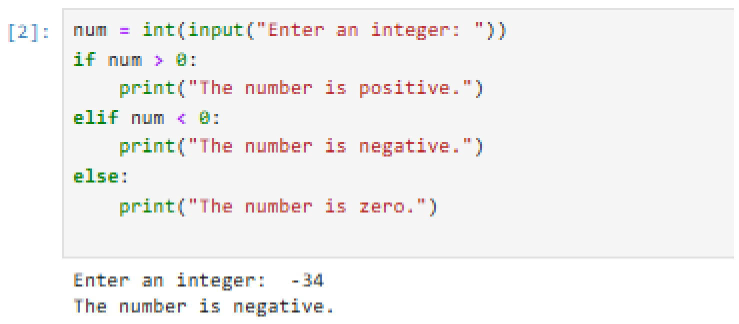
5.Write a Python program that takes an integer input and checks whether the number is positive, negative, or zero using conditional statements (if-else)



```
[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.
```

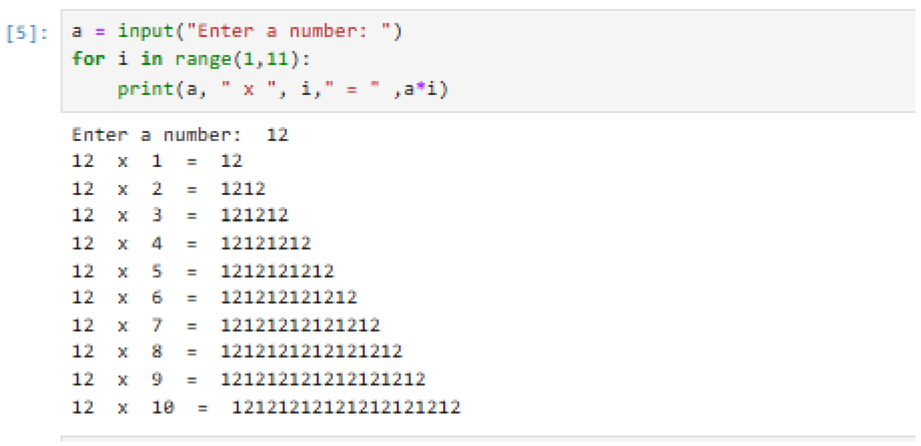


```
[3]: num = int(input("Enter an integer: "))
     if num > 0:
         print("The number is positive.")
     elif num < 0:
         print("The number is negative.")
     else:
         print("The number is zero.")

     Enter an integer:  0
     The number is zero.
```



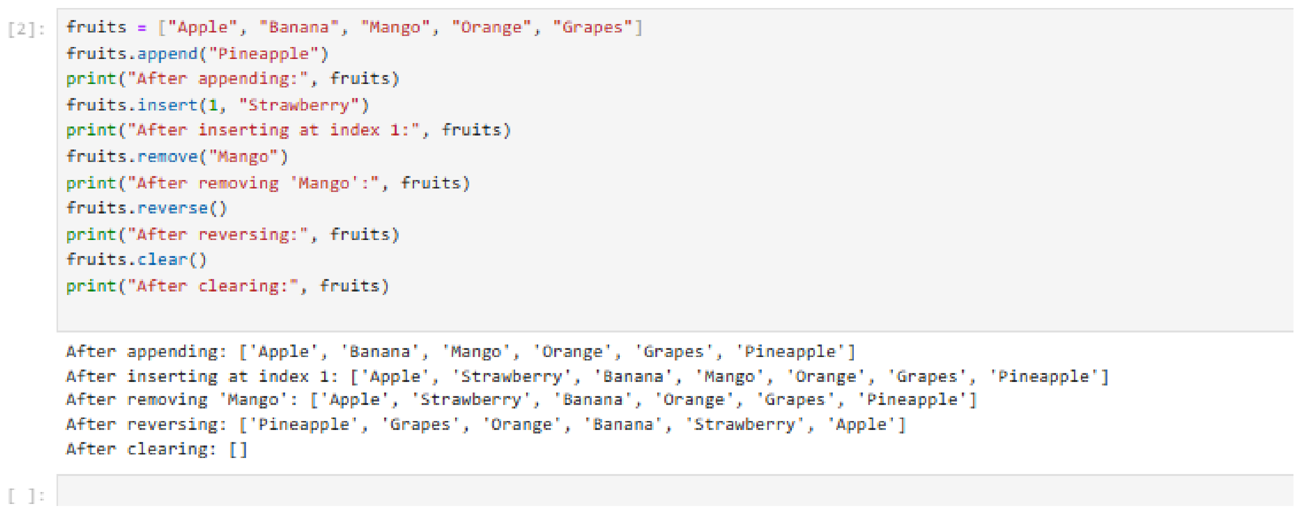
```
[2]: num = int(input("Enter an integer: "))
     if num > 0:
         print("The number is positive.")
     elif num < 0:
         print("The number is negative.")
     else:
         print("The number is zero.")

     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).



```
[5]: a = input("Enter a number: ")
     for i in range(1,11):
         print(a, " x ", i," = " ,a*i)

     Enter a number:  12
     12  x  1  =  12
     12  x  2  =  1212
     12  x  3  =  121212
     12  x  4  =  12121212
     12  x  5  =  1212121212
     12  x  6  =  121212121212
     12  x  7  =  12121212121212
     12  x  8  =  1212121212121212
     12  x  9  =  121212121212121212
     12  x  10  =  12121212121212121212
```

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

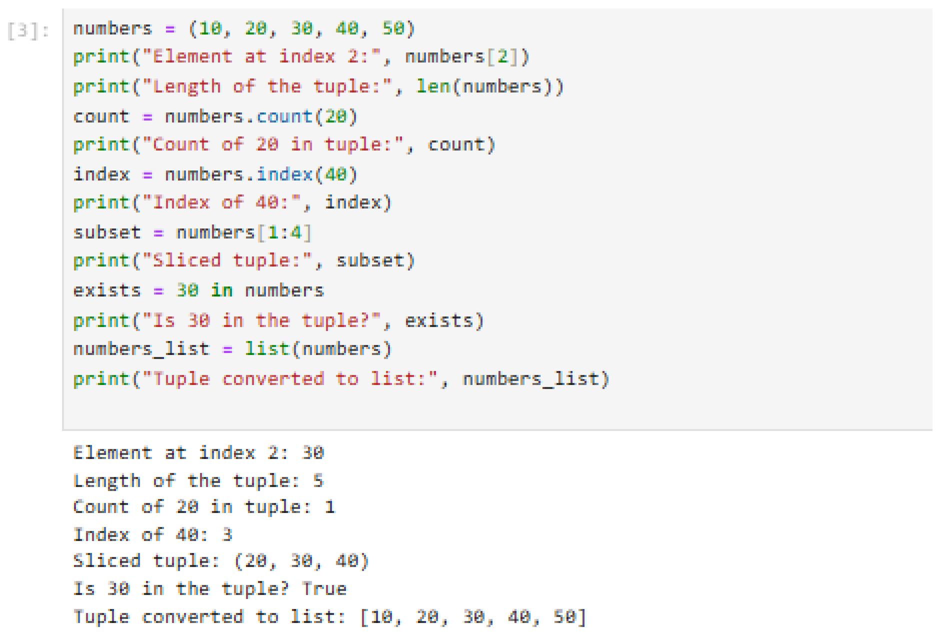


```
[2]: fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"]
     fruits.append("Pineapple")
     print("After appending:", fruits)
     fruits.insert(1, "Strawberry")
     print("After inserting at index 1:", fruits)
     fruits.remove("Mango")
     print("After removing 'Mango':", fruits)
     fruits.reverse()
     print("After reversing:", fruits)
     fruits.clear()
     print("After clearing:", fruits)
```

```
After appending: ['Apple', 'Banana', 'Mango', 'Orange', 'Grapes', 'Pineapple']
After inserting at index 1: ['Apple', 'Strawberry', 'Banana', 'Mango', 'Orange', 'Grapes', 'Pineapple']
After removing 'Mango': ['Apple', 'Strawberry', 'Banana', 'Orange', 'Grapes', 'Pineapple']
After reversing: ['Pineapple', 'Grapes', 'Orange', 'Banana', 'Strawberry', 'Apple']
After clearing: []
```

```
[ ]:
```

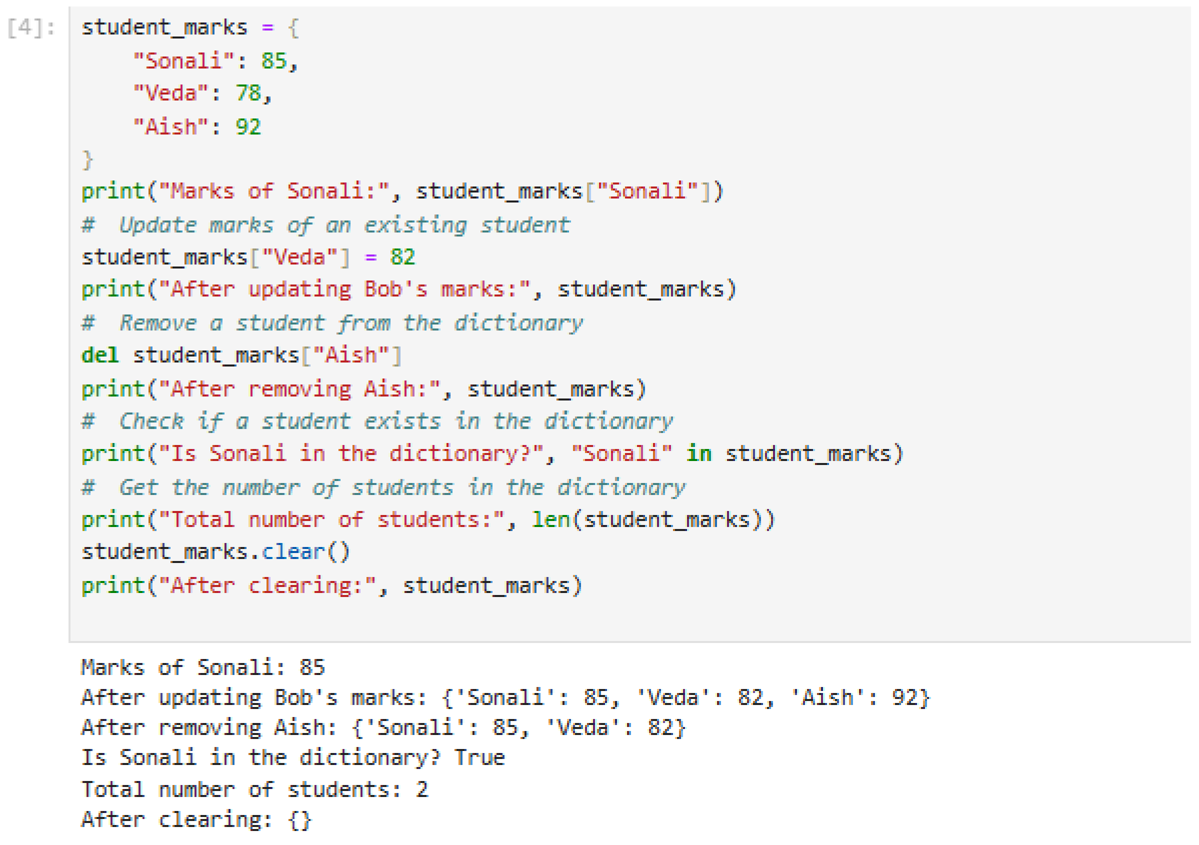
8.write a Python program that creates a tuple containing 5 numbers. Perform the given operations on the tuple.



```
[3]: numbers = (10, 20, 30, 40, 50)
     print("Element at index 2:", numbers[2])
     print("Length of the tuple:", len(numbers))
     count = numbers.count(20)
     print("Count of 20 in tuple:", count)
     index = numbers.index(40)
     print("Index of 40:", index)
     subset = numbers[1:4]
     print("Sliced tuple:", subset)
     exists = 30 in numbers
     print("Is 30 in the tuple?", exists)
     numbers_list = list(numbers)
     print("Tuple converted to list:", numbers_list)
```

```
Element at index 2: 30
Length of the tuple: 5
Count of 20 in tuple: 1
Index of 40: 3
Sliced tuple: (20, 30, 40)
Is 30 in the tuple? True
Tuple converted to list: [10, 20, 30, 40, 50]
```

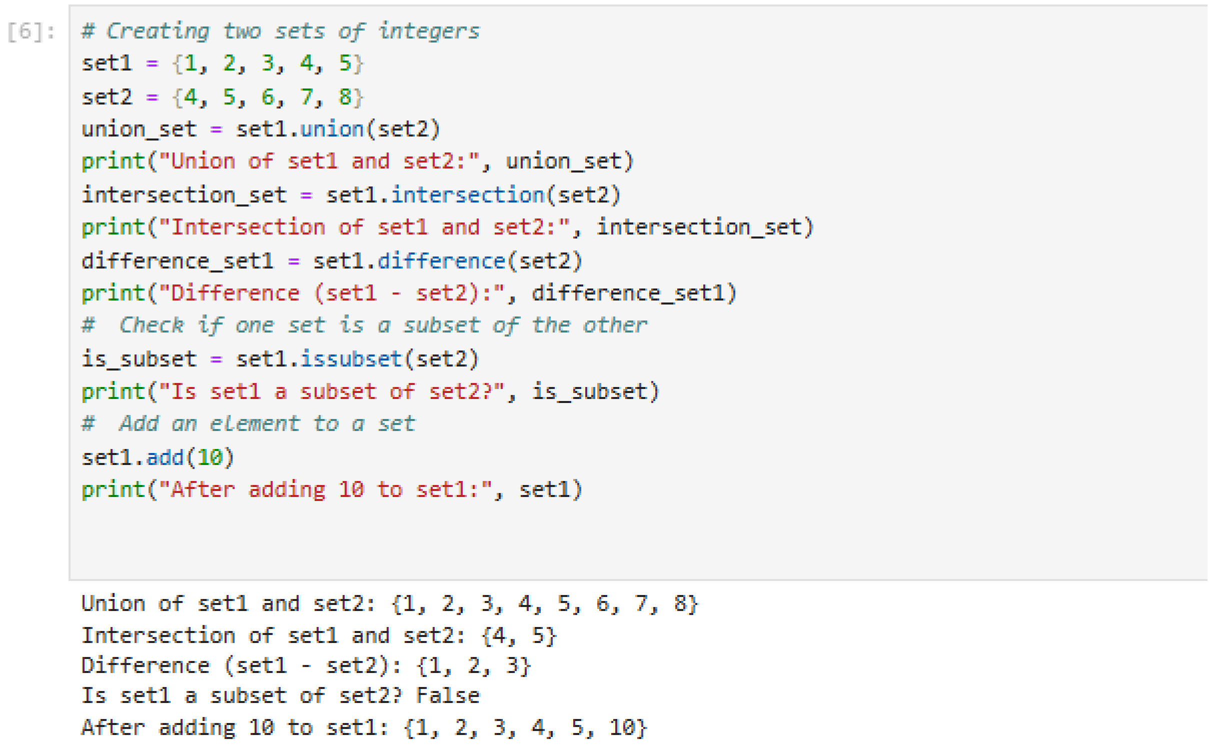
9.Create a dictionary that stores the names of 3 students as keys and their marks in mathematics as values. Perform the given operations.



```
[4]: student_marks = {
         "Sonali": 85,
         "Veda": 78,
         "Aish": 92
     }
     print("Marks of Sonali:", student_marks["Sonali"])
     #  Update marks of an existing student
     student_marks["Veda"] = 82
     print("After updating Bob's marks:", student_marks)
     #  Remove a student from the dictionary
     del student_marks["Aish"]
     print("After removing Aish:", student_marks)
     #  Check if a student exists in the dictionary
     print("Is Sonali in the dictionary?", "Sonali" in student_marks)
     #  Get the number of students in the dictionary
     print("Total number of students:", len(student_marks))
     student_marks.clear()
     print("After clearing:", student_marks)
```

```
Marks of Sonali: 85
After updating Bob's marks: {'Sonali': 85, 'Veda': 82, 'Aish': 92}
After removing Aish: {'Sonali': 85, 'Veda': 82}
Is Sonali in the dictionary? True
Total number of students: 2
After clearing: {}
```

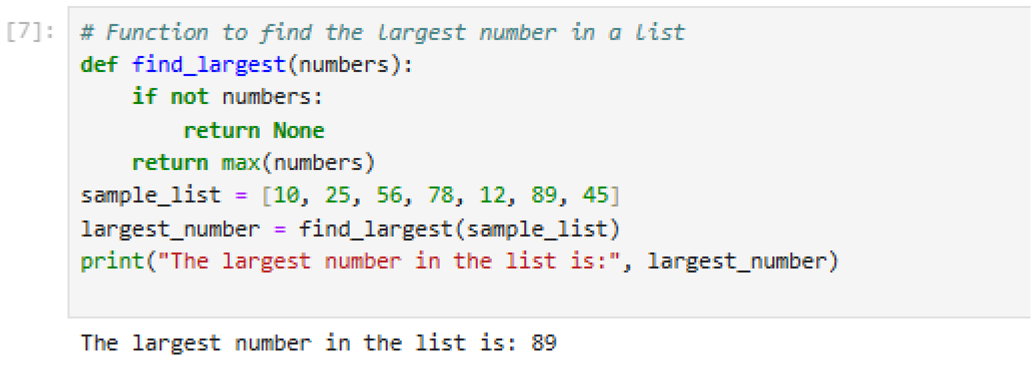
10.Create two sets of integers. Perform the given set operations.



```
[6]: # Creating two sets of integers
     set1 = {1, 2, 3, 4, 5}
     set2 = {4, 5, 6, 7, 8}
     union_set = set1.union(set2)
     print("Union of set1 and set2:", union_set)
     intersection_set = set1.intersection(set2)
     print("Intersection of set1 and set2:", intersection_set)
     difference_set1 = set1.difference(set2)
     print("Difference (set1 - set2):", difference_set1)
     #  Check if one set is a subset of the other
     is_subset = set1.issubset(set2)
     print("Is set1 a subset of set2?", is_subset)
     #  Add an element to a set
     set1.add(10)
     print("After adding 10 to set1:", set1)
```

```
Union of set1 and set2: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of set1 and set2: {4, 5}
Difference (set1 - set2): {1, 2, 3}
Is set1 a subset of set2? False
After adding 10 to set1: {1, 2, 3, 4, 5, 10}
```

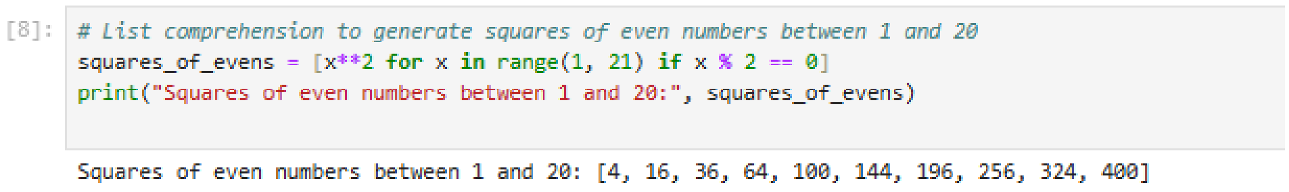
11.Write a Python function called find_largest() that takes a list of numbers as input and returns the largest number from the list. Test the function with a sample list.



[7]:
```
# Function to find the largest number in a list
def find_largest(numbers):
    if not numbers:
        return None
    return max(numbers)
sample_list = [10, 25, 56, 78, 12, 89, 45]
largest_number = find_largest(sample_list)
print("The largest number in the list is:", largest_number)
```

```
The largest number in the list is: 89
```

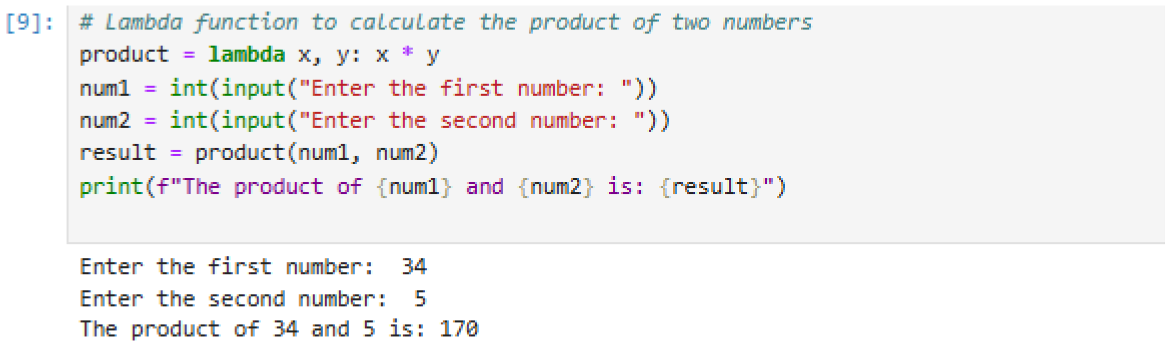
12.Use list comprehension to create a list of squares of all even numbers between 1 and 20.



[8]:
```
# List comprehension to generate squares of even numbers between 1 and 20
squares_of_evens = [x**2 for x in range(1, 21) if x % 2 == 0]
print("Squares of even numbers between 1 and 20:", squares_of_evens)
```

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

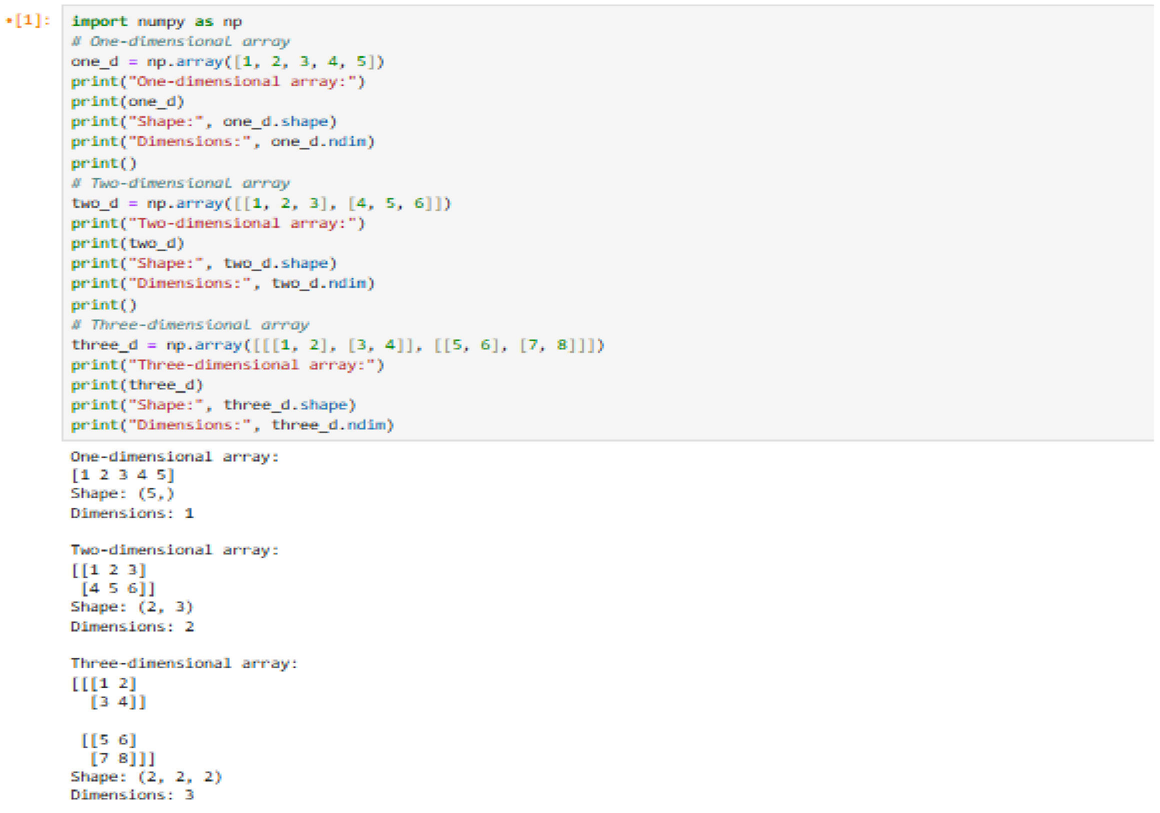
13.Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user



[9]:
```
# Lambda function to calculate the product of two numbers
product = lambda x, y: x * y
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
result = product(num1, num2)
print(f"The product of {num1} and {num2} is: {result}")
```

```
Enter the first number:  34
Enter the second number:  5
The product of 34 and 5 is: 170
```

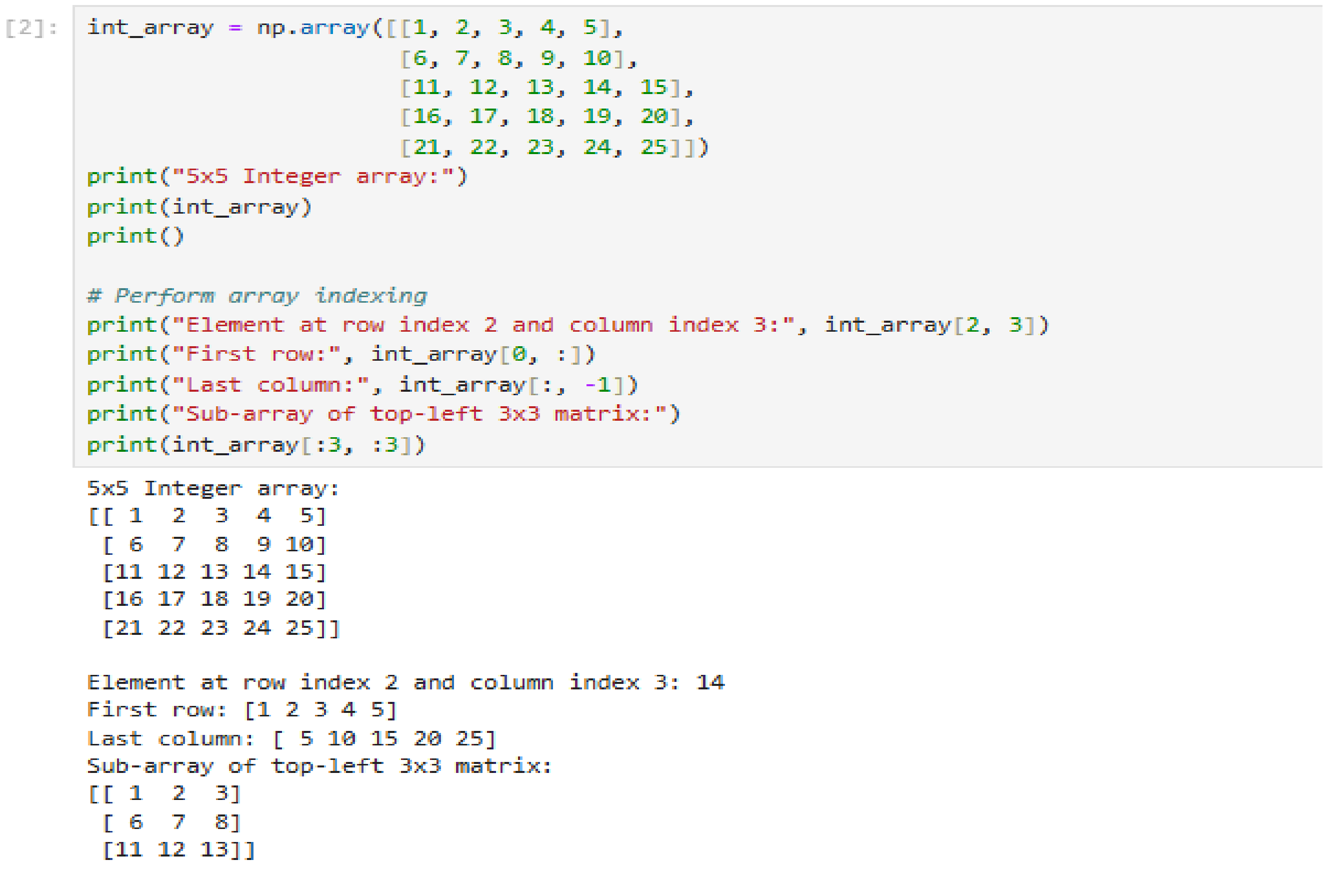
14.Write a Python program to create a one-dimensional, two-dimensional, and three-dimensional NumPy array. Print the shape and dimensions of each array.



```
•[1]: import numpy as np
      # One-dimensional array
      one_d = np.array([1, 2, 3, 4, 5])
      print("One-dimensional array:")
      print(one_d)
      print("Shape:", one_d.shape)
      print("Dimensions:", one_d.ndim)
      print()
      # Two-dimensional array
      two_d = np.array([[1, 2, 3], [4, 5, 6]])
      print("Two-dimensional array:")
      print(two_d)
      print("Shape:", two_d.shape)
      print("Dimensions:", two_d.ndim)
      print()
      # Three-dimensional array
      three_d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
      print("Three-dimensional array:")
      print(three_d)
      print("Shape:", three_d.shape)
      print("Dimensions:", three_d.ndim)
```

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

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

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

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

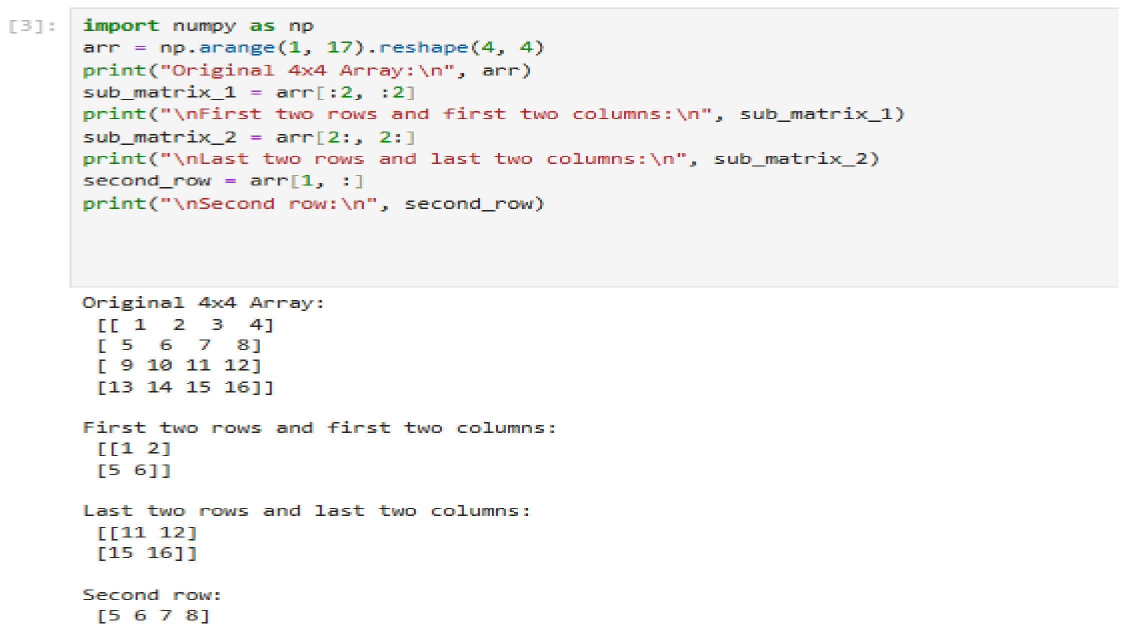
15.Write a Python program to create a 5x5 NumPy array of random integers and Perform array indexing as given.



```
[2]: int_array = np.array([[1, 2, 3, 4, 5],
                           [6, 7, 8, 9, 10],
                           [11, 12, 13, 14, 15],
                           [16, 17, 18, 19, 20],
                           [21, 22, 23, 24, 25]])
     print("5x5 Integer array:")
     print(int_array)
     print()

     # Perform array indexing
     print("Element at row index 2 and column index 3:", int_array[2, 3])
     print("First row:", int_array[0, :])
     print("Last column:", int_array[:, -1])
     print("Sub-array of top-left 3x3 matrix:")
     print(int_array[:3, :3])
```

```
5x5 Integer array:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]

Element at row index 2 and column index 3: 14
First row: [1 2 3 4 5]
Last column: [ 5 10 15 20 25]
Sub-array of top-left 3x3 matrix:
[[ 1  2  3]
 [ 6  7  8]
 [11 12 13]]
```

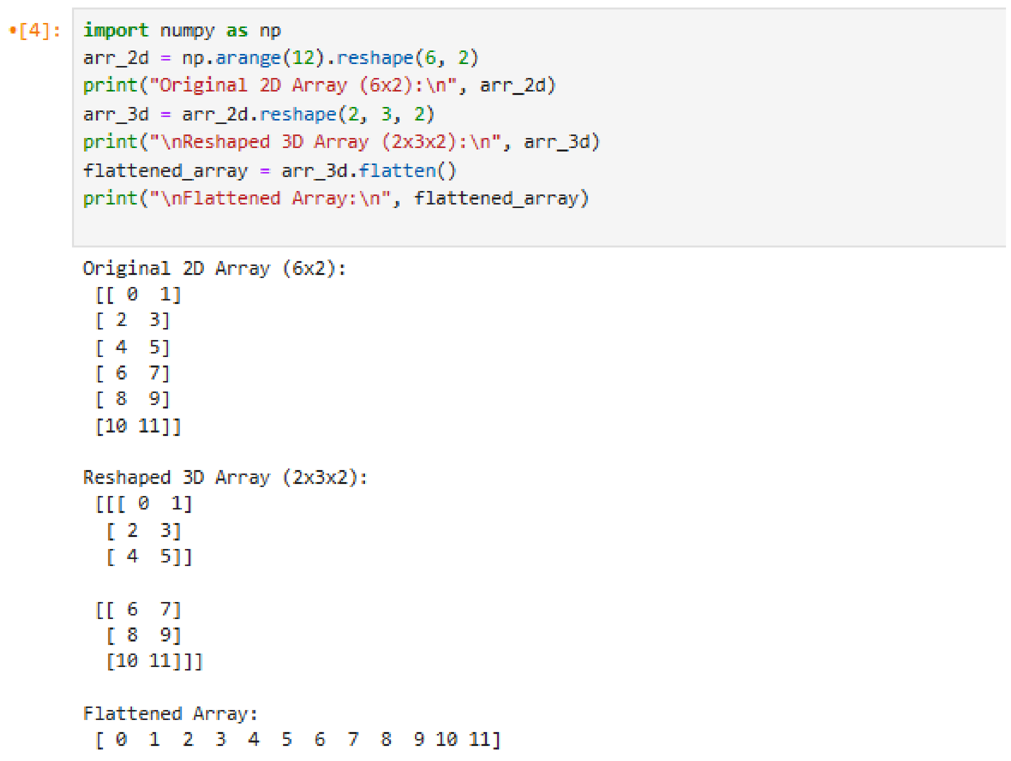
16.create a NumPy array of shape (4, 4) containing numbers from 1 to 16. Use slicing to extract for the given conditions



```
[3]: import numpy as np
     arr = np.arange(1, 17).reshape(4, 4)
     print("Original 4x4 Array:\n", arr)
     sub_matrix_1 = arr[:2, :2]
     print("\nFirst two rows and first two columns:\n", sub_matrix_1)
     sub_matrix_2 = arr[2:, 2:]
     print("\nLast two rows and last two columns:\n", sub_matrix_2)
     second_row = arr[1, :]
     print("\nSecond row:\n", second_row)
```

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

First two rows and first two columns:
 [[1 2]
 [5 6]]

Last two rows and last two columns:
 [[11 12]
 [15 16]]

Second row:
 [5 6 7 8]
```

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.



```
•[4]: import numpy as np
      arr_2d = np.arange(12).reshape(6, 2)
      print("Original 2D Array (6x2):\n", arr_2d)
      arr_3d = arr_2d.reshape(2, 3, 2)
      print("\nReshaped 3D Array (2x3x2):\n", arr_3d)
      flattened_array = arr_3d.flatten()
      print("\nFlattened Array:\n", flattened_array)
```

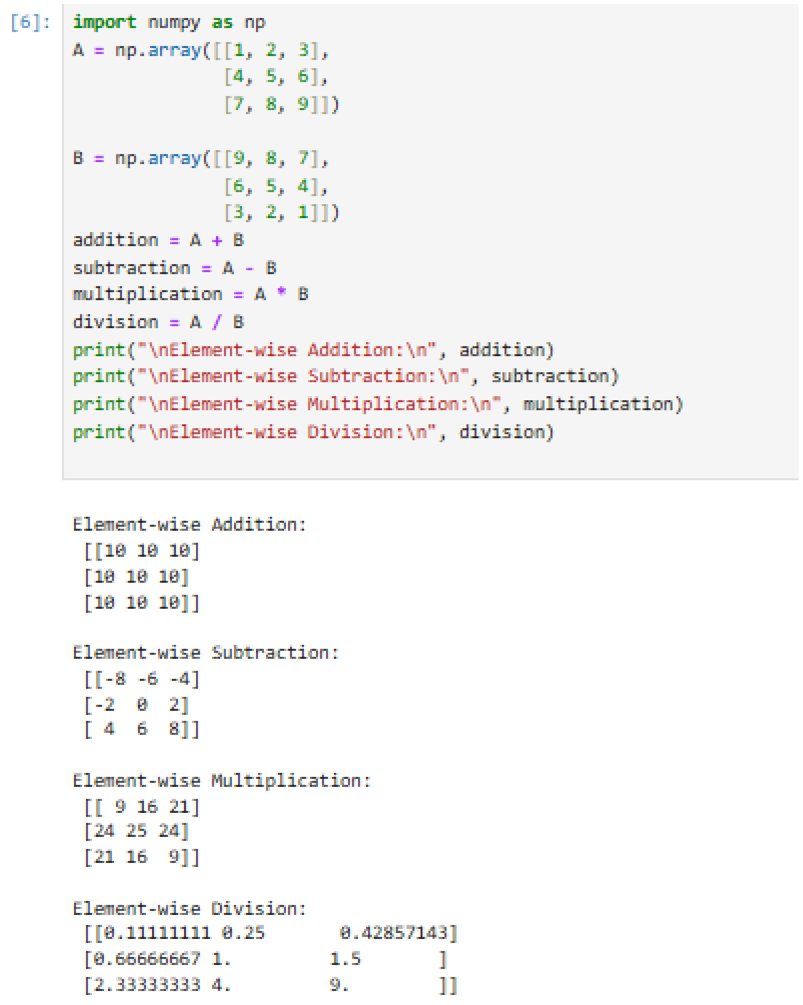
```
Original 2D Array (6x2):
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]

Reshaped 3D Array (2x3x2):
 [[[ 0  1]
  [ 2  3]
  [ 4  5]]

 [[ 6  7]
  [ 8  9]
  [10 11]]]

Flattened Array:
 [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

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

```
[5]: import numpy as np
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
vector = np.array([10, 20, 30])
result = matrix + vector
print("Original 3x3 Matrix:\n", matrix)
print("\n1D Array (1x3):\n", vector)
print("\nResult after Broadcasting Addition:\n", result)
```

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

1D Array (1x3):
 [10 20 30]

Result after Broadcasting Addition:
 [[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.



```
[6]: import numpy as np
     A = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])

     B = np.array([[9, 8, 7],
                   [6, 5, 4],
                   [3, 2, 1]])
     addition = A + B
     subtraction = A - B
     multiplication = A * B
     division = A / B
     print("\nElement-wise Addition:\n", addition)
     print("\nElement-wise Subtraction:\n", subtraction)
     print("\nElement-wise Multiplication:\n", multiplication)
     print("\nElement-wise Division:\n", division)
```

```
Element-wise Addition:
 [[10 10 10]
 [10 10 10]
 [10 10 10]]

Element-wise Subtraction:
 [[-8 -6 -4]
 [-2  0  2]
 [ 4  6  8]]

Element-wise Multiplication:
 [[ 9 16 21]
 [24 25 24]
 [21 16  9]]

Element-wise Division:
 [[0.11111111 0.25       0.42857143]
 [0.66666667 1.         1.5       ]
 [2.33333333 4.         9.        ]]
```

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.

[7]:

```python
import pandas as pd
data = {
    'Name': ['Sona', 'Teja', 'Veda'],
    'Math': [85, 78, 92],
    'Science': [90, 88, 80],
    'English': [75, 82, 89]
}

df = pd.DataFrame(data)
df['Total'] = df[['Math', 'Science', 'English']].sum(axis=1)
def assign_grade(total):
    if total >= 250:
        return 'A'
    elif total >= 220:
        return 'B'
    elif total >= 190:
        return 'C'
    else:
        return 'D'

df['Grade'] = df['Total'].apply(assign_grade)
print(df)
```

```
   Name  Math  Science  English  Total Grade
0  Sona    85       90       75    250     A
1  Teja    78       88       82    248     B
2  Veda    92       80       89    261     A
```