Alliance School of Advanced Computing

Department of Computer Science and Engineering

Class Assignment-1

Course Code: 5CS1025
Course Title: Artificial Intelligence
Semester: 04
Name: Kamili Reddy Rushitha
Registration number:2023BCSE07AED313
Class : AIML-D

2024-25

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

Technologies	Tools	Frameworks
Talking (Speech Recognition and Speech)	High Quality Microphones and Speakers	Text-to-Speech, Speech Recognition
Listening (Natural Language Processing)	NLTK, Spacy	OpenAI GPT, Google Dialog flow
Walking (Robotics Hardware)	Servomotors for joint movements, DC motors (Wheels and Walking)	ROS, Gazebo
Seeing / Recognizing Objects (Computer Vision)	OpenCV	Tensor Flow
Learning (Machine Learning)	Jupyter Notebook, Artificial Intelligence	PyTorch, Keras
Movement (Robotics Hardware)	Raspberry pi, Arduino	Robot Operating System

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]

[2]:
```
import numpy as np
import statistics as stats
data = [15, 21, 29, 21, 15, 24, 32, 21, 15, 30]
mean_value = np.mean(data)
median_value = np.median(data)
mode_value = stats.mode(data)
variance_value = np.var(data, ddof=0)
std_dev_value = np.std(data, ddof=0)
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
print(f"Variance: {variance_value}")
print(f"Standard Deviation: {std_dev_value}")
```

```
Mean: 22.3
Median: 21.0
Mode: 15
Variance: 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

Interaction_Count: Number of conversations the robot had daily.
Steps_Walked: Total steps taken each day.
Objects_Recognized: Number of objects successfully identified by the robot.
Learning_Sessions: Number of learning tasks completed.
Energy_Consumption (kWh): Daily energy usage of robots.

Perform Basic Statistical Operations:

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).
6) What is the **variance** in the number of learning sessions completed?

Ans. Code and Output

```
import pandas as pd
df = pd.read_excel("robot_dataset(robot_dataset)_1.xlsx")
print(df.drop_duplicates().head(5))
```

```
  Robot_ID   Task_Type Component_ID Sensor_Type             Sensor_Data  \
0  RBT_001  Inspection       CMP_460       LIDAR  1 (obstacle detected)
1  RBT_002    Assembly       CMP_252     Thermal             85.3 (Â°C)
2  RBT_003  Inspection       CMP_248     Thermal        92% (visual fit)
3  RBT_004     Welding       CMP_433      Camera       98% (defect-free)
4  RBT_005    Assembly       CMP_992      Camera        92% (visual fit)

   Processing_Time (s)  Accuracy (%) Environmental_Status  \
0                 67.0          90.4                Stable
1                 71.2          98.1                Stable
2                 49.2          95.3              Unstable
3                 74.5          90.2                Stable
4                 64.5          97.2              Unstable

   Energy_Consumption (kWh) Human_Intervention_Needed Obstacle_Detected  \
0                        2.2                       No                Yes
1                        2.7                      Yes                 No
2                        2.4                       No                 No
3                        2.4                      Yes                 No
4                        1.8                       No                 No

  Defect_Detected  Interaction_Count  Steps_Walked  Objects_Recognized  \
0             Yes                  2            10                   2
1              No                  6            23                   4
2              No                  2            25                   3
3             Yes                  7            34                   5
4              No                  8            35                   3

   Learning_Sessions  Energy_Consumption (kWh).1
0                 10                         100
1                 12                         123
2                 45                         321
3                 12                         456
```

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]:
```
name = "Rushitha"
age = 19
height = 5.6
is_student = True


print(f"My name is {name}. I am {age} years old. My height is {height} feet. Am I a student? {is_student}.")
```

```
My name is Rushitha. I am 19 years old. My height is 5.6 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).

```python
def check_number(num):
    if num > 0:
        return f"The number {num} is positive."
    elif num < 0:
        return f"The number {num} is negative."
    else:
        return "The number is zero."
num = int(input("Enter an integer: "))
print(check_number(num))

```

```
Enter an integer: 1
The number 1 is positive.

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

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

```python

num = int(input("Enter a number: "))
print(f"Multiplication Table for {num}:")
for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")

```

```
Enter a number: 9
Multiplication Table for 9:
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

=== Code Execution Successful =
```

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

```python

fruits = ["Pineapple", "Strawberry", "Watermelon", "Blueberry",
    "Kiwi"]
print("Original List of Fruits:", fruits)
fruits.append("Peach")
print("After Adding a Fruit:", fruits)
fruits.remove("Watermelon")
print("After Removing 'Watermelon':", fruits)
fruits.sort()
print("Sorted List:", fruits)
fruits.reverse()
print("Reversed List:", fruits)
print("Fruit at index 2:", fruits[2])

```

```
Original List of Fruits: ['Pineapple', 'Strawberry', 'Watermelon',
    'Blueberry', 'Kiwi']
After Adding a Fruit: ['Pineapple', 'Strawberry', 'Watermelon', 'Blueberry',
    'Kiwi', 'Peach']
After Removing 'Watermelon': ['Pineapple', 'Strawberry', 'Blueberry', 'Kiwi'
    , 'Peach']
Sorted List: ['Blueberry', 'Kiwi', 'Peach', 'Pineapple', 'Strawberry']
Reversed List: ['Strawberry', 'Pineapple', 'Peach', 'Kiwi', 'Blueberry']
Fruit at index 2: Peach

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

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

```python
fruits = ["Pineapple", "Strawberry", "Watermelon", "Blueberry",
    "Kiwi"]
print("Original List of Fruits:", fruits)
fruits.append("Peach")
print("After Adding a Fruit:", fruits)
fruits.remove("Watermelon")
print("After Removing 'Watermelon':", fruits)
fruits.sort()
print("Sorted List:", fruits)
fruits.reverse()
print("Reversed List:", fruits)
print("Fruit at index 2:", fruits[2])
```

```
Original List of Fruits: ['Pineapple', 'Strawberry', 'Watermelon',
    'Blueberry', 'Kiwi']
After Adding a Fruit: ['Pineapple', 'Strawberry', 'Watermelon', 'Blueberry',
    'Kiwi', 'Peach']
After Removing 'Watermelon': ['Pineapple', 'Strawberry', 'Blueberry', 'Kiwi'
    , 'Peach']
Sorted List: ['Blueberry', 'Kiwi', 'Peach', 'Pineapple', 'Strawberry']
Reversed List: ['Strawberry', 'Pineapple', 'Peach', 'Kiwi', 'Blueberry']
Fruit at index 2: Peach

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

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

```python
students_marks = {
    "Rushitha": 85,
    "Karthik": 90,
    "Kara": 78
}
print("Student Marks Dictionary:", students_marks)
print("\nMarks of Karthik:", students_marks["Karthik"])
students_marks["Kara"] = 82
print("\nUpdated Marks of Kara:", students_marks)
students_marks["Yashwanth"] = 88
print("\nDictionary after adding Yashwanth:", students_marks)
del students_marks["Rushitha"]
print("\nDictionary after removing Rushitha:", students_marks)
```

```
Student Marks Dictionary: {'Rushitha': 85, 'Karthik': 90, 'Kara': 78}

Marks of Karthik: 90

Updated Marks of Kara: {'Rushitha': 85, 'Karthik': 90, 'Kara': 82}

Dictionary after adding Yashwanth: {'Rushitha': 85, 'Karthik': 90, 'Kara':
    82, 'Yashwanth': 88}

Dictionary after removing Rushitha: {'Karthik': 90, 'Kara': 82, 'Yashwanth':
    88}

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

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

```python
set_A = {1, 2, 3, 4, 5}
set_B = {4, 5, 6, 7, 8}
print("Set A:", set_A)
print("Set B:", set_B)
union_set = set_A.union(set_B)
print("Union of A and B:", union_set)
intersection_set = set_A.intersection(set_B)
print("Intersection of A and B:", intersection_set)
difference_A_B = set_A.difference(set_B)
print("Difference (A - B):", difference_A_B)
difference_B_A = set_B.difference(set_A)
print("Difference (B - A):", difference_B_A)
symmetric_difference_set = set_A.symmetric_difference(set_B)
print("Symmetric Difference of A and B:", symmetric_difference_set)
print("Is A a subset of B?", set_A.issubset(set_B))
print("Is B a superset of A?", set_B.issuperset(set_A))
```

```
Set A: {1, 2, 3, 4, 5}
Set B: {4, 5, 6, 7, 8}
Union of A and B: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of A and B: {4, 5}
Difference (A - B): {1, 2, 3}
Difference (B - A): {8, 6, 7}
Symmetric Difference of A and B: {1, 2, 3, 6, 7, 8}
Is A a subset of B? False
Is B a superset of A? False

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

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.

```

set_A = {1, 2, 3, 4, 5}
set_B = {4, 5, 6, 7, 8}
print("Set A:", set_A)
print("Set B:", set_B)
union_set = set_A.union(set_B)
print("Union of A and B:", union_set)
intersection_set = set_A.intersection(set_B)
print("Intersection of A and B:", intersection_set)
difference_A_B = set_A.difference(set_B)
print("Difference (A - B):", difference_A_B)
difference_B_A = set_B.difference(set_A)
print("Difference (B - A):", difference_B_A)
symmetric_difference_set = set_A.symmetric_difference(set_B)
print("Symmetric Difference of A and B:", symmetric_difference_set)
print("Is A a subset of B?", set_A.issubset(set_B))
print("Is B a superset of A?", set_B.issuperset(set_A))

```

```
Set A: {1, 2, 3, 4, 5}
Set B: {4, 5, 6, 7, 8}
Union of A and B: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of A and B: {4, 5}
Difference (A - B): {1, 2, 3}
Difference (B - A): {8, 6, 7}
Symmetric Difference of A and B: {1, 2, 3, 6, 7, 8}
Is A a subset of B? False
Is B a superset of A? False

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

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

```
squares_of_even = [x**2 for x in range(1, 21) if x % 2 == 0]
print("Squares of even numbers between 1 and 20:", squares_of_even)

```

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

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

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

```
product = lambda x, y: x * y
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
result = product(num1, num2)
print(f"The product of {num1} and {num2} is: {result}")
```

```
Enter first number: 4
Enter second number: 9
The product of 4.0 and 9.0 is: 36.0

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

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 = np.array([10, 20, 30, 40, 50])
print("1D Array:\n", one_d)
print("Shape of 1D Array:", one_d.shape)
print("Dimensions of 1D Array:", one_d.ndim)
print()
two_d = np.array([[11, 22, 33], [44, 55, 66]])
print("2D Array:\n", two_d)
print("Shape of 2D Array:", two_d.shape)
print("Dimensions of 2D Array:", two_d.ndim)
print()
three_d = np.array([[[101, 102], [103, 104]], [[201, 202], [203, 204]]])
print("3D Array:\n", three_d)
print("Shape of 3D Array:", three_d.shape)
print("Dimensions of 3D Array:", three_d.ndim)
```

```
1D Array:
 [10 20 30 40 50]
Shape of 1D Array: (5,)
Dimensions of 1D Array: 1

2D Array:
 [[11 22 33]
 [44 55 66]]
Shape of 2D Array: (2, 3)
Dimensions of 2D Array: 2

3D Array:
 [[[101 102]
  [103 104]]

 [[201 202]
  [203 204]]]
Shape of 3D Array: (2, 2, 2)
Dimensions of 3D Array: 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
array_5x5 = np.random.randint(1, 100, (5, 5))
print("5x5 Random Integer Array:\n", array_5x5)
print("\nFirst row:", array_5x5[0])
print("Last row:", array_5x5[-1])
print("First column:", array_5x5[:, 0])
print("Last column:", array_5x5[:, -1])
print("Element at (2,3):", array_5x5[2, 3])
print("Sub-array (2x2 from top-left):\n", array_5x5[:2, :2])
```

```
5x5 Random Integer Array:
 [[72 42 92 82 28]
 [57 41 73 82  9]
 [38 11 67 34 15]
 [83 56 78 23 31]
 [89 82 35 41 51]]

First row: [72 42 92 82 28]
Last row: [89 82 35 41 51]
First column: [72 57 38 83 89]
Last column: [28  9 15 31 51]
Element at (2,3): 34
Sub-array (2x2 from top-left):
 [[72 42]
 [57 41]]

=== 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

```python
import numpy as np
array_4x4 = np.array([[5, 10, 15, 20],
                      [25, 30, 35, 40],
                      [45, 50, 55, 60],
                      [65, 70, 75, 80]])
print("4x4 Array:\n", array_4x4)
print("\nFirst two rows:\n", array_4x4[:2, :])
print("\nLast two columns:\n", array_4x4[:, -2:])
print("\nCenter 2x2 Sub-array:\n", array_4x4[1:3, 1:3])
print("\nDiagonal elements:", np.diag(array_4x4))
```

```
4x4 Array:
 [[ 5 10 15 20]
 [25 30 35 40]
 [45 50 55 60]
 [65 70 75 80]]

First two rows:
 [[ 5 10 15 20]
 [25 30 35 40]]

Last two columns:
 [[15 20]
 [35 40]
 [55 60]
 [75 80]]

Center 2x2 Sub-array:
 [[30 35]
 [50 55]]

Diagonal elements: [ 5 30 55 80]

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

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.

```python
import numpy as np
array_2d = np.arange(1, 13).reshape(6, 2)
array_3d = array_2d.reshape(2, 3, 2)
flattened_array = array_3d.flatten()
print("2D Array (6x2):\n", array_2d)
print("\nReshaped 3D Array (2x3x2):\n", array_3d)
print("\nFlattened Array:\n", flattened_array)

```

```
2D Array (6x2):
 [[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]

Reshaped 3D Array (2x3x2):
 [[[ 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]

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

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.

[2]:
```python
import numpy as np

array_3x3 = np.array([[2, 4, 6],
                      [8, 10, 12],
                      [14, 16, 18]])

array_1x3 = np.array([1, 2, 3])

result = array_3x3 + array_1x3

print("3x3 Array:\n", array_3x3)
print("\n1x3 Array:\n", array_1x3)
print("\nResult after Broadcasting:\n", result)
```

```
3x3 Array:
 [[ 2  4  6]
 [ 8 10 12]
 [14 16 18]]

1x3 Array:
 [1 2 3]

Result after Broadcasting:
 [[ 3  6  9]
 [ 9 12 15]
 [15 18 21]]
```

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.

```python
import numpy as np

A = np.array([[5, 15, 25],
              [35, 45, 55],
              [65, 75, 85]])

B = np.array([[2, 4, 6],
              [8, 10, 12],
              [14, 16, 18]])

print("Element-wise Addition:\n", A + B)
print("\nElement-wise Subtraction:\n", A - B)
print("\nElement-wise Multiplication:\n", A * B)
print("\nElement-wise Division:\n", A / B)
```

```
Element-wise Addition:
 [[  7  19  31]
 [ 43  55  67]
 [ 79  91 103]]

Element-wise Subtraction:
 [[ 3 11 19]
 [27 35 43]
 [51 59 67]]

Element-wise Multiplication:
 [[  10   60  150]
 [ 280  450  660]
 [ 910 1200 1530]]

Element-wise Division:
 [[2.5        3.75       4.16666667]
 [4.375      4.5        4.58333333]
 [4.64285714 4.6875     4.72222222]]
```

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.

[1]:
```
students_marks = {
    "Rushitha": 85,
    "Karthik": 90,
    "Kara": 78
}

print("Student Marks Dictionary:", students_marks)

print("\nMarks of Karthik:", students_marks["Karthik"])

students_marks["Kara"] = 82
print("\nUpdated Marks of Kara:", students_marks)

students_marks["Yashwanth"] = 88
print("\nDictionary after adding Yashwanth:", students_marks)

del students_marks["Rushitha"]
print("\nDictionary after removing Rushitha:", students_marks)
```

```
Student Marks Dictionary: {'Rushitha': 85, 'Karthik': 90, 'Kara': 78}

 90ks of Karthik:

 {'Rushitha': 85, 'Karthik': 90, 'Kara': 82}

 {'Rushitha': 85, 'Karthik': 90, 'Kara': 82, 'Yashwanth': 88}

 {'Karthik': 90, 'Kara': 82, 'Yashwanth': 88}
```