NAME - UTKARSH YADAV

ROLL - 23053172

In [1]:
```
%%writefile sample.txt
Hello world.
This is a test file for Python programming.
Machine Learning is interesting.
10
20
30
40
50
```

```
Writing sample.txt
```

In [3]:
```
%%writefile data.csv
Age,Salary,Experience
25,50000,2
30,60000,5
35,75000,8
40,,10
45,90000,15
,55000,3
```

```
Writing data.csv
```

In [5]:
```
#q1 count number of lines in a text file

filename = "sample.txt"

with open(filename, 'r') as file:
    lines = file.readlines()
    print(f"Number of lines: {len(lines)}")
```

```
Number of lines: 8
```

In [7]:
```
#q2 count words in a file

filename = "sample.txt"
word_count = 0

with open(filename, 'r') as file:
    for line in file:
        words = line.split()
        word_count += len(words)

print(f"Total words: {word_count}")
```

```
Total words: 19
```

In [9]:
```
#q3 count characters in a file

filename = "sample.txt"
```

```
with open(filename, 'r') as file:
    content = file.read()
    print(content)
    print(f"Total characters: {len(content)}")
```

```
Hello world.
This is a test file for Python programming.
Machine Learning is interesting.
10
20
30
40
50

Total characters: 105
```

In [11]:
```
#q4 Write to a file and then read from it.
filename = "output.txt"

with open(filename, 'w') as file:
    file.write("This is a newly written line.\n")
    file.write("Writing to files is easy in Python.")

print("Reading from file: ")
with open(filename, 'r') as file:
    print(file.read())
```

```
Reading from file:
This is a newly written line.
Writing to files is easy in Python.
```

In [15]:
```
#q5 read a csv file and print column names

import csv

filename = "data.csv"

with open(filename, 'r') as file:
    reader = csv.reader(file)
    header = next(reader)
    print(f"Column names: {header}")
```

```
Column names: ['Age', 'Salary', 'Experience']
```

In [17]:
```
#q6 find average of numbers stored in a file

filename = "sample.txt"
numbers = []

with open(filename, 'r') as file:
    for line in file:
        line = line.strip()
        if line.isdigit():
            numbers.append(int(line))
print(f"Average: {sum(numbers) / len(numbers)}")
```

```
Average: 30.0
```

In [19]:
```python
#q7 print number of rows in csv

import pandas as pd

df = pd.read_csv("data.csv")
print(f"Number of rows: {df.shape[0]}")
```

```
Number of rows: 6
```

In [21]:
```python
#q8 print first 5 rows using pandas

df = pd.read_csv("data.csv")
print(df.head())
```

```
    Age   Salary  Experience
0  25.0  50000.0           2
1  30.0  60000.0           5
2  35.0  75000.0           8
3  40.0      NaN          10
4  45.0  90000.0          15
```

In [25]:
```python
#q9 Load a CSV using numpy and print shape.
import numpy as np

data = np.genfromtxt("data.csv", delimiter=",", skip_header = 1)
print(f"Shape of array: {data.shape}")
```

```
Shape of array: (6, 3)
```

In [29]:
```python
#q10 Replace missing values in a file with the mean value.

import pandas as pd

df = pd.read_csv("data.csv")
print("Original data with NaNs: \n", df)

df_filled = df.fillna(df.mean())
print("\nData after replacing missing values with Mean:\n", df_filled)
```

```
Original data with NaNs:
     Age   Salary  Experience
0  25.0  50000.0           2
1  30.0  60000.0           5
2  35.0  75000.0           8
3  40.0      NaN          10
4  45.0  90000.0          15
5   NaN  55000.0           3

Data after replacing missing values with Mean:
     Age   Salary  Experience
0  25.0  50000.0           2
1  30.0  60000.0           5
2  35.0  75000.0           8
3  40.0  66000.0          10
4  45.0  90000.0          15
5  35.0  55000.0           3
```

In [31]:
```python
#q11 Compute correlation between two lists.
```

```
import numpy as np

list_x = [1, 2, 3, 4, 5]
list_y = [2, 4, 6, 8, 10]


correlation_matrix = np.corrcoef(list_x, list_y)
correlation = correlation_matrix[0, 1]

print(f"Correlation coefficient: {correlation}")
```

```
Correlation coefficient: 0.9999999999999999
```

In [33]:
```
#q12 Normalize a list of numbers

data = [10, 20, 30, 40, 50]

min_val = min(data)
max_val = max(data)

normalized_data = [(x - min_val) / (max_val - min_val) for x in data]

print(f"Original: {data}")
print(f"Normalized: {normalized_data}")
```

```
Original: [10, 20, 30, 40, 50]
Normalized: [0.0, 0.25, 0.5, 0.75, 1.0]
```

In [35]:
```
#q13 Standardize a list of numbers manually.

import math

data = [10, 20, 30, 40, 50]

mean = sum(data) / len(data)
variance = sum((x - mean) ** 2 for x in data) / len(data)
std_dev = math.sqrt(variance)

standardized_data = [(x - mean) / std_dev for x in data]

print(f"Original: {data}")
print(f"Standardized: {standardized_data}")
```

```
Original: [10, 20, 30, 40, 50]
Standardized: [-1.414213562373095, -0.7071067811865475, 0.0, 0.7071067811865
475, 1.414213562373095]
```

In [42]:
```
#q14 find euclidean distance

import math

point1 = (1,2)
point2 = (4, 6)

distance = math.sqrt((point2[0] - point1[0])**2 + (point2[1] - point1[1])**
print(f"Euclidean Distance between {point1} and {point2} is: {distance}")
```

```
Euclidean Distance between (1, 2) and (4, 6) is: 5.0
```

In [44]:
```python
#q15 implement a simple kNN step: find nearest value from a list.

dataset = [10, 25, 38, 45, 60]
query_point = 40
nearest_neighbor = min(dataset, key=lambda x: abs(x - query_point))

print(f"Dataset: {dataset}")
print(f"Query Point: {query_point}")
print(f"Nearest Neighbor: {nearest_neighbor}")
```

```
Dataset: [10, 25, 38, 45, 60]
Query Point: 40
Nearest Neighbor: 38
```