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

```
-------------------------------------------------
           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]

•[12]:
```python
# 2.
import pandas as pd
import numpy as np
data = [15, 21, 29, 21, 15, 24, 32, 21, 15, 30]
mean = np.mean(data)
med = np.median(data)
mode = pd.Series(data).mode()[0]
var = np.var(data)
std = np.std(data)
print("Mean:", mean, "\nMedian:", med, "\nMode:", mode, "\nVarience:", var, "\nStandard deviation:", std)
```

```
Mean: 22.3
Median: 21.0
Mode: 15
Varience: 36.61
Standard deviation: 6.050619802962338
```

3. You are analysing 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

```
•[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 data types (e.g., string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.

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).

```
: # 4.
name = "Anaya"
age = 5
height = 15.9
is_active = True
print(f"Name: {name}, Age: {age}, Height: {height}m, Active: {is_active}")
```

```
Name: Anaya, Age: 5, Height: 15.9m, Active: True
```

```
: # 5.
a = int(input("Enter a number: "))
if a > 0:
    print(a, "is a positive number.")
elif a < 0:
    print(a, "is a negative number.")
else:
    print(a, "is zero")
```

```
Enter a number:  -13
-13 is a negative number.
```

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

[12]:
```
a = input("Enter a number: ")
for i in range(1,11):
    print(a, " x ", i," = " ,a*i)
```

```
Enter a number:  5
5  x  1  =  5
5  x  2  =  55
5  x  3  =  555
5  x  4  =  5555
5  x  5  =  55555
5  x  6  =  555555
5  x  7  =  5555555
5  x  8  =  55555555
5  x  9  =  555555555
5  x  10  =  5555555555
```

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

In [18]:
```
# 7.
fruits = ["Apple", "Banana", "Cherry", "Mango", "Orange"]
# 1. Accessing the first fruit
print("First fruit:", fruits[0])
# 2. Appending a new fruit to the list
fruits.append("Grapes")
print("List after appending a fruit:", fruits)
# 3. Removing a fruit from the list
fruits.remove("Banana")
print("List after removing Banana:", fruits)
# 4. Sorting the list in alphabetical order
fruits.sort()
print("List after sorting alphabetically:", fruits)
# 5. Length of the list
print("Length of the list:", len(fruits))
```

```
First fruit: Apple
List after appending a fruit: ['Apple', 'Banana', 'Cherry', 'Mango', 'Orange', 'Grapes']
List after removing Banana: ['Apple', 'Cherry', 'Mango', 'Orange', 'Grapes']
List after sorting alphabetically: ['Apple', 'Cherry', 'Grapes', 'Mango', 'Orange']
Length of the list: 5
```

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

In [22]:

```
# 8.
# Creating a tuple with 5 numbers
numbers = (10, 20, 30, 40, 50)
# 1. Accessing an element in the tuple
print("First number:", numbers[0])
# 2. Slicing the tuple (getting the first 3 numbers)
print("First three numbers:", numbers[:3])
# 3. Getting the length of the tuple
print("Length of the tuple:", len(numbers))
# 4. Counting the occurrences of a number in the tuple
count_of_20 = numbers.count(20)
print("Count of 20 in the tuple:", count_of_20)
# 5. Finding the index of a number in the tuple
index_of_40 = numbers.index(40)
print("Index of 40 in the tuple:", index_of_40)
# 6. Converting the tuple to a list
numbers_list = list(numbers)
print("Tuple converted to list:", numbers_list)
```

```
First number: 10
First three numbers: (10, 20, 30)
Length of the tuple: 5
Count of 20 in the tuple: 1
Index of 40 in the tuple: 3
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.

20]:

```
# 9. Creating a dictionary with 3 students and their marks in mathematics
students_marks = {"Alice": 85,"Bob": 90,"Charlie": 78}
# 1. Accessing the marks of a specific student
print("Marks of Alice:", students_marks["Alice"])
# 2. Adding a new student and their marks to the dictionary
students_marks["David"] = 92
print("Dictionary after adding a new student:", students_marks)
# 3. Updating the marks of an existing student
students_marks["Bob"] = 95
print("Dictionary after updating Bob's marks:", students_marks)
# 4. Removing a student from the dictionary
del students_marks["Charlie"]
print("Dictionary after removing Charlie:", students_marks)
# 5. Getting the total number of students
print("Total number of students:", len(students_marks))
# 6. Iterating over the dictionary to print all students and their marks
for student, marks in students_marks.items():
    print(f"{student}: {marks}")
```

```
Marks of Alice: 85
Dictionary after adding a new student: {'Alice': 85, 'Bob': 90, 'Charlie': 78, 'David': 92}
Dictionary after updating Bob's marks: {'Alice': 85, 'Bob': 95, 'Charlie': 78, 'David': 92}
Dictionary after removing Charlie: {'Alice': 85, 'Bob': 95, 'David': 92}
Total number of students: 3
Alice: 85
Bob: 95
David: 92
```

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

[24]:
```
# 10.
# Creating two sets of integers
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
# 1. Union of the sets (all elements from both sets)
u = set1 | set2
print("Union of set1 and set2:", u)
# 2. Intersection of the sets (common elements between both sets)
i = set1 & set2
print("Intersection of set1 and set2:", i)
# 3. Difference of the sets (elements in set1 but not in set2)
d = set1 - set2
print("Difference of set1 and set2:", d)
```

```
Union of set1 and set2: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of set1 and set2: {4, 5}
Difference of set1 and set2: {1, 2, 3}
```

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.
12. Use list comprehension to create a list of squares of all even numbers between 1 and 20.
13. Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user.

```
# 11.
def find_largest(numbers):
    if not numbers:
        return None
    return max(numbers)
list = [12, 45, 78, 34, 89, 23]
largest_num = find_largest(list)
print("The largest number is: ",largest_num)
```

```
The largest number is:  89
```

```
# 12.
even_squares = [x**2 for x in range(1, 21) if x % 2 == 0]
print(even_squares)
```

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

```
# 13.
product = lambda x, y: x * y
x, y = map(int, input("Enter two numbers: ").split())
print(f"Product: {product(x, y)}")
```

```
Enter two numbers:  5 2
Product: 10
```

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])
print("One-dimensional array:")
print(one_d_array)
print("Shape:", one_d_array.shape)
print("Number of dimensions:", one_d_array.ndim)
print()
two_d_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Two-dimensional array:")
print(two_d_array)
print("Shape:", two_d_array.shape)
print("Number of dimensions:", two_d_array.ndim)
print()
three_d_array = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("Three-dimensional array:")
print(three_d_array)
print("Shape:", three_d_array.shape)
print("Number of dimensions:", three_d_array.ndim)
```

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

Two-dimensional array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)
Number of dimensions: 2

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

 [[5 6]
  [7 8]]]
Shape: (2, 2, 2)
Number of dimensions: 3

=== Code Execution Successful ===
```

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
random_array = np.random.randint(0, 10, (5, 5))
print("Random 5x5 Array:")
print(random_array)
print()
element = random_array[1, 2]
print(f"Element at row 2, column 3: {element}")
second_row = random_array[1, :]
print("Second row:", second_row)
third_column = random_array[:, 2]
print("Third column:", third_column)
subarray = random_array[0:2, 1:3]
print("Subarray (rows 1 and 2, columns 2 and 3):")
print(subarray)
```

```
Random 5x5 Array:
[[6 2 9 8 4]
 [4 2 1 1 7]
 [4 0 7 2 8]
 [9 1 8 5 5]
 [0 2 4 1 9]]

Element at row 2, column 3: 1
Second row: [4 2 1 1 7]
Third column: [9 1 7 8 4]
Subarray (rows 1 and 2, columns 2 and 3):
[[2 9]
 [2 1]]

=== Code Execution Successful ===
```

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:
[[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)
```

```
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
```