MANIPAL INSTITUTE OF TECHNOLOGY
BENGALURU
(A constituent unit of MAHE, Manipal)

B.TECH. SECOND SEMESTER

COMMON TO 1ST YEAR

DATA VISUALIZATION LAB

CSS_1022

LABORATORY MANUAL

Course Coordinator: Dr. S. Priya
School of Computer Engineering, Manipal Institute of Technology, Bengaluru, India

Section	Faculty Instructor
B & E	Dr. Amreen Ayesha
D & P	Dr. Sindhu Madhuri
C & F	Dr. Sapna
R & I	Dr. SK Mahmudul Hassan
A & N	Mrs. Devi Sivasankari P
H & O	Dr. S. Priya
M & Q	Dr. Kshama
G & U	Mr. Anil
S & T	Ms. Spurthy Maria Pais

JANUARY 2026

Table of Contents

4. Software and Tools

Programming language: Python

IDE: Anaconda distribution with Jupyter Notebook

Pre-installed in Anaconda:

NumPy: For numerical computations.

Pandas: For data manipulation and

analysis. Matplotlib: For plotting and

visualization.

Additional Package: Seaborn: For advanced visualization

5. Introduction to Anaconda, Jupyter Notebook

5.1 Anaconda

Anaconda is a open-source distribution for Python and R, specifically for data science. It can be used to create an isolated environment for making data intensive applications. It simplifies the process of managing packages and dependencies. Anaconda has over 1,500 packages, including essential tools like Jupyter Notebook, Pandas, NumPy, and Matplotlib, making it a one-stop solution for developers and researchers. Its package manager, Conda, allows seamless installation, updating, and management of software, ensuring compatibility and reducing conflicts.

5.2 Jupyter Notebook

Jupyter is a freely available web application that enables creation and sharing of documents containing equations, live coding, visualizations, and narrative text. Jupyter provides an interactive computing environment and it supports multiple programming languages, including Python, R, Julia. The major components of the Jupyter project is the notebook, a type of interactive document for code, text (including Markdown), data visualizations, and other output. The Jupyter notebook interacts with kernels, which are implementations of the Jupyter interactive computing protocol specific to different programming languages. Jupyter integrates data science libraries and frameworks, such as NumPy, Pandas, Matplotlib, sci-kit-learn, TensorFlow, and PyTorch. This allows users to leverage the full power of these tools within the notebook environment for tasks like data manipulation, visualization, machine learning, and deep learning.

5.2. Getting started with Anaconda, Jupyter notebook on Windows

Step 1: Type Jupyter notebook in the search bar as shown in Figure 1 and click on "Jupyter Notebook (Anaconda3).

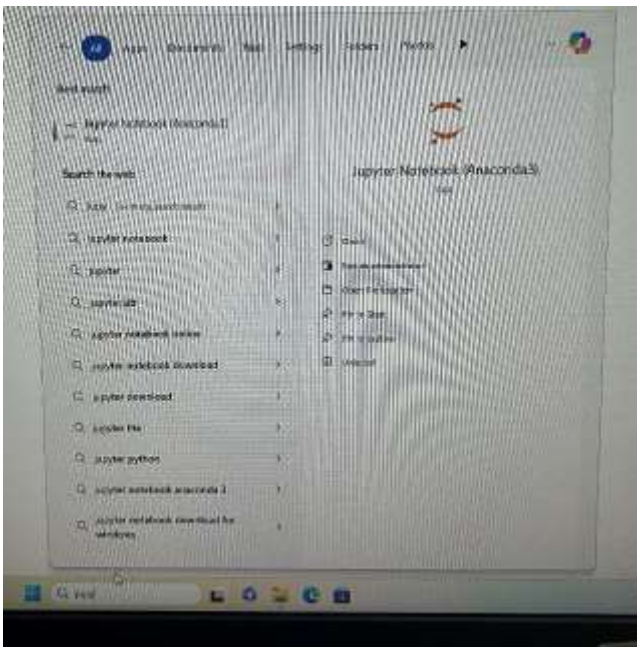


Fig 1: Searching Jupyter application from the search bar.

Step 2: Change drive if required. Creating a new folder. Click on New-> Folder as shown in Figure 2.

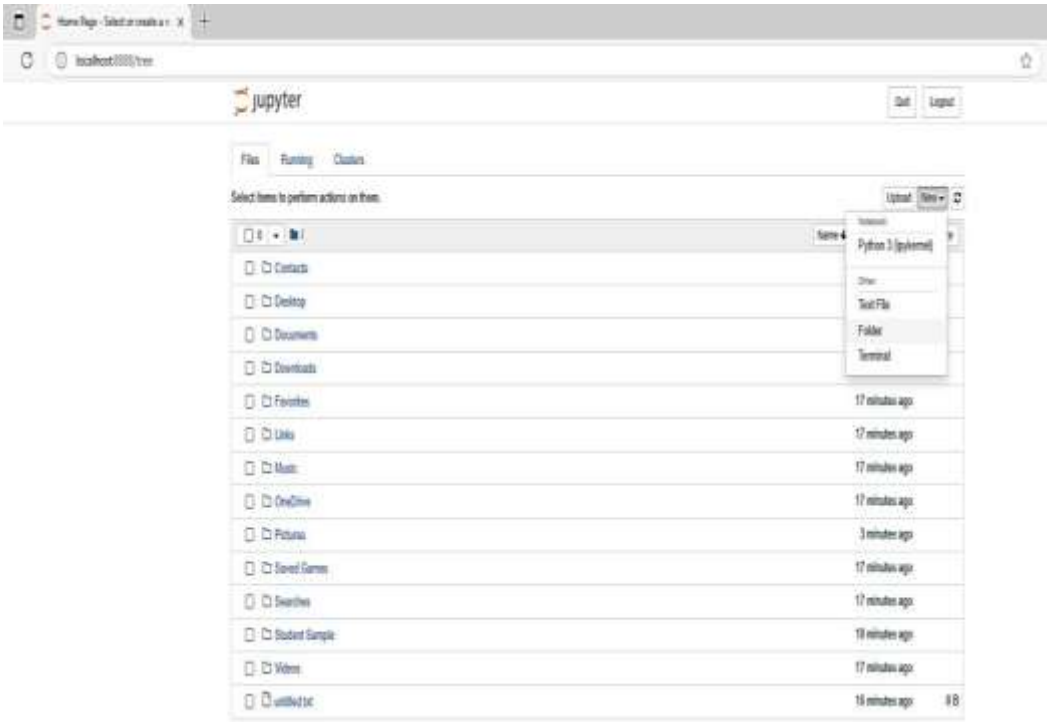


Fig 2: Creating a new folder in Jupyter.

Step 3: Renaming the new folder. The new folder will be created with the default name "Untitled folder". Click on rename and rename the folder appropriately as shown in Figure 3.

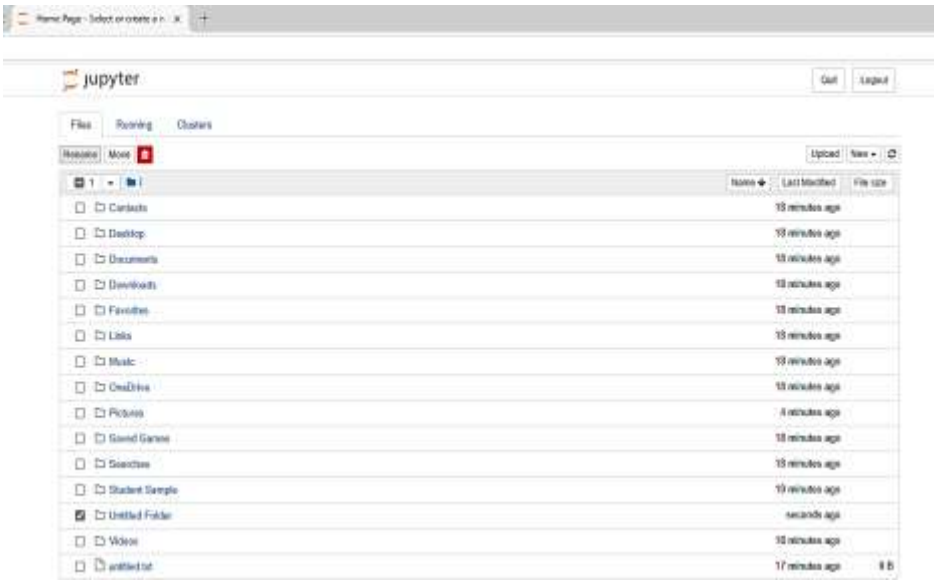


Fig 3: Renaming the new folder.

Step 4: Creating a new notebook inside the newly created folder. Click on New-> Python 3(ipykernel) as shown in Figure 4.

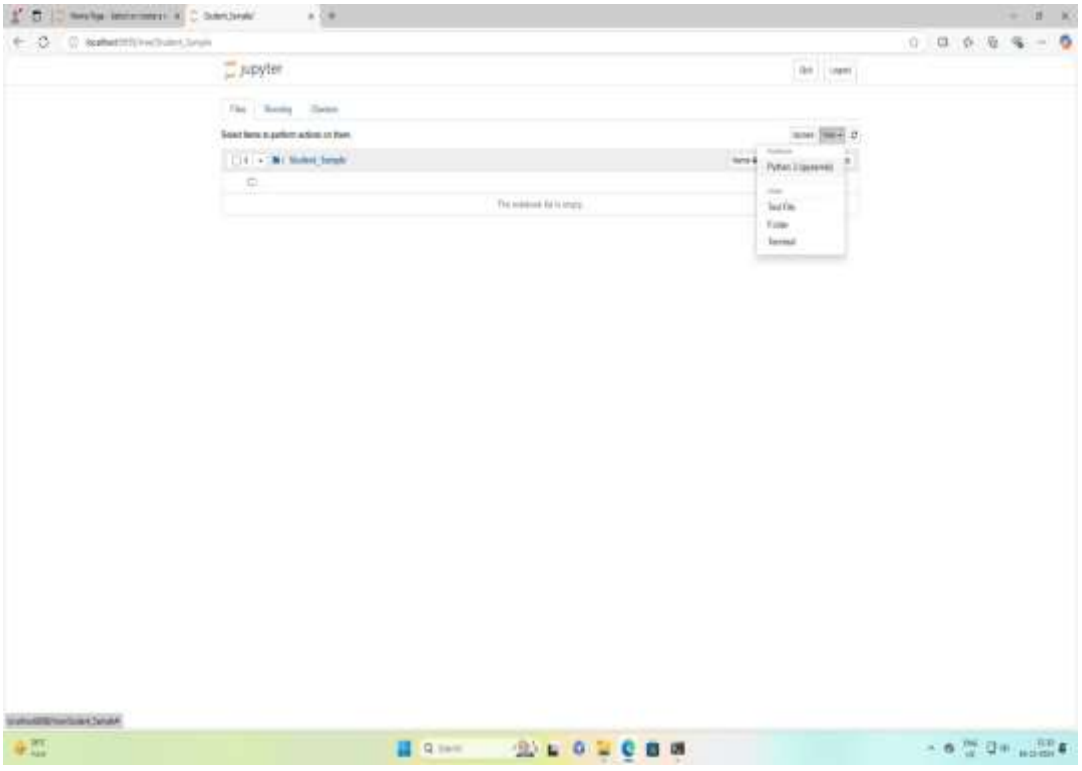


Fig 4: Creating a python notebook.

Step 5: Renaming the python notebook. Initially, the python program will be created with the default name “Untitled” as shown in Figure 5. Click on it to rename the notebook appropriately.

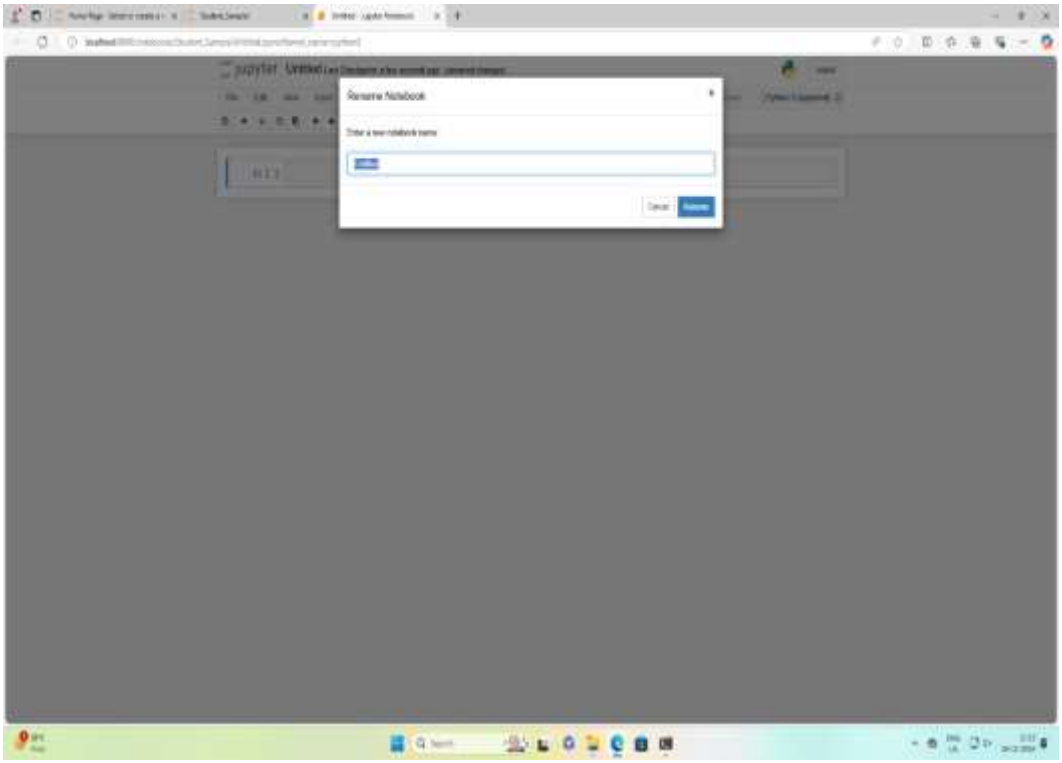


Fig 5: Renaming the python notebook.

Step 6: Start running the python program using notebook cells. After renaming the notebook cells, we can start coding. A sample print statement is displayed in Figure 6. After typing the code, press the run button to run the appropriate cell. The output is displayed below the cell.

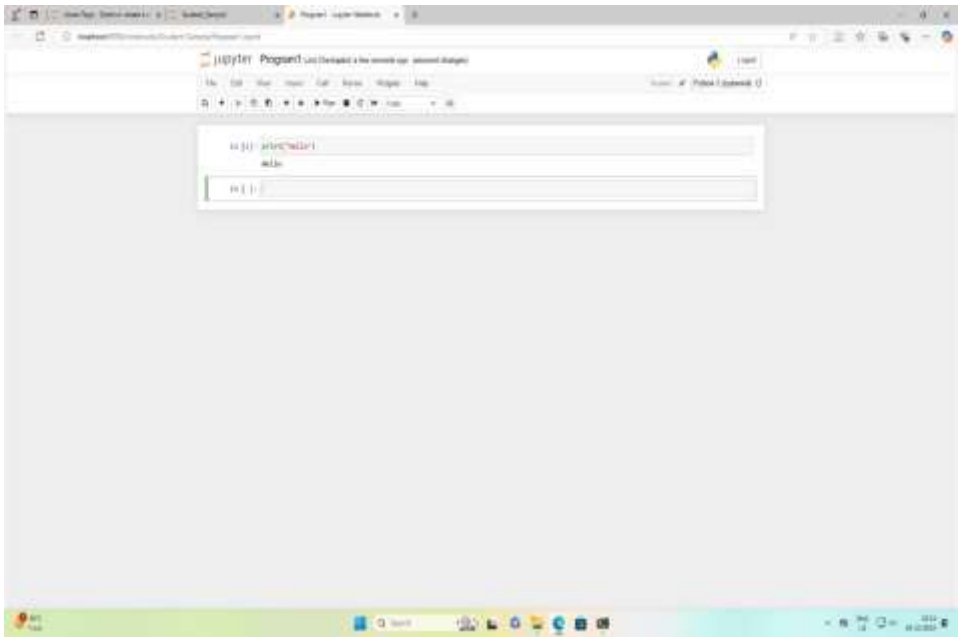


Fig 6: Sample print statement successfully executed in a Jupyter notebook.

5.3 Python Basics

Python is a high-level, interpreted programming language with simple and readable syntax. Python is dynamically typed, and variable types are determined at runtime. Its numerous third-party packages make it suitable for various applications, such as web development, data analysis, artificial intelligence, and scientific computing. For data manipulation, libraries like Pandas and NumPy provide efficient tools for handling structured and unstructured data, performing calculations, and managing large datasets. Python also offers various data visualization libraries, such as Matplotlib and Seaborn, to create interactive charts, graphs, and dashboards. Machine learning and statistical modeling can seamlessly integrate using libraries like Scikit-learn and Statsmodels. Jupyter Notebook enhances the workflow by allowing live coding, visualization, and storytelling in a single environment. Python's scalability, active community support, and compatibility with big data frameworks (e.g., PySpark) make it a go-to language for data-driven projects across industries. Important features of Python are as follows:

- Python uses whitespace (tabs or spaces) to structure code.
- To add comments to code use the hash mark (pound sign) #.
- Variables in Python have no inherent type associated with them; a variable can refer to a different type of object simply by doing an assignment.
- Every number, string, data structure, function, class, module etc. exists in the Python interpreter as a Python object. Each object has an associated type (e.g., integer, string, or function) and internal data.
- Objects in Python typically have both attributes (other Python objects stored “inside” the object) and methods (functions associated with an object that can have access to the object’s internal data).
- In Python, a module is a file with the .py extension containing Python code.
- Python’s data structures like tuples, lists, dictionaries, sets, and sequences, are important aspects of Python programming.
- Vectorization in Python refers to performing operations on entire arrays or data sequences without iterating through individual elements using loops. Vectorised operations are supported in packages like NumPy and Pandas. Python is optimized for vectorized operations and it is much faster than looping constructs.
- Python provides an array of specialized libraries for creating a variety of data visualizations. The most important packages are Matplotlib and Seaborn.

5.4 List of Experiments

Week No	TOPICS	Course Outcome Addressed
Week 1	**Demo 1: Python Language Basics** **Exercises** 1. Write a Python function to input two numbers and perform the Calculator operations of (+, -, *, /). 2. Write a Python function that takes an integer and returns True if it's a prime number and False otherwise. 3. Create a Python function that creates a sequence between 1 and 100 and prints all the odd numbers. Compute and display the sum of all the even numbers. 4. Write a Python function to add two elements and display the result. The elements can be of type integer, float or string. 5. Write a Python function that takes a string input from the user and counts the number of vowels and consonants in the string.	CO1
Week 2	**Demo 2: Python built-in Data structures, Functions, modules, packages** **Exercises** 1. Write a Python code block that inputs numbers into a list. Print the largest, smallest, the sum, and the average of the numbers. Count occurrences of a specific number in the list. 2. Write a Python code block to create a tuple with five elements. Try to change one of the elements and handle the error that occurs. Print a message that explains why the error occurred. 3. Write a Python code block to create a dictionary of cricket World Cup winners. Let the key be the year; the value is the country that won the World Cup that year. Print the name of the best-performing country. Display the unique list of countries that have won the World Cup. 4. Write a Python code block that inputs a sentence from the user. Count the frequency of each word in the sentence and store the result in a dictionary. Prints the dictionary with words as keys and their frequencies as values. 5. Write a Python code block to input numbers into two sets. Perform union, intersection, and difference operations on the sets and print the results.	CO1
Week 3	**Demo 3: NumPy basics and vectorized computation** **Exercises** 1. Generate a 3x4 NumPy array with random integers between 1 and 50. a. Calculate and print the Mean, Median, and Standard Deviation of the array b. Print the Sum of all elements and the sum of each row. c. Reshape the 3x4 array into a 2x6 array and print it.	CO2

	2. Create two (3 * 3) matrices using NumPy and print it. Perform and print the results of the following linear algebra operations a. Matrix addition b. Matrix subtraction c. Matrix multiplication (element-wise and dot product) d. Transpose of a matrix e. Determinant and inverse (if applicable)	
Week 4	**Demo 4: Pandas, Data loading, Storage and File formats** **Exercises** 1. Create a Series from a list of integers representing daily temperatures (in Celsius) over a week. Assign index labels as day of the week. a. Find and print the average (mean) temperature for the week. b. Identify and print the maximum and minimum temperatures and their respective days. c. Display the temperatures greater than a specific value. d. Convert all temperatures to Fahrenheit. e. Print the days had temperatures above the average. 2. Create a data frame with details of 10 students and columns as Roll Number, Name, Gender, Marks1, Marks2, Marks3. a. Create a new column with total marks b. Find the lowest marks in Marks1 c. Find the Highest marks in Marks2 d. Find the average marks in Marks3 e. Find student name with highest average f. Find how many students failed in Marks2 (<40)	CO2
Week 5	**Demo 5: Data Cleaning and Preparation** 1. Create a CSV file called "Movies.csv" with details of 10 movies- Movie Name, Language, Genre, Rating, Review. a. Read CSV file into a dataframe and find the movie with the highest rating. b. Write the details of all "Hindi movies into a file "HindiMovies.csv". 2. For the CEREALS dataset, perform data preprocessing and answer the following questions. a. Create a table with the 5 number summary of all the numeric attributes. b. For each of the numeric attributes (proteins upto vitamins) , identify and replace all missing data(indicated with -1) with the arithmetic mean of the attribute. c. Create a table with the 5 number summary of all the numeric attributes after treating missing values. Do you think the strategy used in dealing with missing values was effective? d. For each numeric attribute (proteins upto vitamins), identify and replace all noisy data with the median of attribute. e. Create a table with the 5 number summary of all the numeric attributes after treating noisy values. Do you think the strategy used in dealing with noisy values was effective?	CO3

Week 6	**Demo 6: Data Visualization: context, effective visuals and storytelling** **Exercise** 1. For the MTCARS dataset, answer the specified questions with summarization and effective visuals. 2. For the CEREALS dataset, answer the specified questions with summarization and effective visuals.	CO3, CO4
Week 7	**Demo 7: Plotting and Visualization using Matplotlib & Seaborn** **Exercise** 1. For the IPL dataset, answer the specified questions with summarization and effective visuals using Matplotlib & Seaborn libraries	CO4
Week 8	**Demo 8: Data Aggregation and Group Operations** **Exercise** 1. For the NORTHWIND dataset, answer the specified questions with summarization and effective visuals.	CO3
Week 9	**Demo 9: String Manipulation and Data Wrangling** **Exercise** 1. For the SENTIMENT dataset, answer the specified questions with string operations and effective visuals.	CO3
Week 10	**Discussion of case study and data set.** 1. For the case study given, answer the questions with a report with story, visuals and data summaries.	CO5
Week 11	**Discussion of case study and data set.** 1. For the case study given, answer the questions with a report with story, visuals and data summaries.	CO5
Week 12	***End-term lab examination***	

WEEK1: Demo 1: Python Language Basics Exercises

1. Write a Python function to input two numbers and perform the Calculator operations of (+, -,*, /).

Program:

```
def calculator():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))

    print("Addition:", a + b)
    print("Subtraction:", a - b)
    print("Multiplication:", a * b)

    if b != 0:
        print("Division:", a / b)
    else:
        print("Division: Cannot divide by zero")

# Function call
calculator()
```

Output:

```
Enter first number: 4
Enter second number: 5
Addition: 9.0
Subtraction: -1.0
Multiplication: 20.0
Division: 0.8
```

2. **Write a Python function that takes an integer and returns True if it's a prime number and False otherwise**

Program:

```
def isprime():
    n=int(input('enter n'))
    count=0
    for i in range(1,n+1):
        if(n%i == 0):
            count=count+1
    if(count==2):
        print("prime")
    else:
        print("not prime")

isprime()
```

Output:

```
enter n13
prime
```

3. Create a Python function that creates a sequence between 1 and 100 and prints all the odd numbers. Compute and display the sum of all the even numbers.

Program:

```
def odd_and_even_sum():
    even_sum = 0

    for num in range(1, 101):
        if num % 2 != 0:
            print(num, end=" ")
        else:
            even_sum += num

    print("\nSum of all even numbers:", even_sum)

# Function call
odd_and_even_sum()
```

Output:

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
    51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93
    95 97 99
Sum of all even numbers: 2550
```

4.Write a Python function to add two elements and display the result. The elements can be of type integer, float or string.

Program:

```
def add_elements(x, y):
    try:
        result = x + y
        print("Result:", result)
    except TypeError:
        print("Cannot add elements of incompatible types:", type(x), type(y))

# Examples
add_elements(5, 3)
add_elements(3.5, 2.5)
add_elements('hello ', 'world')
```

Output:

```
Result: 8
Result: 6.0
Result: hello world
```

5. Write a Python function that takes a string input from the user and counts the number of vowels and consonants in the string

Program:

```
def count_vowels_consonants():
    text = input("Enter a string: ").lower()
    vowels = "aeiou"
    vowel_count = 0
    consonant_count = 0

    for ch in text:
        if ch.isalpha():
            if ch in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    print("Number of vowels:", vowel_count)
    print("Number of consonants:", consonant_count)

# Function call
count_vowels_consonants()
```

Output:

```
Enter a string: appleisa fruit
Number of vowels: 6
Number of consonants: 7
```

WEEK 2: Demo 2: Python built-in Data structures, Functions, modules, packages

1. Write a Python code block that inputs numbers into a list. Print the largest, smallest, the sum, and the average of the numbers. Count occurrences of a specific number in the list.

Program:

```
# Input numbers into a list
n = int(input("Enter number of elements: "))
numbers = []

for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    numbers.append(num)

# Display the list
print("List:", numbers)

# Calculations
maximum = max(numbers)
minimum = min(numbers)
total = sum(numbers)
average = total / n

# Output results
print("Maximum:", maximum)
print("Minimum:", minimum)
print("Sum:", total)
print("Average:", average)

# Count occurrences of a number
x = int(input("Enter number to count: "))
count = numbers.count(x)
print(f"Occurrences of {x}:", count)
```

Output:

```
Enter number of elements: 5
Enter element 1: 12
Enter element 2: 34
Enter element 3: 2
Enter element 4: 34
Enter element 5: 56
List: [12, 34, 2, 34, 56]
Maximum: 56
Minimum: 2
Sum: 138
Average: 27.6
Enter number to count:
```

2. Write a Python code block to create a tuple with five elements. Try to change one of the elements and handle the error that occurs. Print a message that explains why the error occurred.

Program:

```
# Create a tuple with five elements
my_tuple = (10, 20, 30, 40, 50)

try:
    # Try to change one element of the tuple
    my_tuple[2] = 100
except TypeError as e:
    print("Error occurred!")
    print("Reason:", e)
    print("Explanation: Tuples are immutable, so their elements cannot be changed.")
```

Output:

```
Error occurred!
Reason: 'tuple' object does not support item assignment
Explanation: Tuples are immutable, so their elements can
```

3.Write a Python code block to create a dictionary of cricket World Cup winners. Let the key be the year; the value is the country that won the World Cup that year. Print the name of the best-performing country. Display the unique list of countries that have won the World Cup.

Program:

```
# Dictionary of Cricket World Cup winners
# Key: Year, Value: Country
world_cup_winners = {
    1975: "West Indies",
    1979: "West Indies",
    1983: "India",
    1987: "Australia",
    1992: "Pakistan",
    1996: "Sri Lanka",
    1999: "Australia",
    2003: "Australia",
    2007: "Australia",
    2011: "India",
    2015: "Australia",
    2019: "England"
}

# Find the best-performing country (most wins)
winner_counts = {}

for country in world_cup_winners.values():
    winner_counts[country] = winner_counts.get(country, 0) + 1

best_country = max(winner_counts, key=winner_counts.get)
```

```
print("Best-performing country:", best_country)

# Display unique list of countries that won the World Cup
unique_countries = set(world_cup_winners.values())
print("Countries that have won the World Cup:",  unique_countries)
```

Output:

```
Best-performing country: Australia
Countries that have won the World Cup: {'India', 'Pakist
```

4.Write a Python code block that inputs a sentence from the user. Count the frequency of each word in the sentence and store the result in a dictionary. Prints the dictionary with words as keys and their frequencies as values.

Program:

```
# Input a sentence from the user
sentence = input("Enter a sentence: ")

# Convert sentence to lowercase and split into words
words = sentence.lower().split()

# Dictionary to store word frequencies
word_count = {}

# Count frequency of each word
for word in words:
    word_count[word] = word_count.get(word, 0) + 1

# Print the dictionary
print("Word frequencies:")
print(word_count)
```

Output:

```
Enter a sentence: This is data visualization third lab
Word frequencies:
{'this': 1, 'is': 1, 'data': 1, 'visualization': 1, 'third': 1, 'lab': 1}
```

5.Write a Python code block to input numbers into two sets. Perform union, intersection, and difference operations on the sets and print the results.

Program:

```
# Input elements for first set
n1 = int(input("Enter number of elements in Set 1: "))
set1 = set()

for i in range(n1):
    set1.add(int(input(f"Enter element {i+1} for Set 1: ")))

# Input elements for second set
n2 = int(input("Enter number of elements in Set 2: "))
set2 = set()
```

```
for i in range(n2):
    set2.add(int(input(f"Enter element {i+1} for Set 2: ")))

# Display sets
print("Set 1:", set1)
print("Set 2:", set2)

# Set operations
print("Union:", set1.union(set2))
print("Intersection:", set1.intersection(set2))
print("Difference (Set1 - Set2):", set1.difference(set2))
print("Difference (Set2 - Set1):", set2.difference(set1))
```

Output:

```
Set 1: {1, 2, 3, 4}
Set 2: {3, 4, 5, 6}
Union: {1, 2, 3, 4, 5, 6}
Intersection: {3, 4}
Difference (Set1 - Set2): {1, 2}
Difference (Set2 - Set1): {5, 6}
```

WEEK 3: NumPy basics and vectorized computation Exercises

1. Generate a 3x4 NumPy array with random integers between 1 and 50.
   a. Calculate and print the Mean, Median, and Standard Deviation of the array
b. Print the Sum of all elements and the sum of each row.
c. Reshape the 3x4 array into a 2x6 array and print it.

Program:

```
import numpy as np
arr = np.random.randint(1,51,size=(3,4))
print("Array:\n", arr)
print("Mean:", arr.mean())
print("Median:", np.median(arr))
print("Std Dev:", arr.std())
print("Sum of all elements:", arr.sum())
print("Sum of each row:", arr.sum(axis=1))
arr2 = arr.reshape(2,6)
print("Reshaped to 2x6:\n", arr2)
```

Output:

```
Array:
 [[23 29  9 35]
 [26 24 34 29]
 [49  2 24  4]]
Mean: 24.0
Median: 25.0
Std Dev: 12.942179105544785
Sum of all elements: 288
Sum of each row: [ 96 113  79]
Reshaped to 2x6:
 [[23 29  9 35 26 24]
 [34 29 49  2 24  4]]
```

2.Create two (3 * 3) matrices using NumPy and print it. Perform and print the results of the following linear algebra operations

a. Matrix addition
b. Matrix subtraction
c. Matrix multiplication (element-wise and dot product)
d. Transpose of a matrix
e. Determinant and inverse (if applicable).

Program:

```
import numpy as np

# Create two 3x3 matrices
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

B = np.array([[9, 8, 7],
              [6, 5, 4],
              [3, 2, 1]])
```

```
print("Matrix A:")
print(A)

print("\nMatrix B:")
print(B)

# a. Matrix addition
print("\nMatrix Addition (A + B):")
print(A + B)

# b. Matrix subtraction
print("\nMatrix Subtraction (A - B):")
print(A - B)

# c. Matrix multiplication
print("\nElement-wise Multiplication (A * B):")
print(A * B)

print("\nDot Product (A dot B):")
print(np.dot(A, B))

# d. Transpose of a matrix
print("\nTranspose of Matrix A:")
print(A.T)

# e. Determinant and Inverse
det_A = np.linalg.det(A)
print("\nDeterminant of Matrix A:")
print(det_A)

if det_A != 0:
    inv_A = np.linalg.inv(A)
    print("\nInverse of Matrix A:")
    print(inv_A)
else:
    print("\nInverse of Matrix A does not exist (Determinant is zero).")
```

Output:

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

Matrix B:
[[9 8 7]
 [6 5 4]
 [3 2 1]]

Matrix Addition (A + B):
[[10 10 10]
 [10 10 10]
 [10 10 10]]
```

Matrix Subtraction (A - B):
[[-8 -6 -4]
 [-2 0 2]
 [4 6 8]]

Element-wise Multiplication (A * B):
[[9 16 21]
 [24 25 24]
 [21 16 9]]

Dot Product (A dot B):
[[30 24 18]
 [84 69 54]
 [138 114 90]]

Transpose of Matrix A:
[[1 4 7]
 [2 5 8]
 [3 6 9]]

Determinant of Matrix A:
0.0

Inverse of Matrix A does not exist (Determinant is zero).

WEEK 4: Pandas, Data loading, Storage and File formats Exercises

1. Create a Series from a list of integers representing daily temperatures (in Celsius) over a week. Assign index labels as day of the week.
    a. Find and print the average (mean) temperature for the week.
    b. Identify and print the maximum and minimum temperatures and their respective days.
    c. Display the temperatures greater than a specific value.
    d. Convert all temperatures to Fahrenheit.
    e. Print the days had temperatures above the average

Program:

```
import pandas as pd
temps = [30, 32, 29, 31, 28, 35, 33]
days = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']
s = pd.Series(temps, index=days)
print("Mean temp:", s.mean())
print("Max:", s.max(), "on", s.idxmax())
print("Min:", s.min(), "on", s.idxmin())
val = float(input("Show temps greater than: "))
print(s[s>val])
f = s * 9/5 + 32
print("Fahrenheit:\n", f)
print("Days above average:", list(s[s > s.mean()].index))
```

Output:

```
Mean temp: 31.142857142857142
Max: 35 on Sat
Min: 28 on Fri
Show temps greater than:
```

2. Create a data frame with details of 10 students and columns as Roll Number, Name, Gender, Marks1, Marks2, Marks3.
    a. Create a new column with total marks
    b. Find the lowest marks in Marks1
    c. Find the Highest marks in Marks2
    d. Find the average marks in Marks3
    e. Find student name with highest average
    f. Find how many students failed in Marks2 (<40)

Program:

```
import pandas as pd

# Create DataFrame with student details
data = {
   "Roll Number": [1,2,3,4,5,6,7,8,9,10],
   "Name": ["Asha","Bala","Charan","Divya","Eshan",
         "Farah","Ganesh","Hema","Irfan","Jaya"],
   "Gender": ["F","M","M","F","M","F","M","F","M","F"],
   "Marks1": [78, 45, 66, 89, 34, 56, 90, 41, 72, 60],
   "Marks2": [65, 38, 70, 88, 42, 35, 91, 55, 39, 76],
   "Marks3": [80, 67, 72, 90, 45, 60, 85, 50, 78, 69]
```

```
}

df = pd.DataFrame(data)

print("Student DataFrame:")
print(df)

# a. Create a new column with total marks
df["Total"] = df["Marks1"] + df["Marks2"] + df["Marks3"]

# Average marks column (needed for part e)
df["Average"] = df["Total"] / 3

print("\nDataFrame with Total and Average:")
print(df)

# b. Lowest marks in Marks1
print("\nLowest marks in Marks1:", df["Marks1"].min())

# c. Highest marks in Marks2
print("Highest marks in Marks2:", df["Marks2"].max())

# d. Average marks in Marks3
print("Average marks in Marks3:", df["Marks3"].mean())

# e. Student name with highest average
top_student = df.loc[df["Average"].idxmax(), "Name"]
print("Student with highest average marks:", top_student)

# f. Number of students failed in Marks2 (< 40)
failed_count = (df["Marks2"] < 40).sum()
print("Number of students failed in Marks2:", failed_count)
```

Output:

```
Student DataFrame:
   Roll Number    Name Gender  Marks1  Marks2  Marks3
0            1    Asha      F      78      65      80
1            2    Bala      M      45      38      67
2            3  Charan      M      66      70      72
3            4   Divya      F      89      88      90
4            5   Eshan      M      34      42      45
5            6   Farah      F      56      35      60
6            7  Ganesh      M      90      91      85
7            8    Hema      F      41      55      50
8            9   Irfan      M      72      39      78
9           10    Jaya      F      60      76      69

DataFrame with Total and Average:
   Roll Number    Name Gender  Marks1  Marks2  Marks3  Total    Average
0            1    Asha      F      78      65      80    223  74.333333
1            2    Bala      M      45      38      67    150  50.000000
2            3  Charan      M      66      70      72    208  69.333333
3            4   Divya      F      89      88      90    267  89.000000
4            5   Eshan      M      34      42      45    121  40.333333
```

5	6	Farah	F	56	35	60	151	50.333333
6	7	Ganesh	M	90	91	85	266	88.666667
7	8	Hema	F	41	55	50	146	48.666667
8	9	Irfan	M	72	39	78	189	63.000000
9	10	Jaya	F	60	76	69	205	68.333333

Lowest marks in Marks1: 34
Highest marks in Marks2: 91
Average marks in Marks3: 69.6
Student with highest average marks: Divya
Number of students failed in Marks2: 3

WEEK5: Data Cleaning and Preparation

1. Create a CSV file called “Movies.csv” with details of 10 movies- Movie Name, Language, Genre, Rating, Review.
   a. Read CSV file into a dataframe and find the movie with the highest rating.
   b. Write the details of all “Hindi movies into a file “HindiMovies.csv”

Program:

```
import pandas as pd

# Create movie data
data = {
    "Movie Name": ["Dangal", "Inception", "Bahubali", "Titanic", "3 Idiots",
                   "Avatar", "KGF", "Interstellar", "PK", "Gladiator"],
    "Language": ["Hindi", "English", "Telugu", "English", "Hindi",
                 "English", "Kannada", "English", "Hindi", "English"],
    "Genre": ["Drama", "Sci-Fi", "Action", "Romance", "Comedy",
              "Sci-Fi", "Action", "Sci-Fi", "Comedy", "Action"],
    "Rating": [8.4, 8.8, 8.0, 7.9, 8.4, 7.8, 8.2, 8.6, 8.1, 8.5],
    "Review": ["Excellent", "Excellent", "Very Good", "Good", "Excellent",
               "Good", "Very Good", "Excellent", "Very Good", "Excellent"]
}

# Create DataFrame and save to CSV
df = pd.DataFrame(data)
df.to_csv("Movies.csv", index=False)

print("Movies.csv file created")
# Read CSV file
df = pd.read_csv("Movies.csv")

# Find movie with highest rating
top_movie = df.loc[df["Rating"].idxmax(), "Movie Name"]

print("Movie with highest rating:", top_movie)

# Filter Hindi movies
hindi_movies = df[df["Language"] == "Hindi"]

# Write to new CSV file
hindi_movies.to_csv("HindiMovies.csv", index=False)

print("HindiMovies.csv file created")
```

Output:
Movies.csv file created
Movie with highest rating: Inception
HindiMovies.csv file created

2. For the CEREALS dataset, perform data preprocessing and answer the following questions.
   a. Create a table with the 5 number summary of all the numeric attributes.
   b. For each of the numeric attributes (proteins upto vitamins) , identify and replace all missing data(indicated with -1) with the arithmetic mean of the attribute.
   c. Create a table with the 5 number summary of all the numeric attributes after treating missing values. Do you think the strategy used in dealing with missing values was effective?
   d. For each numeric attribute (proteins upto vitamins), identify and replace all noisy data with the median of attribute.
   e. Create a table with the 5 number summary of all the numeric attributes after treating noisy values. Do you think the strategy used in dealing with noisy values was effective?
   Program:

```
import pandas as pd
import numpy as np

# Read the dataset
df = pd.read_csv("cereals.csv")
print("CEREALS dataset loaded successfully")

# Select numeric attributes from proteins to vitamins
numeric_cols = df.loc[:, "proteins":"vitamins"].columns

# a. 5-number summary before preprocessing
print("\n5-number summary before preprocessing:")
print(df[numeric_cols].describe())

# b. Replace missing values (-1) with mean
for col in numeric_cols:
    mean_val = df[col][df[col] != -1].mean()
    df[col] = df[col].replace(-1, mean_val)

print("\nMissing values (-1) replaced with mean")

# c. 5-number summary after treating missing values
print("\n5-number summary after replacing missing values:")
print(df[numeric_cols].describe())

print("\nConclusion:")
print("Replacing missing values with mean is effective as it preserves data distribution.")

# d. Replace noisy data (outliers) with median
for col in numeric_cols:
    median_val = df[col].median()
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1

    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr

    df[col] = np.where((df[col] < lower) | (df[col] > upper),
                       median_val,
                       df[col])

print("\nNoisy values replaced with median")
```

```
# e. 5-number summary after treating noisy values
print("\n5-number summary after treating noisy values:")
print(df[numeric_cols].describe())

print("\nConclusion:")
print("Replacing noisy values with median is effective as median is robust to outliers.")
```

WEEK 6: Data Visualization: context, effective visuals and storytelling

1. For the MTCARS dataset, answer the specified questions with summarization and effective visuals

Program:

```
# mtcars_analysis.py
# Program: Summarization and Visualization of MTCARS Dataset

import pandas as pd
import matplotlib.pyplot as plt

# ------------------------------
# 1. Load the dataset
# ------------------------------
df = pd.read_csv("mtcars.csv")
print("MTCARS dataset loaded successfully\n")

# ------------------------------
# 2. Basic information
# ------------------------------
print("Dataset Info:")
print(df.info())
print("\nStatistical Summary:")
print(df.describe())

# ------------------------------
# 3. Average mileage (mpg)
# ------------------------------
avg_mpg = df["mpg"].mean()
print(f"\nAverage Mileage (mpg): {avg_mpg:.2f}")

# ------------------------------
# 4. Car with highest mileage
# ------------------------------
highest_mpg_car = df.loc[df["mpg"].idxmax()]
print("\nCar with highest mileage:")
print(highest_mpg_car)

# ------------------------------
# 5. Count of cars by cylinders
# ------------------------------
cyl_counts = df["cyl"].value_counts()
print("\nNumber of cars by cylinders:")
print(cyl_counts)

# ------------------------------
# 6. VISUALIZATIONS
# ------------------------------

# a. Histogram of Mileage
plt.figure(figsize=(6,4))
plt.hist(df["mpg"], bins=10, color='skyblue', edgecolor='black')
plt.xlabel("Miles per Gallon (mpg)")
plt.ylabel("Frequency")
plt.title("Distribution of Mileage")
plt.grid(axis='y')
```

```
plt.show()

# b. Bar chart: Number of cars by cylinders
plt.figure(figsize=(6,4))
cyl_counts.plot(kind="bar", color='lightgreen', edgecolor='black')
plt.xlabel("Number of Cylinders")
plt.ylabel("Count")
plt.title("Cars by Cylinder Count")
plt.show()

# c. Scatter plot: Weight vs Mileage
plt.figure(figsize=(6,4))
plt.scatter(df["wt"], df["mpg"], color='coral', edgecolor='black')
plt.xlabel("Weight (1000 lbs)")
plt.ylabel("Miles per Gallon (mpg)")
plt.title("Weight vs Mileage")
plt.grid(True)
plt.show()

# d. Box plot of mileage
plt.figure(figsize=(6,4))
plt.boxplot(df["mpg"], patch_artist=True, boxprops=dict(facecolor='lightblue'))
plt.ylabel("Miles per Gallon (mpg)")
plt.title("Boxplot of Mileage")
plt.grid(axis='y')
plt.show()

print("\nAnalysis complete. Visualizations displayed successfully.")
```

Output:

```
MTCARS dataset loaded successfully

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   model   32 non-null     object
 1   mpg     32 non-null     float64
 2   cyl     32 non-null     int64
 3   disp    32 non-null     float64
 4   hp      32 non-null     int64
 5   drat    32 non-null     float64
 6   wt      32 non-null     float64
 7   qsec    32 non-null     float64
 8   vs      32 non-null     int64
 9   am      32 non-null     int64
 10  gear    32 non-null     int64
 11  carb    32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
None
```

Statistical Summary:

	mpg	cyl	disp	hp	drat	wt	\
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	
mean	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	
std	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	
min	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	
25%	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	
50%	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	
75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	

	qsec	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.0000
mean	17.848750	0.437500	0.406250	3.687500	2.8125
std	1.786943	0.504016	0.498991	0.737804	1.6152
min	14.500000	0.000000	0.000000	3.000000	1.0000
25%	16.892500	0.000000	0.000000	3.000000	2.0000
50%	17.710000	0.000000	0.000000	4.000000	2.0000
75%	18.900000	1.000000	1.000000	4.000000	4.0000
max	22.900000	1.000000	1.000000	5.000000	8.0000

Average Mileage (mpg): 20.09

Car with highest mileage:

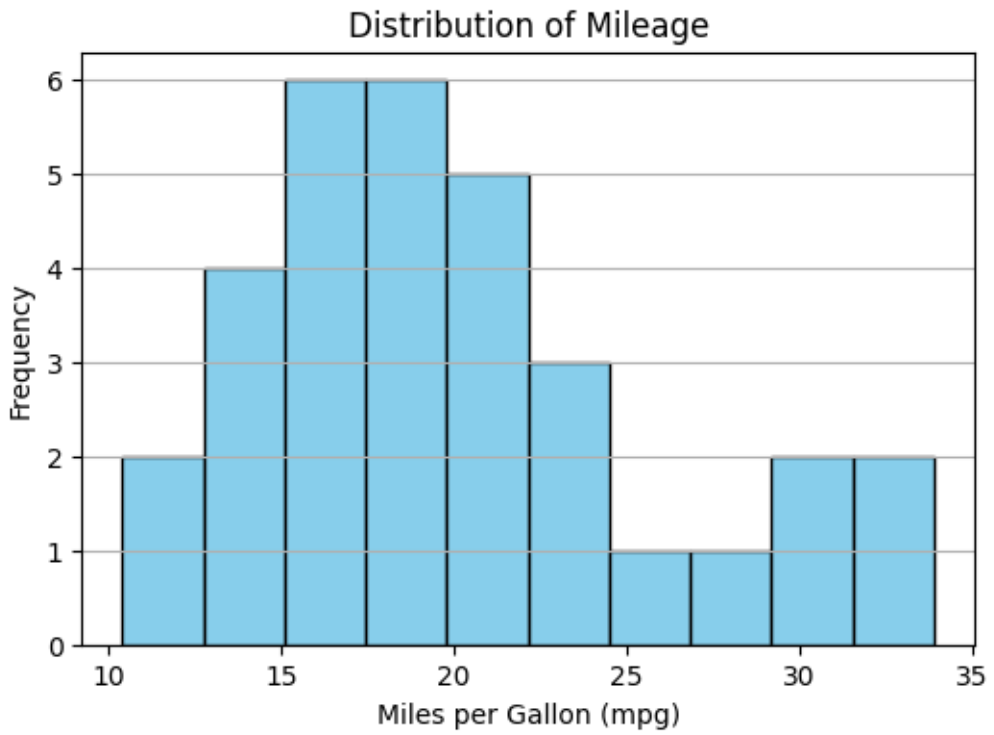
model	Toyota Corolla
mpg	33.9
cyl	4
disp	71.1
hp	65
drat	4.22
wt	1.835
qsec	19.9
vs	1
am	1
gear	4
carb	1

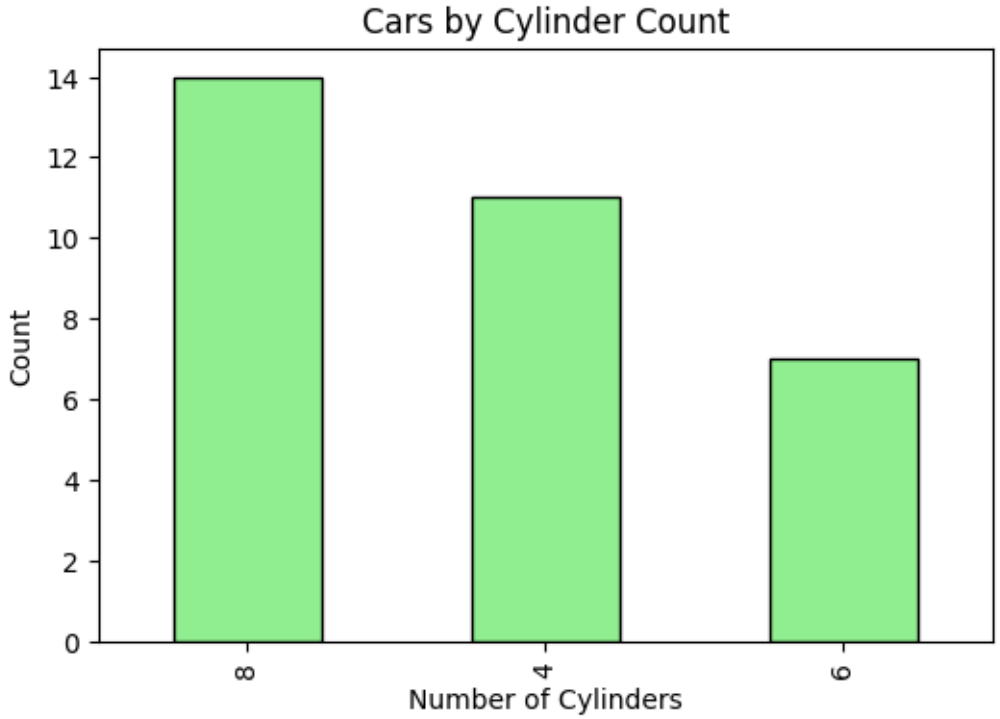
Name: 19, dtype: object

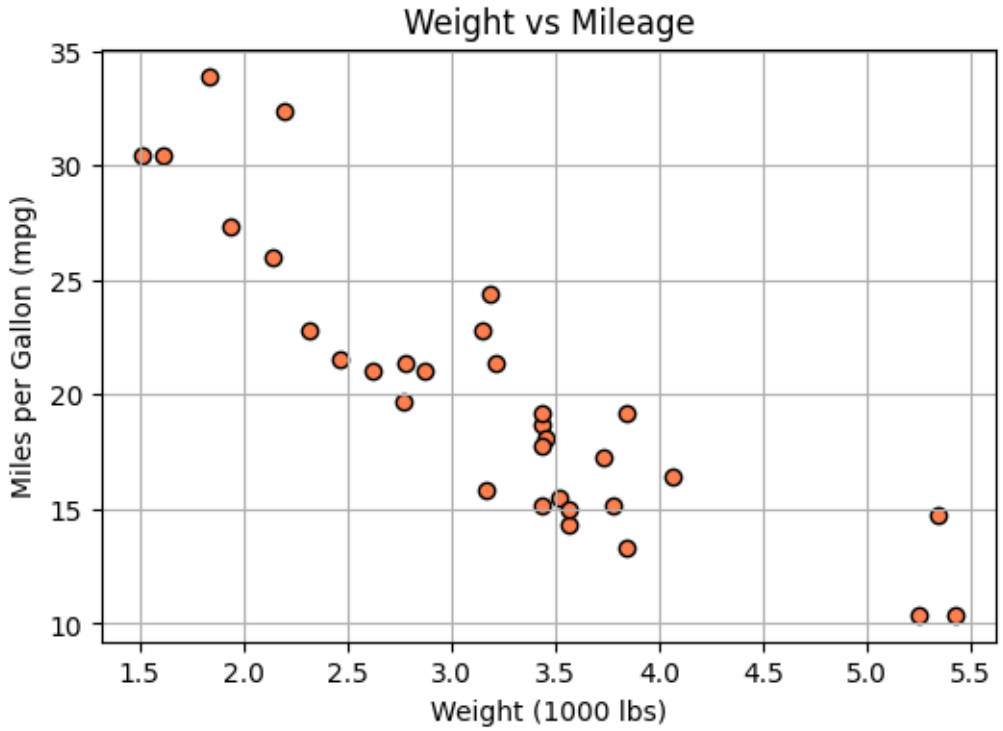
Number of cars by cylinders:

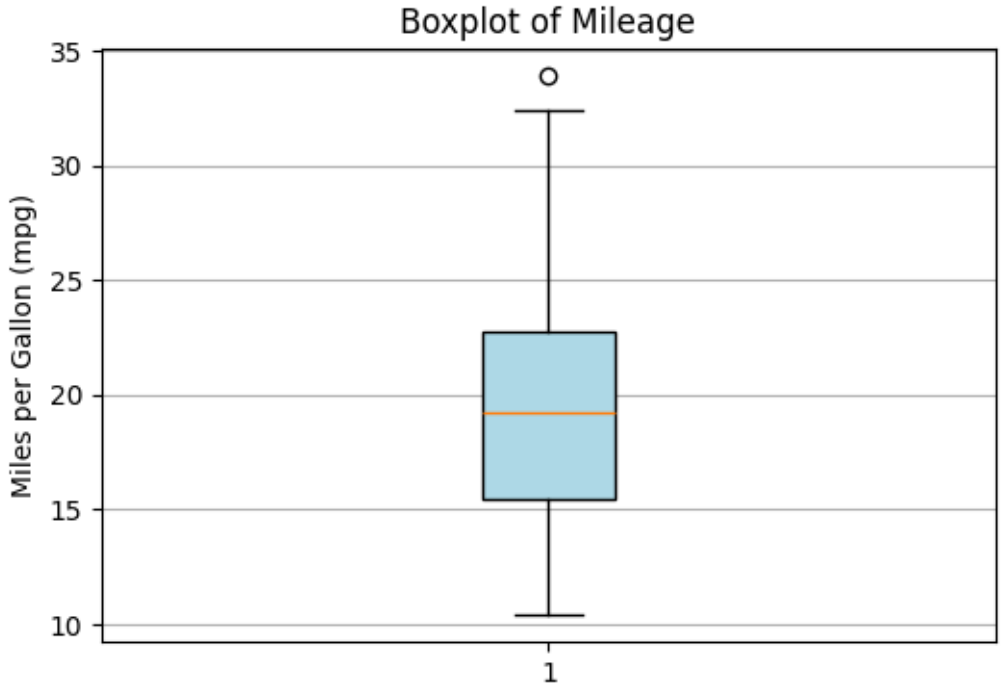
cyl	
8	14
4	11
6	7

Name: count, dtype: int64


Distribution of Mileage
Frequency
Miles per Gallon (mpg)
0
1
2
3
4
5
6
10
15
20
25
30
35


Cars by Cylinder Count
Count
Number of Cylinders
0
2
4
6
8
10
12
14
8
4
6





Analysis complete. Visualizations displayed successfully.

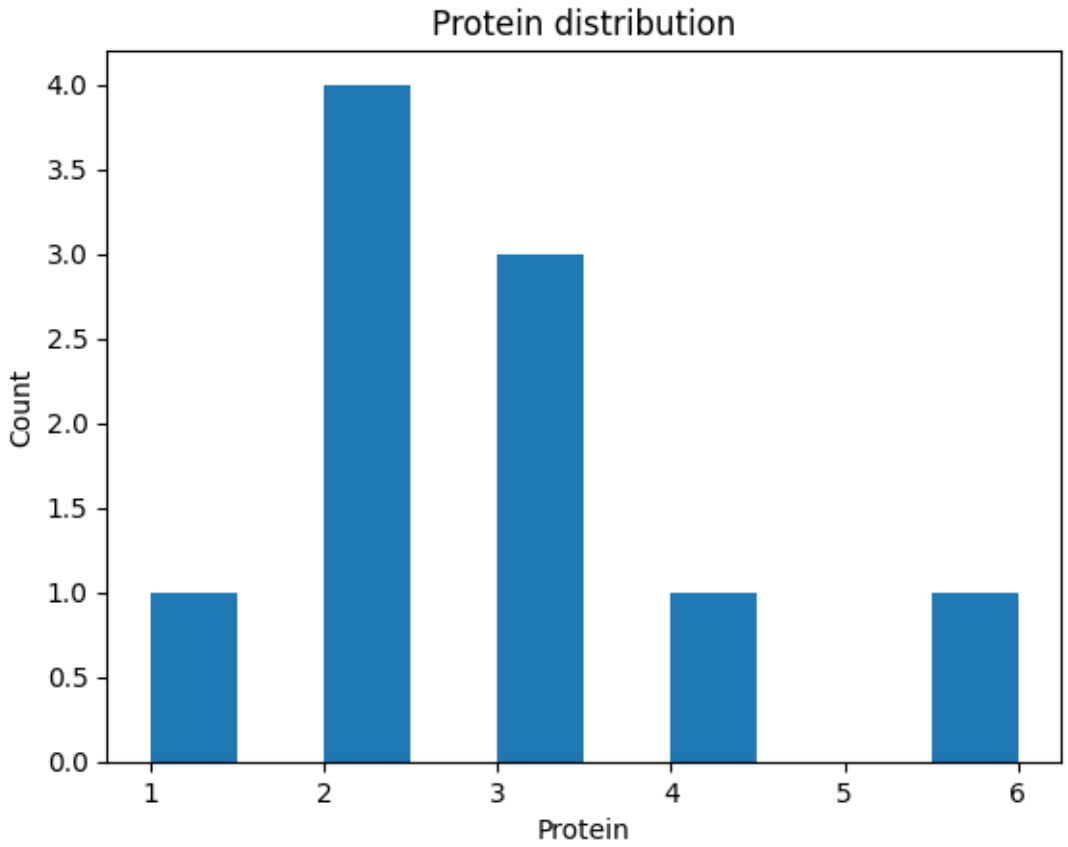
2. For the CEREALS dataset, answer the specified questions with summarization and effective visuals.
    a) 5-number summary of numeric attributes
    b) Handling missing values (-1) using mean
    c) Handling noisy/outlier values using median
    d) Visualizations for numeric attributes

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('cereals.csv')
plt.hist(df['protein'].dropna())
plt.title('Protein distribution')
plt.xlabel('Protein')
plt.ylabel('Count')
plt.savefig('protein_hist.png')
print('Plot saved to protein_hist.png')
```

Output:

Plot saved to protein_hist.png



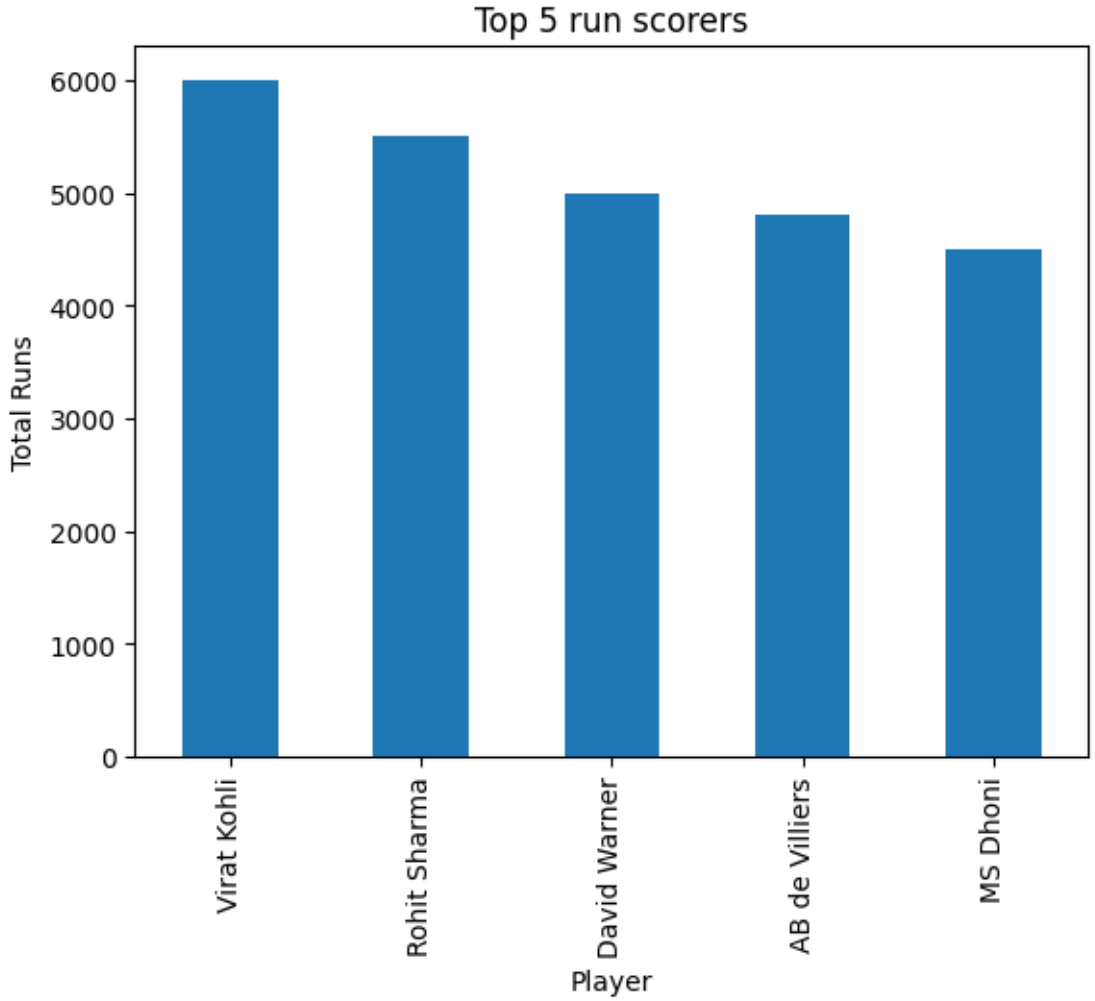
WEEK 7: : Plotting and Visualization using Matplotlib & Seaborn (IPL dataset)

For the IPL dataset, answer the specified questions with summarization and effective visuals using Matplotlib & Seaborn libraries

Question: Top 5 run scorers across all seasons and bar chart.
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
# Ensure ipl_batting.csv with columns 'Player,'Runs' is present
df = pd.read_csv('ipl batting.csv')
top = df.groupby('Player')['Runs'].sum().nlargest(5)
top.plot(kind='bar')
plt.ylabel('Total Runs')
plt.title('Top 5 run scorers')
plt.savefig('ipl_top5_runs.png')
```

Output:



Question: Team with most match wins.
Program:

```
import pandas as pd
df_matches = pd.read_csv('ipl_matches.csv')  # ensure 'winner' column exists
winners = df_matches['winner'].value_counts()
print('Team with most wins:', winners.idxmax(), 'Count:', winners.max())
```

Question: Average score batting first vs second with boxplot.
Program:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('ipl_matches.csv')  # ensure 'bat_first' and 'total_runs' columns or similar
# Example assumes a column 'batting_order' with values 'first'/'second'
sns.boxplot(x='batting_order', y='total_runs', data=df)
plt.title('Scores by batting order')
plt.savefig('ipl_scores_boxplot.png')
```

Question: Player strike rates for a season - top 10.
Program:

```
import pandas as pd
df = pd.read_csv('ipl_batting.csv')  # columns: 'player','runs','balls','season'
df['strike_rate'] = df['runs'] / df['balls'] * 100
season = input("Enter season year (e.g., 2020): ")
top10 = df[df['season']==int(season)].nlargest(10,'strike_rate')
print(top10[['player','strike_rate']])
```

Question: Distribution of extras conceded by teams; bar chart.
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('ipl_matches.csv')
extras_by_team = df.groupby('bowling_team')['extras'].sum().sort_values(ascending=False)
extras_by_team.plot(kind='bar')
plt.title('Extras conceded by team')
plt.savefig('ipl_extras_by_team.png')
```

Week 8: Data Aggregation and Group Operations (NORTHWIND dataset)

Question: Top 5 products by total sales and bar chart.
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('northwind_orders.csv') # columns: 'ProductName','Quantity','UnitPrice'
df['Sales'] = df['Quantity'] * df['UnitPrice']
top = df.groupby('ProductName')['Sales'].sum().nlargest(5)
top.plot(kind='bar')
plt.savefig('nw_top5_products.png')
```

Question: Country with highest number of orders.
Program:

```
import pandas as pd
orders = pd.read_csv('northwind_orders.csv')
print('Country with highest orders:', orders['ShipCountry'].value_counts().idxmax())
```

Question: Monthly sales trend line chart.

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
orders = pd.read_csv('northwind_orders.csv', parse_dates=['OrderDate'])
orders['Sales'] = orders['Quantity'] * orders['UnitPrice']
monthly = orders.set_index('OrderDate').resample('M')['Sales'].sum()
monthly.plot()
plt.savefig('nw_monthly_sales.png')
```

Question: Customers with most returns (if returns dataset exists).
Program:

```
import pandas as pd
returns = pd.read_csv('northwind_returns.csv')  # columns: 'CustomerID', ...
cust_returns = returns['CustomerID'].value_counts().head(10)
print(cust_returns)
```

Question: Salesperson performance: total sales per employee.
Program:

```
import pandas as pd
emp = pd.read_csv('northwind_orders.csv')
emp['Sales'] = emp['Quantity'] * emp['UnitPrice']
sales_by_emp = emp.groupby('EmployeeID')['Sales'].sum().sort_values(ascending=False)
print(sales_by_emp.head())
```

WEEK 9: String Manipulation and Data Wrangling (SENTIMENT dataset)

Question: Compute class distribution of sentiment dataset.
Program:

```
import pandas as pd
df = pd.read_csv('sentiment.csv') # columns: 'text','label'
print(df['label'].value_counts())
```

Question: Preprocess text: lowercase, remove punctuation, tokenize, remove stopwords (simple).
Program:

```
import re
stop = set(['the','is','in','and','to','a','of'])
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s]','', text)
    tokens = text.split()
    return [t for t in tokens if t not in stop]

# Example:
print(preprocess("This is an example!"))
```

Question: Train TF-IDF + Logistic Regression and report accuracy.

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df = pd.read_csv('sentiment.csv')
X = df['text']; y = df['label']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, random_state=42)
vec = TfidfVectorizer()
Xtr = vec.fit_transform(X_train)
Xte = vec.transform(X_test)
clf = LogisticRegression(max_iter=1000).fit(Xtr,y_train)
pred = clf.predict(Xte)
print('Accuracy:', accuracy_score(y_test,pred))
```

Question: Create confusion matrix heatmap for classifier predictions.
Program:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.savefig('sentiment_cm.png')
```

Question: Visualize most common words for positive and negative classes.
Program:

```
from collections import Counter
import pandas as pd
df = pd.read_csv('sentiment.csv')
pos = ' '.join(df[df['label']==1]['text']).lower().split()
neg = ' '.join(df[df['label']==0]['text']).lower().split()
print('Top positive words:', Counter(pos).most_common(10))
print('Top negative words:', Counter(neg).most_common(10))
```

WEEK 10: Case Study 1 — Analysis and Report

Question: Provide a 500-word narrative summarising the dataset and three key insights (use exported summary).

Program:

```
import pandas as pd
df = pd.read_csv('case_study_1.csv')
summary = df.describe(include='all')
summary.to_csv('case1_summary.csv')
print('Summary exported to case1_summary.csv. Use these numbers to write the narrative.')
```

Question: Produce 3 visualizations: distribution, trend, correlation heatmap.

Program:

```
import matplotlib.pyplot as plt
import seaborn as sns
df.hist(figsize=(8,6))
plt.savefig('case1_hist.png')
sns.heatmap(df.select_dtypes(include=[float,int]).corr(), annot=True)
plt.savefig('case1_corr.png')
```

Question: Identify two segments using clustering and describe them.

Program:

```
from sklearn.cluster import KMeans
import pandas as pd
df = pd.read_csv('case_study_1.csv')
num = df.select_dtypes(include=[float,int]).dropna()
kmeans = KMeans(n_clusters=2, random_state=42).fit(num)
df['segment'] = kmeans.labels_
print(df.groupby('segment').mean())
```

Question: Recommend 3 actionable business steps based on analysis.

Program:

```
# Interpretative: use insights from summary and plots to write recommendations.
print('Based on summary and visuals, recommend 3 actionable steps in the report.')
```

Question: Export a one-page PDF summary (template) for stakeholders (save summary CSV for conversion).

Program:

```
# Save summary to CSV that can be converted to PDF using external tools
df.describe(include='all').to_csv('case1_summary.csv')
print('Saved case1_summary.csv — convert to PDF using Word or other tools.')
```

WEEK 11: Case Study 2 — Analysis and Report

Question: Provide executive summary and 4 charts illustrating main findings.
Program:

```
import pandas as pd
df = pd.read_csv('case_study_2.csv')
print(df.describe())
# Create and save 4 charts using matplotlib/seaborn for slides.
```

Question: Time series analysis: seasonal decomposition if Date column exists.
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
df = pd.read_csv('case_study_2.csv', parse_dates=['Date'])
df.set_index('Date', inplace=True)
ts = df['Value']
res = seasonal_decompose(ts.dropna(), model='additive', period=12)
res.plot()
plt.savefig('case2_seasonal.png')
```

Question: Run hypothesis test (e.g., t-test) and report p-value.
Program:

```
from scipy.stats import ttest_ind
import pandas as pd
df = pd.read_csv('case_study_2.csv')
groupA = df[df['Group']=='A']['Metric']
groupB = df[df['Group']=='B']['Metric']
stat, p = ttest_ind(groupA.dropna(), groupB.dropna())
print('t-statistic:', stat, 'p-value:', p)
```

Question: Build simple predictive model (linear regression) and report R^2 on test set.
Program:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
df = pd.read_csv('case_study_2.csv')
X = df[['Feature1','Feature2']]; y = df['Target']
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2, random_state=42)
model = LinearRegression().fit(Xtr,ytr)
print('R^2 on test:', model.score(Xte,yte))
```

Question: Prepare a slide-ready one-page chart summarising action items (save PNG).
Program:

```
# Use matplotlib to compose a single figure summarising top metrics and save as PNG for
slides.
import matplotlib.pyplot as plt
# compose figure...
plt.figure(figsize=(8,6))
```

```
plt.text(0.1, 0.8, 'Action Item 1: ...', fontsize=12)
plt.text(0.1, 0.6, 'Action Item 2: ...', fontsize=12)
plt.axis('off')
plt.savefig('case2_action_items.png')
print('Saved case2_action_items.png')
```

References:

1	**Text Book:** Wes McKinney , Python for Data Analysis: Data Wrangling with pandas, NumPy & Jupyter. 3rd edition. O'Reilly Media, 2022.
2	Cole Nussbaumer Knaflic, Storytelling With Data: A Data Visualization Guide for Business Professionals, John Wiley and Sons, 2015.
3	Jake VanderPlas, Python Data Science Handbook. O'Reilly Media, 2016.
4	Alberto Boschetti and Luca Massaron, Python Data Science Essentials, 3rd edition, Packt Publishing Ltd. 2018.
5	Manaranjan Pradhan, U Dinesh Kumar, "Machine Learning using Python", Wiley India, 2019.
6	Python documentation: https://docs.python.org/3/