

Practical-1

Aim: Understanding Python environment Setup: Installing python with anaconda, Introduction to various Python IDEs like IDLE, Jupyter notebook, Pycharm , spyder.

Introduction: An IDE (Integrated Development Environment) is a software application that provides a set of tools for software development with a specific programming language, such as Python. An IDE typically includes features like code editing, syntax highlighting, code completion, debugging, testing, and project management. There are many IDEs available for Python programming, each with its own advantages and disadvantages. Some of the most popular ones are:

Jupyter: A web-based IDE that allows you to create and share interactive notebooks that contain code, text, images, and graphs. It is widely used for data science, machine learning, and scientific computing. It supports many programming languages besides Python, such as R, Julia, and Scala.

PyCharm: A powerful and versatile IDE that supports many web development frameworks, scientific tools, cross-technology development, remote development, and database integration. It has a professional edition that requires a license and a free community edition that is open source.

Spyder: A scientific IDE that is designed for data analysis, visualization, and exploration. It has a user-friendly interface that features a code editor, a console, a variable explorer, a plot pane, and a help pane. It integrates with popular scientific packages like NumPy, SciPy, Matplotlib, and Pandas.

Installation Guide for Anaconda Navigator In

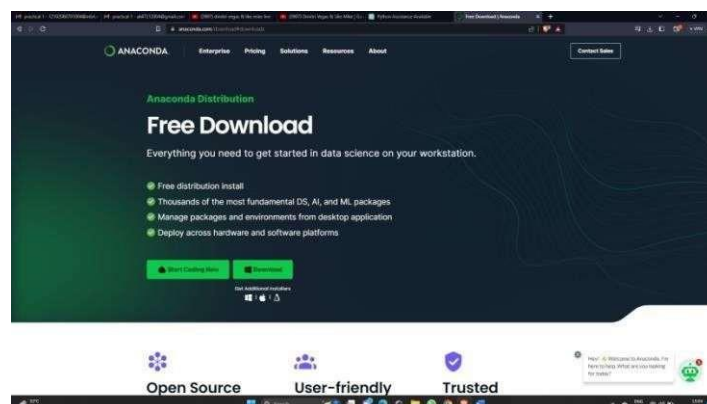
Introduction:

Anaconda Navigator is a powerful tool for Python development and data science. This installation guide will walk you through the steps to install Anaconda Navigator on your computer.

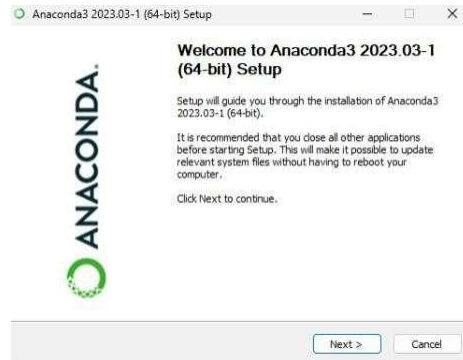
Step 1: Visit the Anaconda website:

<https://www.anaconda.com/products/individual>

Step 2: Click on the "Download" button to access the download page.

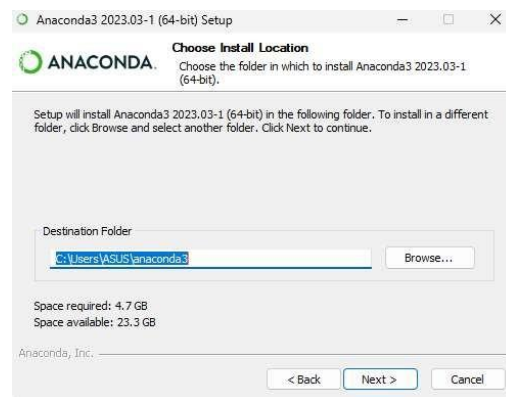


Step 3: Choose the appropriate installer for your operating system. Anaconda supports both 32-bit and 64-bit systems.



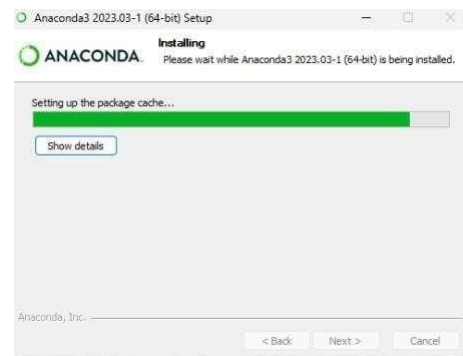
Locate the downloaded installer file on your computer. Double-click on the installer file to launch the installation. Read and accept the license agreement.

Step 4: Choose the installation location (you can usually leave the default location as is).

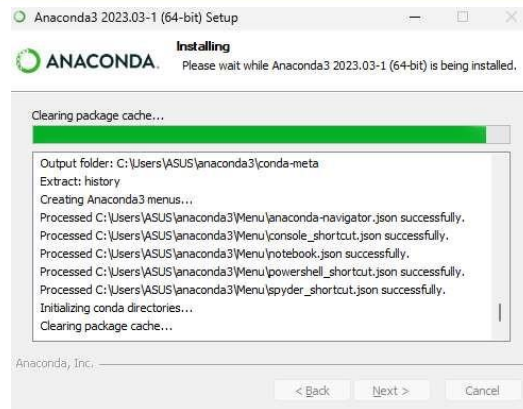


Start the installation process by clicking the “Next” button.

Step 5: Wait for the installation to complete:



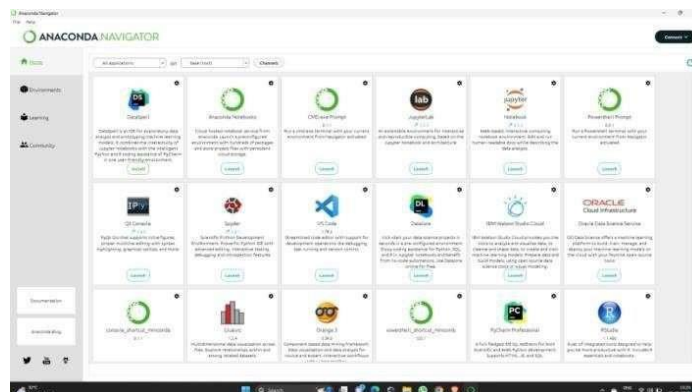
The installer will extract and install all the necessary files. This may take a few minutes.



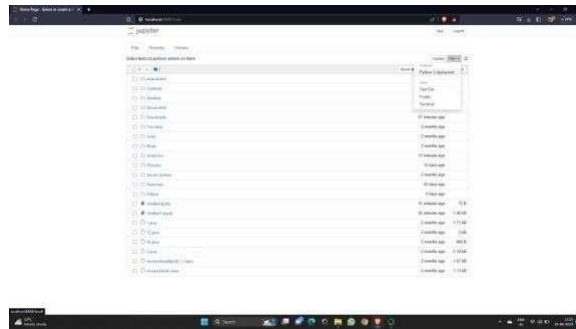
Step6: After the installation completes, you can launch Anaconda Navigator by clicking the “Next” button.



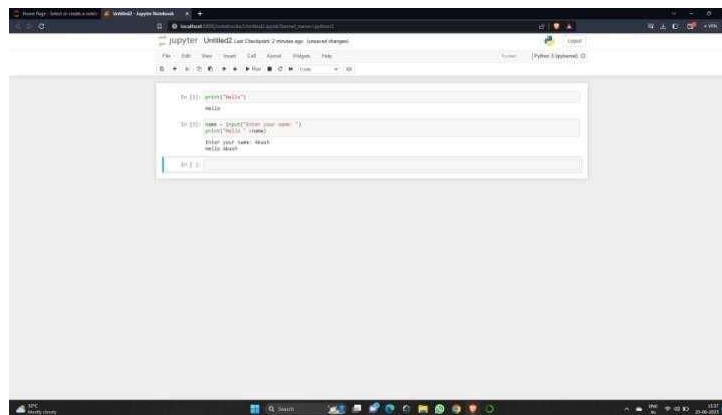
Step7: Open “Jupyter Notebook”.



Step8: Now click to “new”, then click on “Python3 (ipykernel)”.



Start Coding with python.

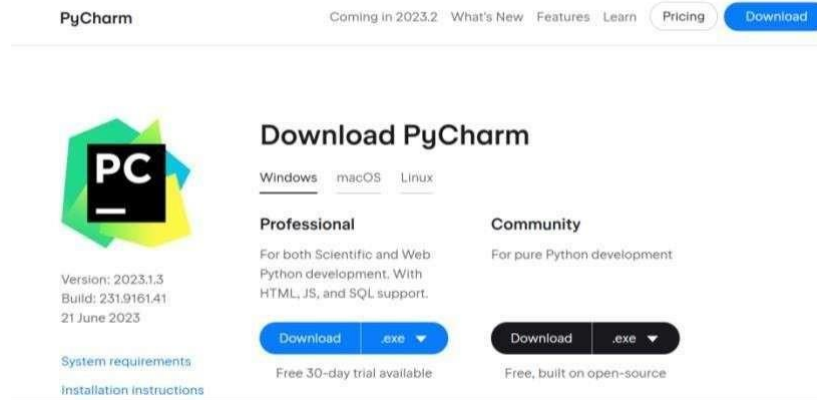


Installation Guide for PyCharm Introduction

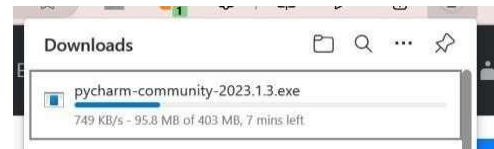
PyCharm is a widely used integrated development environment (IDE) for Python programming. Developed by JetBrains, it offers a range of features including an intelligent code editor, powerful debugging capabilities, and code refactoring tools. With its user-friendly interface, PyCharm is a popular choice for Python developers to write, debug, and manage their code efficiently.

Step1: Go to the official PyCharm website at <https://www.jetbrains.com/pycharm/>.

Step 2: Choose the edition you want (Community or Professional) and click on the respective download button.



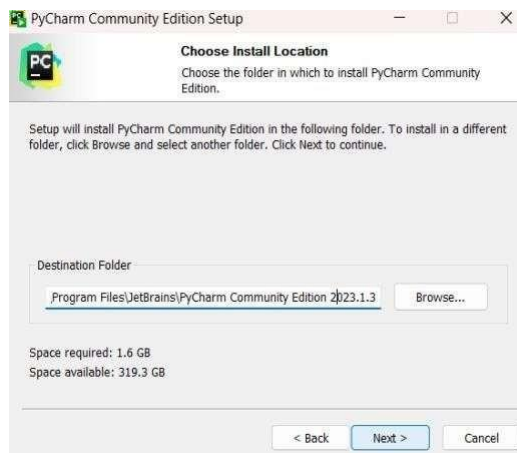
Step 3: Once the download is complete, locate the installer file.



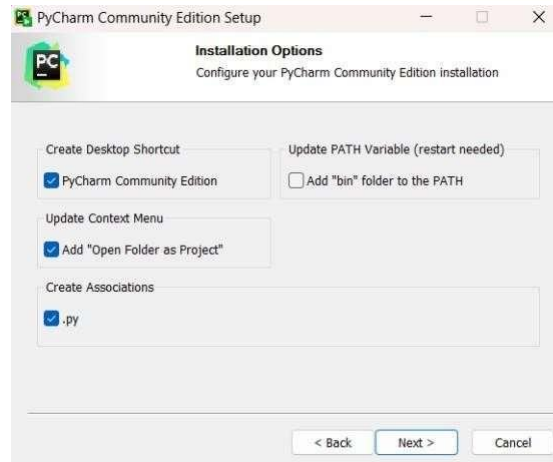
Double-click on the installer file to start the installation process.



Step 4: Choose the installation location (you can usually leave the default location as is).



Step 5: Choose Installation options



Step6: Click on the "Install" button to begin the installation. Wait

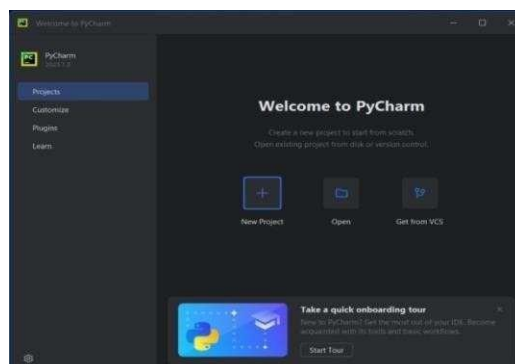
for the installation to complete.

Step 7: After the installation, choose whether to run PyCharm immediately or exit and launch it later from the Start menu.



Step8: Once the installation is complete, go to search menu and open PyCharm.

Now by clicking on "New Project" it will create a folder where you code.



Practical-2

Aim:-1) Create a list named "Subjects" by inserting 10 subjects into it through any loop and create a list "ElectiveSubjects" with 5 subjects through direct initialization. Extend list "Subject" by another list "ElectiveSubjects". Append 3 duplicate subjects into "Subject" list. Find the index of first occurrence of that duplicate value and then remove all the occurrences of that specific subject through loop. Define function remove_range(i1,i2) to remove range of element from i1 to i2 through del keyword and return the resultant list. Pop 5th element after reversing and sorting your list. Count total elements in your list and finally clear the list. Which of the above operations can be performed directly? Which of the above operations cannot be performed directly on Tuple and why? Update and remove specific item from the tuple by converting it into list.

Create a dictionary named "StudentsData" with 5 students and id_no, name and marks as the key values. Provide the separate list of all the keys and values. Add details of one more student. Retrieve value corresponding to specific key through get method. Define a function update_detail(k) by looping over keys to search for specific key 'k' whose details to be updated and then update it with new details and return updated dictionary. If specific detail is not available in list print appropriate message. Convert dictionary's keys into a list by looping through keys and appending it to the other list. Convert dictionary values into a list through list constructor. Count and display total number of students in the dictionary. Remove all the details from the dictionary. Define a dictionary named "exam_data_array" with 4 keys, namely 'name', 'score', 'attempts' and 'qualify'. Values for each of these 4 keys will be an array with 5 elements. By creating a dictionary named "exam_data_list" with 5 lists and each list stores all 4 key-value pairs for single student.

→code:-

```
subjects=[]
for i in range(10):
    subjects.append(input())
print(subjects)
elective_subjects=["PM","DM","FLEA","CG","ADJ"]
print("elective subjects are :",elective_subjects)
subjects.extend(elective_subjects)
print("after extending subjects by elective subjects:",subjects)
subjects.append("AI")
subjects.append("AI")
subjects.append("AI")
dup_subjects="AI"
print("after appending duplicates subjects:",subjects)
first_index=subjects.index(dup_subjects)
while dup_subjects in subjects:
    subjects.remove(dup_subjects)
print("after removing duplicates subjects:",subjects)
def remove_range(i1,i2):
    del subjects[i1:i2]
    return subjects
subjects=remove_range(2,5)
print("after removing elements from range 2 to 5:",subjects)
subjects.sort()
subjects.reverse()
```

```
print("aftersortigandreversingthelist",subjects)
popped_ele=subjects.pop(4)
print("poppedelemnet:",popped_ele)
print("afterpoppingthe5thelemnetlist:",subjects)
print("total number of list:",len(subjects))
subjects.clear()
print("afterclearthelist",subjects)
print("\nTuple Limitations:")
print("Youcanaccesselementsandcount/indexintupledirectly.")
print("Butyoucannotadd,remove,orupdatetuplevaluesdirectlybecausetuplesare immutable.") #
Update tuple (convert to list and back)
my_tuple=("Maths","Science","History","English")
temp_list = list(my_tuple)
temp_list[2] ="Geography" # updating
temp_list.remove("Science")#removing
my_tuple = tuple(temp_list)
print("Updated Tuple:", my_tuple)
```

solution:-

```
PS C:\Users\HP\Downloads> python subject18.py
DAA
DBMS
DS
DF
WD
PDS
LAVC
PSNM
CALCULUS
EES
['DAA', 'DBMS', 'DS', 'DF', 'WD', 'PDS', 'LAVC', 'PSNM', 'CALCULUS', 'EES']
elective subjects are : ['PM', 'DM', 'FLEA', 'CG', 'ADJ']
after extending subjects by elective subjects : ['DAA', 'DBMS', 'DS', 'DF', 'WD', 'PDS', 'LAVC', 'PSNM', 'CALCULUS', 'EES', 'PM', 'DM', 'FLEA', 'CG', 'ADJ']
after appending duplicate subjects: ['DAA', 'DBMS', 'DS', 'DF', 'WD', 'PDS', 'LAVC', 'PSNM', 'CALCULUS', 'EES', 'PM', 'DM', 'FLEA', 'CG', 'ADJ', 'AI', 'AI', 'AI']
after removing duplicate subjects: ['DAA', 'DBMS', 'DS', 'DF', 'WD', 'PDS', 'LAVC', 'PSNM', 'CALCULUS', 'EES', 'PM', 'DM', 'FLEA', 'CG', 'ADJ']
after removing elements from range 2 to 5: ['DAA', 'DBMS', 'PDS', 'LAVC', 'PSNM', 'CALCULUS', 'EES', 'PM', 'DM', 'FLEA', 'CG', 'ADJ']
after sortig and reversing the list ['PSNM', 'PM', 'PDS', 'LAVC', 'FLEA', 'EES', 'DM', 'DBMS', 'DAA', 'CG', 'CALCULUS', 'ADJ']
popped elemnet: FLEA
after popping the 5th elemnet list: ['PSNM', 'PM', 'PDS', 'LAVC', 'EES', 'DM', 'DBMS', 'DAA', 'CG', 'CALCULUS', 'ADJ']
total number of list: 11
after clear the list []

Tuple Limitations:
You can access elements and count/index in tuple directly.
But you cannot add, remove, or update tuple values directly because tuples are immutable.
Updated Tuple: ('Maths', 'Geography', 'English')
PS C:\Users\HP\Downloads>
```


Aim:-2) Create a dictionary named “Students Data” with 5 students and id_no, name and marks as the key values. Provide the separate list of all the keys and values. Add details of one more student. Retrieve value corresponding to specific key through get method. Define a function update detail(k) by looping over keys to search for specific key ‘k’ whose details to be updated and then update it with new details and return updated dictionary. If specific detail is not available in list print appropriate message. Convert dictionary’s keys into a list by looping through keys and appending it to the other list. Convert dictionary values into list through list constructor. Count and display total number of students in the dictionary. Remove all the details from the dictionary. Define a dictionary named “exam_data_array” with 4 keys, namely ‘name’, ‘score’, ‘attempts’ and ‘qualify’. Values for each of these 4 keys will be an 1Darray with 5 elements. by creating a dictionary named “exam_data_list” with 5 list and each list stores all 4 key-value pairs for single student.

→code:-

```
Students_Data = {
    1: {'id_no': 'S001', 'name': 'Alice', 'marks': 85},
    2: {'id_no': 'S002', 'name': 'Bob', 'marks': 78},
    3: {'id_no': 'S003', 'name': 'Charlie', 'marks': 92},
    4: {'id_no': 'S004', 'name': 'Diana', 'marks': 88},
    5: {'id_no': 'S005', 'name': 'Edward', 'marks': 74}
}
keys_list = list(Students_Data.keys())
values_list = list(Students_Data.values())
print("Keys List:", keys_list)
print("Values List:", values_list)
Students_Data[6] = {'id_no': 'S006', 'name': 'Fiona', 'marks': 90}
student_id = 3
student_details = Students_Data.get(student_id, "Student not found")
print(f'Details of student with ID {student_id}:', student_details)
def update_detail(k, new_details):
    if k in Students_Data:
        Students_Data[k].update(new_details)
        return Students_Data
    else:
        print(f'Student with ID {k} not found.')
        return Students_Data
new_details = {'name': 'Charles', 'marks': 95}
updated_data = update_detail(3, new_details)
print("Updated Data:", updated_data)
keys_list = [key for key in Students_Data.keys()]
values_list = list(Students_Data.values())
print("Keys List:", keys_list)
print("Values List:", values_list)
total_students = len(Students_Data)
print("Total Number of Students:", total_students)
Students_Data.clear()
print("Students Data after clearing:", Students_Data)
exam_data_array = {
    'name': ['Alice', 'Bob', 'Charlie', 'Diana', 'Edward'],
    'score': [85, 78, 92, 88, 74],
    'attempts': [1, 2, 1, 2, 3],
```

```
'qualify': ['Yes', 'No', 'Yes', 'Yes', 'No']
}
exam_data_list = [
    {
        'name': exam_data_array['name'][i],
        'score': exam_data_array['score'][i],
        'attempts': exam_data_array['attempts'][i],
        'qualify': exam_data_array['qualify'][i]
    }
    for i in range(len(exam_data_array['name']))
]
print("Exam Data List:", exam_data_list)
```

solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/k18.py
Keys List: [1, 2, 3, 4, 5]
Values List: [{ 'id_no': 'S001', 'name': 'Alice', 'marks': 85}, { 'id_no': 'S002', 'name': 'Bob', 'marks': 78}, { 'id_no': 'S003', 'name': 'Charlie', 'marks': 92}, { 'id_no': 'S004', 'name': 'Diana', 'marks': 88}, { 'id_no': 'S005', 'name': 'Edward', 'marks': 74}]
Details of student with ID 3: { 'id_no': 'S003', 'name': 'Charlie', 'marks': 92}
Updated Data: {1: { 'id_no': 'S001', 'name': 'Alice', 'marks': 85}, 2: { 'id_no': 'S002', 'name': 'Bob', 'marks': 78}, 3: { 'id_no': 'S003', 'name': 'Charles', 'marks': 95}, 4: { 'id_no': 'S004', 'name': 'Diana', 'marks': 88}, 5: { 'id_no': 'S005', 'name': 'Edward', 'marks': 74}, 6: { 'id_no': 'S006', 'name': 'Fiona', 'marks': 90}}
Keys List: [1, 2, 3, 4, 5, 6]
Values List: [{ 'id_no': 'S001', 'name': 'Alice', 'marks': 85}, { 'id_no': 'S002', 'name': 'Bob', 'marks': 78}, { 'id_no': 'S003', 'name': 'Charles', 'marks': 95}, { 'id_no': 'S004', 'name': 'Diana', 'marks': 88}, { 'id_no': 'S005', 'name': 'Edward', 'marks': 74}, { 'id_no': 'S006', 'name': 'Fiona', 'marks': 90}]
Total Number of Students: 6
Students Data after clearing: {}
Exam Data List: [{ 'name': 'Alice', 'score': 85, 'attempts': 1, 'qualify': 'Yes'}, { 'name': 'Bob', 'score': 78, 'attempts': 2, 'qualify': 'No'}, { 'name': 'Charlie', 'score': 92, 'attempts': 1, 'qualify': 'Yes'}, { 'name': 'Diana', 'score': 88, 'attempts': 2, 'qualify': 'Yes'}, { 'name': 'Edward', 'score': 74, 'attempts': 3, 'qualify': 'No'}]
PS C:\Users\HP>
```

Practical-3

Aim:-1) Do the slicing of a given String to generate various substring by passing different index (like positive index, negative index, end index > string length, entire string), split this string into chunks of length 3 using list comprehension, split the string with specific character, iterate over the words of string. Apply trim, toupper, tolower, replace string and character, title, join and other operations on String.

→code:-

```
print("=== STRING OPERATIONS ===")
sample = "HelloPythonWorld"
print("\n--- String Slicing ---")
print("Original String:", sample)
print("Positive index [0:5]:", sample[0:5])
print("Negative index [-5:]:", sample[-5:])
print("End index > length [0:50]:", sample[0:50])
print("Entire string [:]:", sample[:])
print("\n--- Chunk of Length 3 ---")
chunks = [sample[i:i+3] for i in range(0, len(sample), 3)]
print("Chunks:", chunks)
print("\n--- Split with Specific Character ---")
split_string = "Hello-World-Python"
print("Split on '-':", split_string.split('-'))
print("\n--- Iterate over Words ---")
sentence = "Python is powerful"
for word in sentence.split():
    print("Word:", word)
print("\n--- String Methods ---")
str1 = " Welcome to Python "
print("Original:", repr(str1))
print("Trim:", repr(str1.strip()))
print("Upper:", str1.upper())
print("Lower:", str1.lower())
print("Replace 'Python' with 'Programming':", str1.replace("Python", "Programming"))
print("Title Case:", str1.title())
print("Join with '-':", "-".join(["Join", "These", "Words"]))
```

solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/k18.py
=== STRING OPERATIONS ===

--- String Slicing ---
Original String: HelloPythonWorld
Positive index [0:5]: Hello
Negative index [-5:]: World
End index > length [0:50]: HelloPythonWorld
Entire string [:]: HelloPythonWorld

--- Chunk of Length 3 ---
Chunks: ['Hel', 'loP', 'yth', 'onW', 'orl', 'd']

--- Split with Specific Character ---
Split on '-': ['Hello', 'World', 'Python']

--- Iterate over Words ---
Word: Python
Word: is
Word: powerful

--- String Methods ---
Original: ' Welcome to Python '
Trim: 'Welcome to Python'
Upper:  WELCOME TO PYTHON
Lower:  welcome to python
Replace 'Python' with 'Programming':  Welcome to Programming
Title Case:  Welcome To Python
Join with '-': Join-These-Words
PS C:\Users\HP>
```

Aim:-2) Perform add, union, intersection, difference, symmetric_difference, union, intersection_update, symmetric_difference_update, difference_update, discard, issubset, issuperset, isdisjoint, remove, pop and clear operations on Set. Import array module in python and perform all operations available in the module.

Code:-

```
print("\n=== SET OPERATIONS ===")
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
print("Set1:", set1)
print("Set2:", set2)
```

```
# Add
set1.add(6)
print("After add(6):", set1)
# Union
print("Union:", set1.union(set2))
# Intersection
print("Intersection:", set1.intersection(set2))
# Difference
print("Difference (set1 - set2):", set1.difference(set2))
# Symmetric Difference
print("Symmetric Difference:", set1.symmetric_difference(set2))
# intersection_update
set1_temp = set1.copy()
set1_temp.intersection_update(set2)
print("After intersection_update:", set1_temp)
# symmetric_difference_update
set1_temp = set1.copy()
set1_temp.symmetric_difference_update(set2)
print("After symmetric_difference_update:", set1_temp)
# difference_update
set1_temp = set1.copy()
set1_temp.difference_update(set2)
print("After difference_update:", set1_temp)
# discard
set1.discard(3)
print("After discard(3):", set1)
# remove (raises error if not present)
set1.remove(2)
print("After remove(2):", set1)
# pop (removes arbitrary element)
popped = set1.pop()
print("After pop():", set1, "| Popped:", popped)
# issubset
print("Is subset:", {4, 5}.issubset(set2))
# issuperset
print("Is superset:", set2.issuperset({4, 5}))
# isdisjoint
print("Is disjoint (set1 & set2):", set1.isdisjoint(set2))
# clear
set1.clear()
print("After clear():", set1)
```


solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/k18.py

=== SET OPERATIONS ===
Set1: {1, 2, 3, 4, 5}
Set2: {4, 5, 6, 7, 8}
After add(6): {1, 2, 3, 4, 5, 6}
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5, 6}
Difference (set1 - set2): {1, 2, 3}
Symmetric Difference: {1, 2, 3, 7, 8}
After intersection_update: {4, 5, 6}
After symmetric_difference_update: {1, 2, 3, 7, 8}
After difference_update: {1, 2, 3}
After discard(3): {1, 2, 4, 5, 6}
After remove(2): {1, 4, 5, 6}
After pop(): {4, 5, 6} | Popped: 1
Is subset: True
Is superset: True
Is disjoint (set1 & set2): False
After clear(): set()
PS C:\Users\HP>
```

Aim:-3) Import array module in python and perform all operations available in the module.

Code:-

```
# --- ARRAY OPERATIONS ---
print("\n=== ARRAY OPERATIONS ===")
import array
# Creating an array of integers
arr = array.array('i', [1, 2, 3, 4, 5])
print("Original Array:", arr)
# Append
arr.append(6)
print("After append(6):", arr)
# Insert
arr.insert(2, 10)
print("After insert(2, 10):", arr)
# Pop
arr.pop()
print("After pop():", arr)
# Remove
arr.remove(3)
print("After remove(3):", arr)
# Index
```



```
print("Index of 10:", arr.index(10))
# Reverse
arr.reverse()
print("After reverse():", arr)
# Buffer info
print("Buffer info:", arr.buffer_info())
# Count
print("Count of 2:", arr.count(2))
# Extend
arr.extend([7, 8, 9])
print("After extend([7,8,9]):", arr)
# Convert to list
print("Array to list:", arr.tolist())
```

Solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/k18.py

=== ARRAY OPERATIONS ===
Original Array: array('i', [1, 2, 3, 4, 5])
After append(6): array('i', [1, 2, 3, 4, 5, 6])
After insert(2, 10): array('i', [1, 2, 10, 3, 4, 5, 6])
After pop(): array('i', [1, 2, 10, 3, 4, 5])
After remove(3): array('i', [1, 2, 10, 4, 5])
Index of 10: 2
After reverse(): array('i', [5, 4, 10, 2, 1])
Buffer info: (1540051132240, 5)
Count of 2: 1
After extend([7,8,9]): array('i', [5, 4, 10, 2, 1, 7, 8, 9])
Array to list: [5, 4, 10, 2, 1, 7, 8, 9]
PS C:\Users\HP>
```

Practical-4

Aim:-Download “heart_2020_cleaned.csv” dataset from “<https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease>” and perform all the descriptive statistics on above dataset using statistics module of python and scipy.stats package (Measures of central tendency, measure of dispersion/variation, measure of location, measure of shape and symmetry).

Code:-

```
import pandas as pd
import statistics
import scipy.stats as stats
# Load the dataset
heart_data = pd.read_csv('C:/Users/YourUsername/Documents/heart_2020_cleaned.csv')
# Display basic info about the dataset
print("Dataset Info:")
print(heart_data.info())
# List of numerical columns
numerical_columns = heart_data.select_dtypes(include='number').columns
# Measures of central tendency, dispersion, location, and shape/symmetry
descriptive_stats = {}
for col in numerical_columns:
    data = heart_data[col].dropna()
    descriptive_stats[col] = {
        'mean': statistics.mean(data),
        'median': statistics.median(data),
        'mode': statistics.mode(data),
        'variance': statistics.variance(data),
        'stdev': statistics.stdev(data),
        'min': min(data),
        'max': max(data),
        'range': max(data) - min(data),
        'skewness': stats.skew(data),
        'kurtosis': stats.kurtosis(data)
    }
# Display the descriptive statistics
for col, stats in descriptive_stats.items():
    print(f"\nDescriptive Statistics for {col}:")
    for stat, value in stats.items():
        print(f'{stat}: {value}')
```

Solution:-

```
PS C:\Users\admin\Downloads> python prac4.py
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319795 entries, 0 to 319794
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  -
0   HeartDisease           319795 non-null object
1   BMI                    319795 non-null float64
2   Smoking                319795 non-null object
3   AlcoholDrinking        319795 non-null object
4   Stroke                 319795 non-null object
5   PhysicalHealth          319795 non-null float64
6   MentalHealth            319795 non-null float64
7   DiffWalking            319795 non-null object
8   Sex                    319795 non-null object
9   AgeCategory            319795 non-null object
10  Race                   319795 non-null object
11  Diabetic                319795 non-null object
12  PhysicalActivity        319795 non-null object
13  GenHealth               319795 non-null object
14  SleepTime               319795 non-null float64
15  Asthma                  319795 non-null object
16  KidneyDisease           319795 non-null object
17  SkinCancer              319795 non-null object
dtypes: float64(4), object(14)
memory usage: 43.9+ MB
None
```

```
Descriptive Statistics for BMI:
mean: 28.325398520927468
median: 27.34
mode: 26.63
variance: 40.400009758424176
stdev: 6.356100200470739
min: 12.02
max: 94.85
range: 82.83
skewness: 1.3324243931174056
kurtosis: 3.889963770419131
variance: 63.21601862569516
stdev: 7.950850182571369
min: 0.0
max: 30.0
range: 30.0
skewness: 2.6039610482829407
kurtosis: 5.5283444393183565
```

```
Descriptive Statistics for MentalHealth:
mean: 3.898366140808956
median: 0.0
mode: 0.0
variance: 63.28576738872078
stdev: 7.955235218943609
min: 0.0
max: 30.0
range: 30.0
skewness: 2.331100615031879
kurtosis: 4.403849003140329
```

```
Descriptive Statistics for SleepTime:
mean: 7.097074688472302
median: 7.0
mode: 7.0
variance: 2.0621162791392766
stdev: 1.4360070609642825
min: 1.0
max: 24.0
range: 23.0
skewness: 0.6790314357818188
kurtosis: 7.854726994730127
```

Practical-5

Aim:-1) Write following program on Pandas DataFrame:

- Create an array “rank” with 5 element(rank1,rank2,...,rank5). Create and display a DataFrame “exam” from a specified dictionary “exam_data_array” with “rank” as label. Also display a summary of basic information and its data. Perform following operations on DataFrame “exam” :
- Select the rows where the score is between 15 and 20 (inclusive).
- Sort the data first by “score” in ascending order , then by “name” in descending order.
- Replace the 'yes' and 'no' values from column “qualify” with True and False.
- Display specified columns(columns: 2 and 4) and rows(row: 1,3 and 5).
- Select the rows where number of attempts in the examination is less than 2 and score greater than 15.
- Change the name 'James' to 'Suresh' in “name” column of the data frame.
- Calculate the sum of the examination attempts by the students
- Append one row.
- Insert a new column “exam_name” and then Delete the ”exam_name” column.
- Convert a NumPy array, dictionary and first column of a DataFrame to a series.

Code:-

```
import pandas as pd
import numpy as np
# Step 1: Create an array “rank” with 5 elements
rank = ['rank1', 'rank2', 'rank3', 'rank4', 'rank5']
# Step 2: Create and display a DataFrame “exam” from a specified dictionary “exam_data_array” with “rank” as label
exam_data_array = {
    'name': ['Annie', 'James', 'Catherine', 'Michael', 'Laura'],
    'score': [18, 20, 16, 19, 17],
    'attempts': [1, 3, 2, 1, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'yes']
}
exam = pd.DataFrame(exam_data_array, index=rank)
# Display the DataFrame
print("DataFrame 'exam':\n", exam)
# Display a summary of basic information and its data
print("\nBasic Information Summary:")
print(exam.info())
print("\nData Summary:")
print(exam.describe(include='all'))
```

```
# Step 3: Select the rows where the score is between 15 and 20 (inclusive)
filtered_exam = exam[(exam['score'] >= 15) & (exam['score'] <= 20)]
print("\nRows where the score is between 15 and 20:\n", filtered_exam)

# Step 4: Sort the data first by "score" in ascending order, then by "name" in descending order
sorted_exam = exam.sort_values(by=['score', 'name'], ascending=[True, False])
print("\nData sorted by 'score' (ascending) and 'name' (descending):\n", sorted_exam)

# Step 5: Replace the 'yes' and 'no' values from the "qualify" column with True and False
exam['qualify'] = exam['qualify'].map({'yes': True, 'no': False})
print("\nDataFrame after replacing 'yes'/'no' in 'qualify' with True/False:\n", exam)

# Step 6: Display specified columns (columns: 2 and 4) and rows (row: 1, 3, and 5)
selected_data = exam.iloc[[0, 2, 4], [1, 3]]
print("\nSpecified columns (2 and 4) and rows (1, 3, and 5):\n", selected_data)

# Step 7: Select the rows where number of attempts in the examination is less than 2 and score greater than 15
filtered_attempts = exam[(exam['attempts'] < 2) & (exam['score'] > 15)]
print("\nRows where attempts < 2 and score > 15:\n", filtered_attempts)

# Step 8: Change the name 'James' to 'Suresh' in the "name" column of the DataFrame
exam.loc[exam['name'] == 'James', 'name'] = 'Suresh'
print("\nDataFrame after changing 'James' to 'Suresh':\n", exam)

# Step 9: Calculate the sum of the examination attempts by the students
total_attempts = exam['attempts'].sum()
print("\nSum of examination attempts by the students:", total_attempts)

# Step 10: Append one row to the DataFrame
new_row = pd.DataFrame({'name': ['Kevin'], 'score': [15], 'attempts': [1], 'qualify': [True]}, index=['rank6'])
exam = exam.append(new_row)
print("\nDataFrame after appending one row:\n", exam)

# Step 11: Insert a new column "exam_name" and then delete the "exam_name" column
exam['exam_name'] = 'Final Exam'
print("\nDataFrame after inserting 'exam_name' column:\n", exam)
exam.drop('exam_name', axis=1, inplace=True)
print("\nDataFrame after deleting 'exam_name' column:\n", exam)

# Step 12: Convert a NumPy array, dictionary, and first column of a DataFrame to a series
# Converting a NumPy array to a series
numpy_array = np.array([1, 2, 3, 4, 5])
numpy_series = pd.Series(numpy_array)
print("\nNumPy array converted to Series:\n", numpy_series)
```



```
# Converting a dictionary to a series
dictionary = {'a': 10, 'b': 20, 'c': 30}
dictionary_series=pd.Series(dictionary)
print("\nDictionary converted to Series:\n", dictionary_series)

# Converting the first column of the DataFrame to a series
name_series= exam['name']
print("\nFirst column ('name') of DataFrame converted to Series:\n", name_series)
```

Solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/Prac5_1.py
DataFrame 'exam':
   name  score  attempts  qualify
rank1  Annie    18         1     yes
rank2  James    20         3     no
rank3  Catherine  16         2     yes
rank4  Michael   19         1     no
rank5   Laura    17         1     yes

Basic Information Summary:
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, rank1 to rank5
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   name         5 non-null      object
1   score        5 non-null      int64
2   attempts     5 non-null      int64
3   qualify      5 non-null      object
dtypes: int64(2), object(2)
memory usage: 200.0+ bytes
None

Data Summary:
   name  score  attempts  qualify
count    5  5.000000  5.000000    5
unique    5      NaN      NaN     2
top  Annie      NaN      NaN    yes
freq     1      NaN      NaN     3
mean   NaN  18.000000  1.600000   NaN
std    NaN   1.581139  0.894427   NaN
min    NaN  16.000000  1.000000   NaN
25%    NaN  17.000000  1.000000   NaN
50%    NaN  18.000000  1.000000   NaN
75%    NaN  19.000000  2.000000   NaN
max    NaN  20.000000  3.000000   NaN
```


Rows where the score is between 15 and 20:

	name	score	attempts	qualify
rank1	Annie	18	1	yes
rank2	James	20	3	no
rank3	Catherine	16	2	yes
rank4	Michael	19	1	no
rank5	Laura	17	1	yes

Data sorted by 'score' (ascending) and 'name' (descending):

	name	score	attempts	qualify
rank3	Catherine	16	2	yes
rank5	Laura	17	1	yes
rank1	Annie	18	1	yes
rank4	Michael	19	1	no
rank2	James	20	3	no

DataFrame after replacing 'yes'/'no' in 'qualify' with True/False:

	name	score	attempts	qualify
rank1	Annie	18	1	True
rank2	James	20	3	False
rank3	Catherine	16	2	True
rank4	Michael	19	1	False
rank5	Laura	17	1	True

Specified columns (2 and 4) and rows (1, 3, and 5):

	score	qualify
rank1	18	True
rank3	16	True
rank5	17	True

Rows where attempts < 2 and score > 15:

	name	score	attempts	qualify
rank1	Annie	18	1	True
rank4	Michael	19	1	False
rank5	Laura	17	1	True

DataFrame after changing 'James' to 'Suresh':

	name	score	attempts	qualify
rank1	Annie	18	1	True
rank2	Suresh	20	3	False
rank3	Catherine	16	2	True
rank4	Michael	19	1	False
rank5	Laura	17	1	True

Sum of examination attempts by the students: 8

DataFrame after appending one row:

	name	score	attempts	qualify
rank1	Annie	18	1	True
rank2	Suresh	20	3	False
rank3	Catherine	16	2	True
rank4	Michael	19	1	False
rank5	Laura	17	1	True
rank6	Kevin	15	1	True

DataFrame after inserting 'exam_name' column:

	name	score	attempts	qualify	exam_name
rank1	Annie	18	1	True	Final Exam
rank2	Suresh	20	3	False	Final Exam
rank3	Catherine	16	2	True	Final Exam
rank4	Michael	19	1	False	Final Exam
rank5	Laura	17	1	True	Final Exam
rank6	Kevin	15	1	True	Final Exam

DataFrame after deleting 'exam_name' column:

	name	score	attempts	qualify
rank1	Annie	18	1	True
rank2	Suresh	20	3	False
rank3	Catherine	16	2	True
rank4	Michael	19	1	False
rank5	Laura	17	1	True
rank6	Kevin	15	1	True

NumPy array converted to Series:

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

Dictionary converted to Series:

```
a    10
b    20
c    30
dtype: int64
```

First column ('name') of DataFrame converted to Series:

```
rank1    Annie
rank2    Suresh
rank3    Catherine
rank4    Michael
rank5    Laura
rank6    Kevin
Name: name, dtype: object
PS C:\Users\HP>
```

Aim:-2) Write following program on NumPy Array:

- Create an array of all the even integers from 30 to 70.
- Create an array of 10 zeros, other with 10 ones, and one more with 10 fives.
- Create a vector of length 10 with values evenly distributed between 5 and 50.
- Create a 3x4 matrix filled with values from 10 to 21 and compute sum of all elements, sum of each column and sum of each row of a given array.
- Create a 3x4 array and find the missing data in the array.
- Calculate round, floor, ceiling, truncated and round (to the given number of decimals) of the input, element-wise of an array.
- Find the maximum and minimum value, median, Weighted average, mean, standard deviation, variance, covariance matrix, of a given flattened array, minimum and maximum value along the second axis. Create a structured array from given student name, height, class and their data types. Now sort by class, then height if class are equal.

Code:-

```
import numpy as np

# Step 1: Create an array of all the even integers from 30 to 70
even_integers = np.arange(30, 71, 2)
print("Array of even integers from 30 to 70:\n", even_integers)

# Step 2: Create an array of 10 zeros, another with 10 ones, and one more with 10 fives
zeros_array = np.zeros(10)
ones_array = np.ones(10)
fives_array = np.full(10, 5)
print("\nArray of 10 zeros:\n", zeros_array)
print("\nArray of 10 ones:\n", ones_array)
print("\nArray of 10 fives:\n", fives_array)

# Step 3: Create a vector of length 10 with values evenly distributed between 5 and 50
evenly_distributed = np.linspace(5, 50, 10)
print("\nVector of length 10 with values evenly distributed between 5 and 50:\n", evenly_distributed)

# Step 4: Create a 3x4 matrix filled with values from 10 to 21 and compute sums
matrix_3x4 = np.arange(10, 22).reshape(3, 4)
sum_all_elements = np.sum(matrix_3x4)
sum_each_column = np.sum(matrix_3x4, axis=0)
sum_each_row = np.sum(matrix_3x4, axis=1)
print("\n3x4 Matrix filled with values from 10 to 21:\n", matrix_3x4)
print("\nSum of all elements in the matrix:", sum_all_elements)
print("\nSum of each column:", sum_each_column)
print("\nSum of each row:", sum_each_row)

# Step 5: Create a 3x4 array and find the missing data in the array
array_with_nan = np.array([[1, 2, np.nan, 4], [5, np.nan, 7, 8], [9, 10, 11, np.nan]])
missing_data = np.isnan(array_with_nan)
```

```
print("\n3x4 Array with missing data:\n", array_with_nan)
print("\nMissing data in the array (True represents NaN):\n", missing_data)
# Step 6: Calculate round, floor, ceiling, truncated and round (to the given number of decimals) element-wise of an array
input_array = np.array([1.55, 2.75, -3.14, 4.99, -5.87])
rounded_array = np.round(input_array)
floor_array = np.floor(input_array)
ceiling_array = np.ceil(input_array)
truncated_array = np.trunc(input_array)
rounded_to_decimals = np.round(input_array, 1)
print("\nOriginal Array:\n", input_array)
print("\nRounded Array:\n", rounded_array)
print("\nFloor Array:\n", floor_array)
print("\nCeiling Array:\n", ceiling_array)
print("\nTruncated Array:\n", truncated_array)
print("\nRounded Array to 1 decimal place:\n", rounded_to_decimals)
# Step 7: Statistical calculations on a given flattened array
flattened_array = matrix_3x4.flatten()
max_value = np.max(flattened_array)
min_value = np.min(flattened_array)
median_value = np.median(flattened_array)
weighted_avg = np.average(flattened_array, weights=np.arange(1, len(flattened_array) + 1))
mean_value = np.mean(flattened_array)
std_deviation = np.std(flattened_array)
variance_value = np.var(flattened_array)
covariance_matrix = np.cov(flattened_array)
min_second_axis = np.min(matrix_3x4, axis=1)
max_second_axis = np.max(matrix_3x4, axis=1)
print("\nFlattened Array:\n", flattened_array)
print("\nMaximum value:", max_value)
print("\nMinimum value:", min_value)
print("\nMedian value:", median_value)
print("\nWeighted average:", weighted_avg)
print("\nMean value:", mean_value)
print("\nStandard Deviation:", std_deviation)
print("\nVariance:", variance_value)
print("\nCovariance matrix:\n", covariance_matrix)
```

```
print("\nMinimum value along the second axis:", min_second_axis)
print("\nMaximum value along the second axis:", max_second_axis)
# Step 8: Create a structured array from given student name, height, class, and their data types
student_data = np.array([('John', 5.9, 10), ('Alice', 5.5, 12), ('Bob', 6.1, 10), ('Mary', 5.7, 11)],
                        dtype=[('name', 'U10'), ('height', 'f4'), ('class', 'i4')])
# Sort by class, then height if class are equal
sorted_students = np.sort(student_data, order=['class', 'height'])
print("\nStructured Array sorted by 'class' and then 'height':\n", sorted_students)
```

Solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/Prac5_2.py
Array of even integers from 30 to 70:
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]

Array of 10 zeros:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Array of 10 ones:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

Array of 10 fives:
[5 5 5 5 5 5 5 5 5 5]

Vector of length 10 with values evenly distributed between 5 and 50:
[ 5. 10. 15. 20. 25. 30. 35. 40. 45. 50.]

3x4 Matrix filled with values from 10 to 21:
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]]

Sum of all elements in the matrix: 186

Sum of each column: [42 45 48 51]

Sum of each row: [46 62 78]

3x4 Array with missing data:
[[ 1.  2. nan  4.]
 [ 5. nan  7.  8.]
 [ 9. 10. 11. nan]]

Missing data in the array (True represents NaN):
[[False False  True False]
 [False  True False False]
 [False False False  True]]
```

```
Original Array:
[ 1.55  2.75 -3.14  4.99 -5.87]

Rounded Array:
[ 2.  3. -3.  5. -6.]

Floor Array:
[ 1.  2. -4.  4. -6.]

Ceiling Array:
[ 2.  3. -3.  5. -5.]

Truncated Array:
[ 1.  2. -3.  4. -5.]

Rounded Array to 1 decimal place:
[ 1.6  2.8 -3.1  5. -5.9]

Flattened Array:
[10 11 12 13 14 15 16 17 18 19 20 21]

Maximum value: 21

Minimum value: 10

Median value: 15.5

Weighted average: 17.333333333333332

Mean value: 15.5

Standard Deviation: 3.452052529534663

Variance: 11.916666666666666

Covariance matrix:
13.0
```

```
Minimum value along the second axis: [10 14 18]

Maximum value along the second axis: [13 17 21]

Structured Array sorted by 'class' and then 'height':
[('John', 5.9, 10) ('Bob', 6.1, 10) ('Mary', 5.7, 11) ('Alice', 5.5, 12)]
PS C:\Users\HP> |
```


Practical-6

Aim:-1) Write following python programs on BeautifulSoup Perform following operations on a HTML document.

- Find the title tag from a given html document.
- Count and retrieve all the paragraph tags and extract the text in the first paragraph tag.
- Find the text of the first tag and length of the text of the first tag.
- Find the href of the first tag.
- Find the first tag with a given attribute value in an html document.

Code:-

```
from bs4 import BeautifulSoup
# Example HTML document
html_doc = """
<!DOCTYPE html>
<html>
<head>
  <title>Sample HTML Document</title>
</head>
<body>
  <h1>Main Heading</h1>
  <p>This is the first paragraph.</p>
  <p>This is the second paragraph.</p>
  <a href="https://example.com">Click here</a>
  <h2>Sub Heading</h2>
</body>
</html>
"""

# Parse the HTML document with BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
# 1. Find the title tag from a given HTML document
title_tag = soup.title
print("Title tag:", title_tag)
# 2. Count and retrieve all the paragraph tags and extract the text in the first paragraph tag
paragraph_tags = soup.find_all('p')
print("\nNumber of paragraph tags:", len(paragraph_tags))
print("Text of the first paragraph tag:", paragraph_tags[0].get_text())
```

3. Find the text of the first <a> tag and length of the text of the first <h2> tag

```
first_a_tag_text = soup.find('a').get_text()
first_h2_tag_text = soup.find('h2').get_text()
print("\nText of the first <a> tag:", first_a_tag_text)
print("Length of the text of the first <h2> tag:", len(first_h2_tag_text))
```

4. Find the href of the first <a> tag

```
first_a_tag_href = soup.find('a')['href']
print("\nHref of the first <a> tag:", first_a_tag_href)
```

5. Find the first tag with a given attribute value in an HTML document

Example: Find the first tag with href="https://example.com"

```
first_tag_with_attribute = soup.find(attrs={"href": "https://example.com"})
print("\nFirst tag with href='https://example.com':", first_tag_with_attribute)
```

Solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/Prac6_1.py
Title tag: <title>Sample HTML Document</title>

Number of paragraph tags: 2
Text of the first paragraph tag: This is the first paragraph.

Text of the first <a> tag: Click here
Length of the text of the first <h2> tag: 11

Href of the first <a> tag: https://example.com

First tag with href='https://example.com': <a href="https://example.com">Click here</a>
PS C:\Users\HP> 
```


Aim:-2) Write a python program for MySQL Database connectivity(import sqlite3 module) Establish the connection with Education database named “Education” in SQLite , create a table named Student(with id_no, name, department, gander, total_marks) in Education database. Perform insert, update,select and delete operation on Student table.

Code:-

```
import sqlite3
# Step 1: Establish connection with the SQLite database
conn = sqlite3.connect('Education.db')
# Step 2: Create a cursor object to interact with the database
cursor = conn.cursor()
# Step 3: Create a table named "Student" in the "Education" database
cursor.execute("""CREATE TABLE IF NOT EXISTS Student (
    id_no INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    department TEXT,
    gender TEXT,
    total_marks INTEGER)""")
# Step 4: Insert records into the "Student" table
cursor.execute("INSERT INTO Student (id_no, name, department, gender, total_marks) VALUES (1, 'John Doe', 'Computer Science', 'Male', 85)")
cursor.execute("INSERT INTO Student (id_no, name, department, gender, total_marks) VALUES (2, 'Jane Doe', 'Electrical Engineering', 'Female', 90)")
cursor.execute("INSERT INTO Student (id_no, name, department, gender, total_marks) VALUES (3, 'Alice', 'Mechanical Engineering', 'Female', 75)")
conn.commit()
# Step 5: Select and display all records from the "Student" table
cursor.execute("SELECT * FROM Student")
students = cursor.fetchall()
print("\nStudents table after insertion:")
for student in students:
    print(student)
# Step 6: Update a record in the "Student" table
cursor.execute("UPDATE Student SET total_marks = 95 WHERE id_no = 2")
conn.commit()
# Step 7: Select and display the updated records
cursor.execute("SELECT * FROM Student")
updated_students = cursor.fetchall()
print("\nStudents table after update:")
for student in updated_students:
    print(student)
# Step 8: Delete a record from the "Student" table
cursor.execute("DELETE FROM Student WHERE id_no = 3")
conn.commit()
```

```
# Step 9: Select and display the records after deletion
cursor.execute("SELECT * FROM Student")
remaining_students = cursor.fetchall()
print("\nStudents table after deletion:")
for student in remaining_students:
    print(student)
# Step 10: Close the connection to the database
conn.close()
```

Solution:-

```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/Prac6_2.py

Students table after insertion:
(1, 'John Doe', 'Computer Science', 'Male', 85)
(2, 'Jane Doe', 'Electrical Engineering', 'Female', 90)
(3, 'Alice', 'Mechanical Engineering', 'Female', 75)

Students table after update:
(1, 'John Doe', 'Computer Science', 'Male', 85)
(2, 'Jane Doe', 'Electrical Engineering', 'Female', 95)
(3, 'Alice', 'Mechanical Engineering', 'Female', 75)

Students table after deletion:
(1, 'John Doe', 'Computer Science', 'Male', 85)
(2, 'Jane Doe', 'Electrical Engineering', 'Female', 95)
PS C:\Users\HP>
```

Practical-7

Aim:- Write a python program to download appropriate dataset and explore random variable, Probability mass function, Probability density function, Cumulative distribution function, Discrete probability distribution and continuous probability distribution using scipy.stats , rv_discrete class and rv_continuousclass .

Code:-

```
#install below lib first
#pip install numpy pandas scipy matplotlib seaborn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Step 1: Download a Dataset
# Let's use the "tips" dataset from seaborn, which is a built-in dataset.
data = sns.load_dataset('tips')
# Display the first few rows of the dataset
print("First few rows of the tips dataset:")
print(data.head())

# Step 2: Explore a Random Variable
# We'll explore the "total_bill" as a continuous random variable and "size" as a discrete random variable
total_bill = data['total_bill']
size = data['size']

# Step 3: Explore Probability Mass Function (PMF) for Discrete Distribution (using rv_discrete)
# Create a probability mass function for the 'size' variable
values, counts = np.unique(size, return_counts=True)
pmf = counts / len(size)
rv_discrete = stats.rv_discrete(name='size', values=(values, pmf))

# Plot the PMF
plt.bar(values, pmf, alpha=0.7)
plt.title('Probability Mass Function (PMF) for "size"')
plt.xlabel('Size')
plt.ylabel('Probability')
plt.show()
```

Step 4: Explore Probability Density Function (PDF) for Continuous Distribution (using rv_continuous)

Fit a normal distribution to the "total_bill" data

```
mu, std = stats.norm.fit(total_bill)
```

Create a PDF using the fitted normal distribution

```
rv_continuous = stats.norm(loc=mu, scale=std)
```

Plot the PDF

```
x = np.linspace(min(total_bill), max(total_bill), 100)
```

```
pdf = rv_continuous.pdf(x)
```

```
plt.plot(x, pdf, label=rf'Normal PDF\n$\mu={mu:.2f}, \sigma={std:.2f}$')
```

```
plt.hist(total_bill, density=True, bins=20, alpha=0.5, color='g', label='Histogram of total_bill')
```

```
plt.title('Probability Density Function (PDF) for "total_bill"')
```

```
plt.xlabel('Total Bill')
```

```
plt.ylabel('Density')
```

```
plt.legend()
```

```
plt.show()
```

Step 5: Explore Cumulative Distribution Function (CDF) for Continuous Distribution

```
cdf = rv_continuous.cdf(x)
```

```
plt.plot(x, cdf, label='CDF')
```

```
plt.title('Cumulative Distribution Function (CDF) for "total_bill"')
```

```
plt.xlabel('Total Bill')
```

```
plt.ylabel('Cumulative Probability')
```

```
plt.legend()
```

```
plt.show()
```

Step 6: Discrete Probability Distribution

We have already explored the PMF, which is the discrete probability distribution for "size".

Step 7: Continuous Probability Distribution

The PDF of "total_bill" is an example of a continuous probability distribution.

Additional Exploration: Descriptive Statistics

```
mean_bill = np.mean(total_bill)
```

```
median_bill = np.median(total_bill)
```

```
std_bill = np.std(total_bill)
```

```
var_bill = np.var(total_bill)
```

```
print(f'Mean of total_bill: {mean_bill:.2f}')
```

```
print(f'Median of total_bill: {median_bill:.2f}')
```

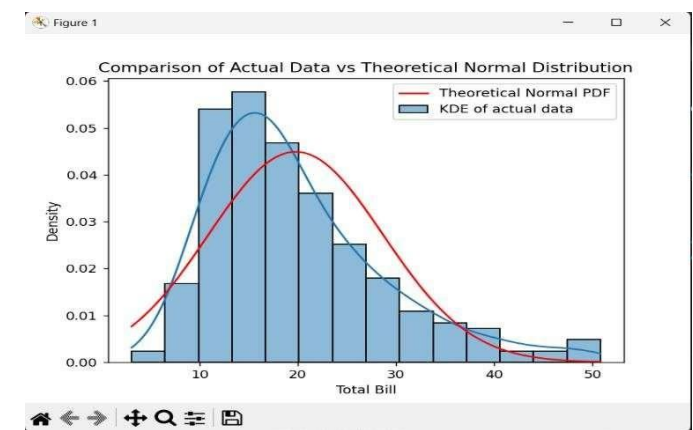
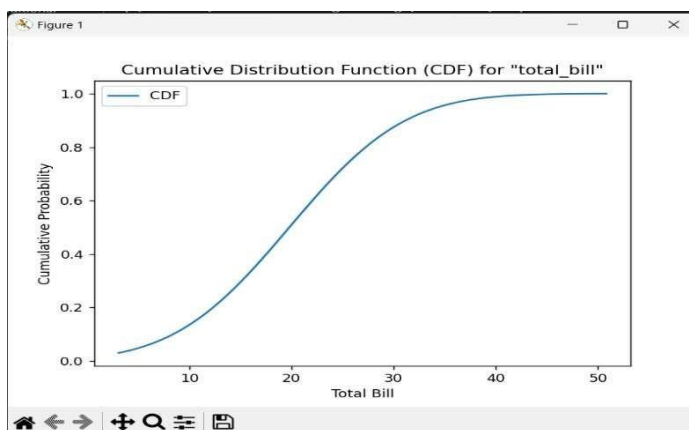
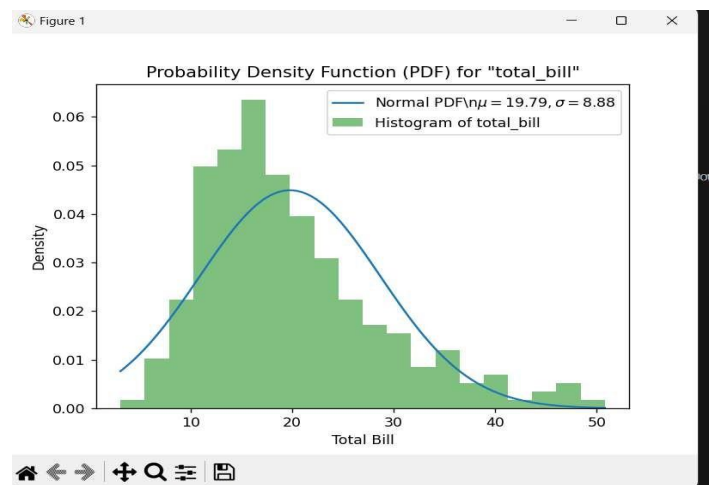
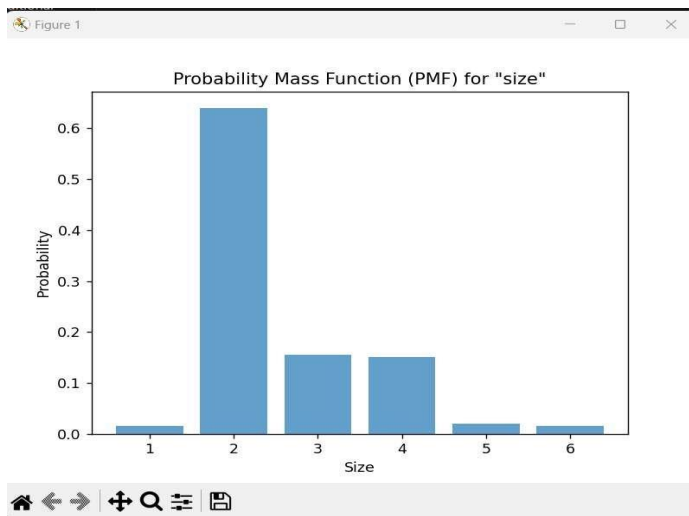
```
print(f'Standard Deviation of total_bill: {std_bill:.2f}')
```

```
print(f'Variance of total_bill: {var_bill:.2f}')
```

Step 8: Comparison of Actual vs Theoretical Distribution

```
sns.histplot(total_bill, kde=True, stat="density", label='KDE of actual data')
plt.plot(x, pdf, label='Theoretical Normal PDF', color='red')
plt.title('Comparison of Actual Data vs Theoretical Normal Distribution')
plt.xlabel('Total Bill')
plt.ylabel('Density')
plt.legend()
plt.show()
```

Solution:-



```
PS C:\Users\HP> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Downloads/Prac7.py
First few rows of the tips dataset:
  total_bill  tip  sex smoker  day  time  size
0    16.99   1.01 Female   No  Sun  Dinner    2
1    10.34   1.66  Male   No  Sun  Dinner    3
2    21.01   3.50  Male   No  Sun  Dinner    3
3    23.68   3.31  Male   No  Sun  Dinner    2
4    24.59   3.61 Female   No  Sun  Dinner    4
Mean of total_bill: 19.79
Median of total_bill: 17.80
Standard Deviation of total_bill: 8.88
Variance of total_bill: 78.93
PS C:\Users\HP>
```

Practical-8

Aim:- Write a python program to compute and explore normal distribution, central limit theorem, point estimate, interval estimation and hypothesis testing.

Code:-

```
#pip install numpy scipy matplotlib seaborn
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
# Step 1: Explore Normal Distribution
# Generate a normal distribution with a mean of 0 and standard deviation of 1
mu, sigma = 0, 1
data = np.random.normal(mu, sigma, 1000)
# Plot the normal distribution
sns.histplot(data, kde=True, stat="density")
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 100)
pdf = stats.norm.pdf(x, mu, sigma)
plt.plot(x, pdf, label=f'Normal PDF\n$\mu={mu}$, $\sigma={sigma}$', color='red')
plt.title('Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.show()
# Step 2: Explore Central Limit Theorem (CLT)
# Generate samples from a uniform distribution
uniform_data = np.random.uniform(0, 1, 1000)
# Draw samples of increasing size and calculate their means
means = []
sample_sizes = [5, 10, 30, 50, 100]
for size in sample_sizes:
    sample_means = [np.mean(np.random.choice(uniform_data, size=size)) for _ in range(1000)]
    means.append(sample_means)
```



```
# Plot histograms of sample means for different sample sizes
for i, size in enumerate(sample_sizes):
    sns.histplot(means[i], kde=True, stat="density")
    plt.title(f'Central Limit Theorem - Sample Size {size}')
    plt.xlabel('Sample Mean')
    plt.ylabel('Density')
    plt.show()

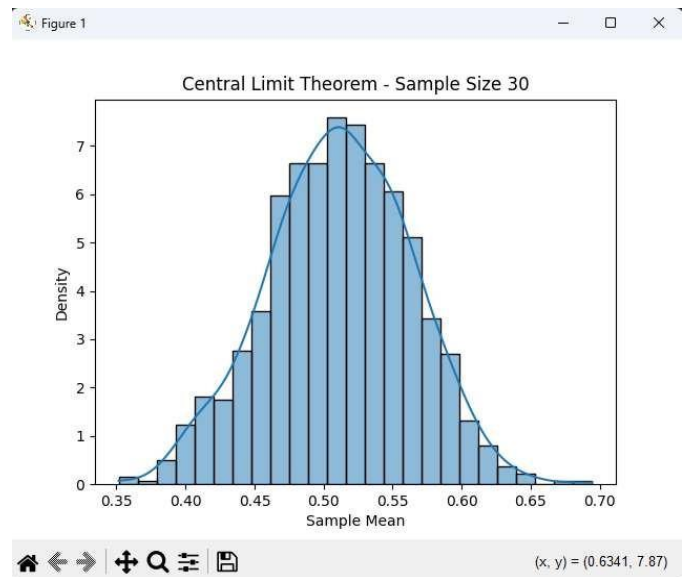
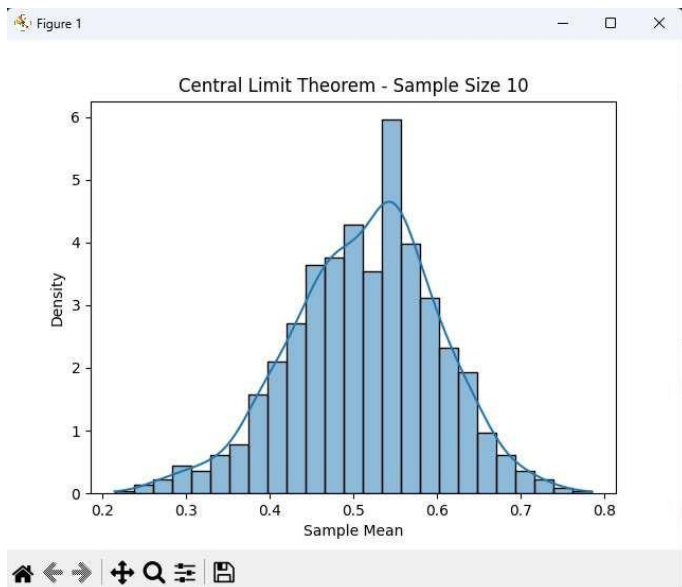
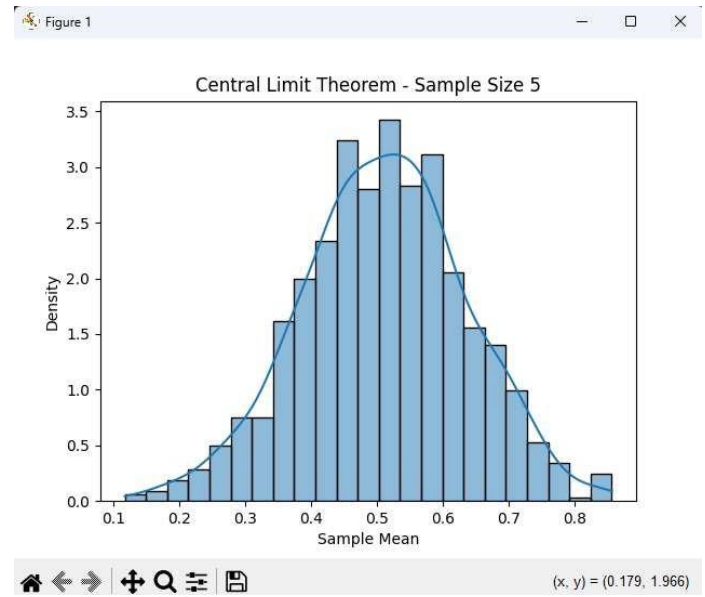
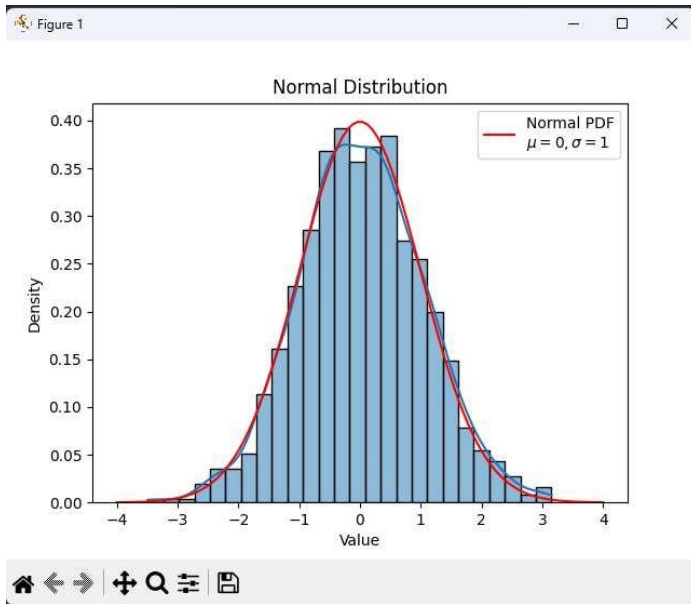
# Step 3: Point Estimation
# Point estimate for the population mean using sample mean
sample = np.random.normal(mu, sigma, 50)
sample_mean = np.mean(sample)
print(f'Point Estimate (Sample Mean): {sample_mean:.2f}')

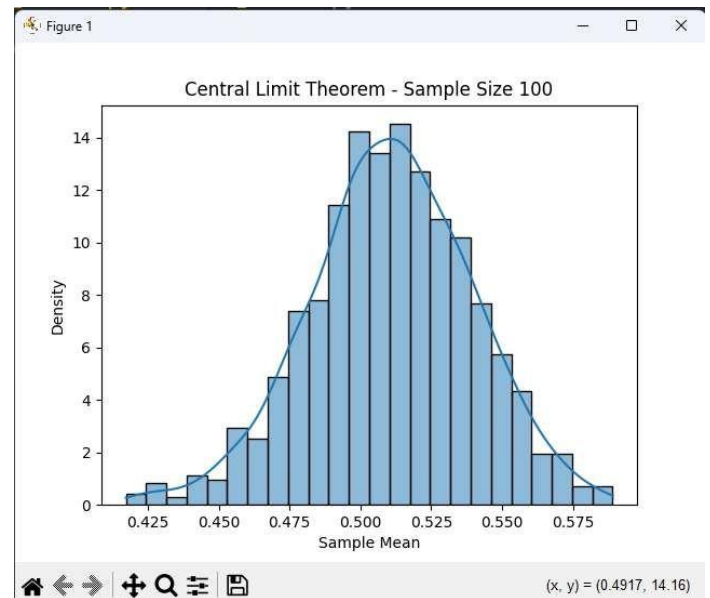
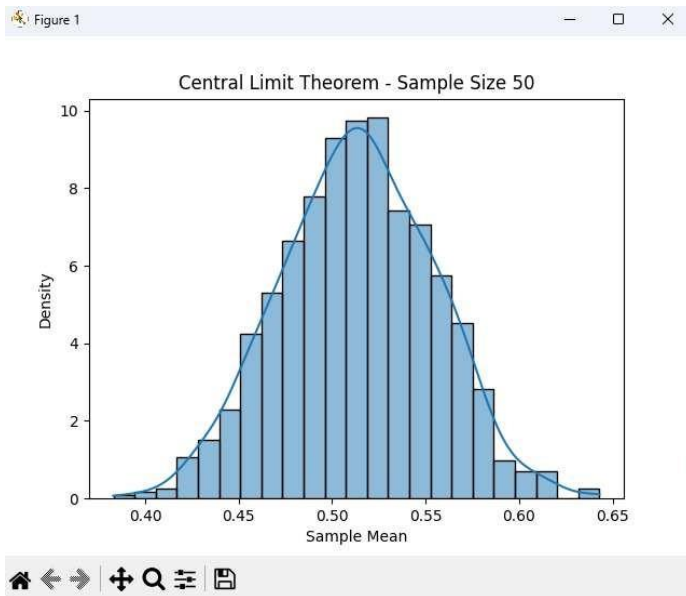
# Step 4: Interval Estimation (Confidence Interval)
# Calculate the 95% confidence interval for the sample mean
confidence_level = 0.95
ci = stats.norm.interval(confidence_level, loc=sample_mean, scale=sigma/np.sqrt(len(sample)))
print(f'95% Confidence Interval for the Mean: {ci}')

# Step 5: Hypothesis Testing
# Perform a one-sample t-test
# Null hypothesis: The sample mean is equal to 0 ( $\mu = 0$ )
t_statistic, p_value = stats.ttest_1samp(sample, popmean=0)
print(f't-statistic: {t_statistic:.2f}, p-value: {p_value:.4f}')

# Determine if we can reject the null hypothesis
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The sample mean is significantly different from 0.")
else:
    print("Fail to reject the null hypothesis: The sample mean is not significantly different from 0.")
```

Solution:-





```
PS C:\Users\student\Downloads> python prac8.py
Point Estimate (Sample Mean): 0.24
95% Confidence Interval for the Mean: (-0.04076763942877201, 0.513593890311099)
t-statistic: 1.62, p-value: 0.1113
Fail to reject the null hypothesis: The sample mean is not significantly different from 0.
PS C:\Users\student\Downloads>
```

Practical-9

Aim:-

- Identify the column(s) of a given DataFrame which have at least one missing value, count the number of missing values in each column and drop the rows and columns with missing values. Check for the null values. Also remove the duplicate values from the DataFrame. Handle outliers in the Data Frame.
- Access subset of data through indexing(Select data using labels(column headings)), Slicing(Extract range based subset, subset of rows, subset of columns, select a subset of rows and columns from our DataFrame using iloc method).
- Perform other data processing on a given dataset.

Code:-

```
#pip install pandas numpy
import pandas as pd
import numpy as np
# Step 1: Create a sample DataFrame with some missing values and duplicates
data = {
    'A': [1, 2, np.nan, 4, 5, 5, 7, 8],
    'B': [10, np.nan, np.nan, 40, 50, 50, 70, 80],
    'C': [100, 200, 300, 400, 500, 500, 700, 800],
    'D': [1000, 2000, 3000, np.nan, np.nan, 5000, 6000, 7000]
}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
# Step 2: Identify columns with at least one missing value
columns_with_nan = df.columns[df.isnull().any()].tolist()
print("\nColumns with at least one missing value:")
print(columns_with_nan)
# Step 3: Count the number of missing values in each column
missing_values_count = df.isnull().sum()
print("\nCount of missing values in each column:")
print(missing_values_count)
```

```
# Step 4: Drop rows and columns with missing values
df_dropped = df.dropna(axis=0) # Drop rows with any missing value
df_dropped_columns = df.dropna(axis=1) # Drop columns with any missing value
print("\nDataFrame after dropping rows with missing values:")
print(df_dropped)
print("\nDataFrame after dropping columns with missing values:")
print(df_dropped_columns)
# Step 5: Check for null values in the DataFrame
print("\nCheck for null values in the DataFrame:")
print(df.isnull())
# Step 6: Remove duplicate values
df_no_duplicates = df.drop_duplicates()
print("\nDataFrame after removing duplicate values:")
print(df_no_duplicates)
# Step 7: Handle outliers in the DataFrame
# For this example, we'll handle outliers by capping them within 1.5 * IQR range
def cap_outliers(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return np.where(series < lower_bound, lower_bound, np.where(series > upper_bound, upper_bound,
series))
df_capped = df.apply(cap_outliers)
print("\nDataFrame after capping outliers:")
print(df_capped)
# Step 8: Accessing subset of data through indexing
# Selecting specific columns
subset_columns = df[['A', 'B']]
print("\nSubset of DataFrame with columns A and B:")
print(subset_columns)
# Slicing - Extract a range-based subset (subset of rows)
subset_rows = df[2:5]
print("\nSubset of DataFrame with rows 2 to 4:")
print(subset_rows)
```



```
# Subset of rows and columns using iloc method
subset_iloc = df.iloc[2:5, [0, 2]]
print("\nSubset of DataFrame with rows 2 to 4 and columns 0 and 2:")
print(subset_iloc)

# Perform other data processing on a given dataset
# Example: Calculating the mean of each column
mean_values = df.mean()
print("\nMean of each column:")
print(mean_values)

# Example: Normalizing the data using Min-Max scaling
df_normalized = (df - df.min()) / (df.max() - df.min())
print("\nNormalized DataFrame:")
print(df_normalized)
```

Solution:-

[Running] python -u "c:\Users\student\Downloads\Prac9.py"

Original DataFrame:

	A	B	C	D
0	1.0	10.0	100	1000.0
1	2.0	NaN	200	2000.0
2	NaN	NaN	300	3000.0
3	4.0	40.0	400	NaN
4	5.0	50.0	500	NaN
5	5.0	50.0	500	5000.0
6	7.0	70.0	700	6000.0
7	8.0	80.0	800	7000.0

Columns with at least one missing value:

['A', 'B', 'D']

Count of missing values in each column:

A 1

B 2

C 0

D 2

dtype: int64

DataFrame after dropping rows with missing values:

	A	B	C	D
0	1.0	10.0	100	1000.0
5	5.0	50.0	500	5000.0
6	7.0	70.0	700	6000.0
7	8.0	80.0	800	7000.0

DataFrame after dropping columns with missing values:

	C
0	100
1	200
2	300
3	400
4	500
5	500
6	700
7	800

Check for null values in the DataFrame:

	A	B	C	D
0	False	False	False	False
1	False	True	False	False
2	True	True	False	False
3	False	False	False	True
4	False	False	False	True
5	False	False	False	False
6	False	False	False	False
7	False	False	False	False

DataFrame after removing duplicate values:

	A	B	C	D
0	1.0	10.0	100	1000.0
1	2.0	NaN	200	2000.0
2	NaN	NaN	300	3000.0
3	4.0	40.0	400	NaN
4	5.0	50.0	500	NaN
5	5.0	50.0	500	5000.0
6	7.0	70.0	700	6000.0
7	8.0	80.0	800	7000.0

DataFrame after capping outliers:

	A	B	C	D
0	1.0	10.0	100.0	1000.0
1	2.0	NaN	200.0	2000.0
2	NaN	NaN	300.0	3000.0
3	4.0	40.0	400.0	NaN
4	5.0	50.0	500.0	NaN
5	5.0	50.0	500.0	5000.0
6	7.0	70.0	700.0	6000.0
7	8.0	80.0	800.0	7000.0

Subset of DataFrame with columns A and B:

	A	B
0	1.0	10.0
1	2.0	NaN
2	NaN	NaN
3	4.0	40.0
4	5.0	50.0
5	5.0	50.0
6	7.0	70.0
7	8.0	80.0

Subset of DataFrame with rows 2 to 4:

	A	B	C	D
2	NaN	NaN	300	3000.0
3	4.0	40.0	400	NaN
4	5.0	50.0	500	NaN

Subset of DataFrame with rows 2 to 4 and columns 0 and 2:

	A	C
2	NaN	300
3	4.0	400
4	5.0	500

Mean of each column:

A	4.571429
B	50.000000
C	437.500000
D	4000.000000

dtype: float64

Normalized DataFrame:

	A	B	C	D
0	0.000000	0.000000	0.000000	0.000000
1	0.142857	NaN	0.142857	0.166667
2	NaN	NaN	0.285714	0.333333
3	0.428571	0.428571	0.428571	NaN
4	0.571429	0.571429	0.571429	NaN
5	0.571429	0.571429	0.571429	0.666667
6	0.857143	0.857143	0.857143	0.833333
7	1.000000	1.000000	1.000000	1.000000

[Done] exited with code=0 in 1.45 seconds

Practical-10

Aim:- Write a python program to perform data visualization through Matplotlib.

Code:-

```
#pip install matplotlib pandas numpy seaborn
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns

# Step 1: Create or load a sample dataset
# For this example, let's use a simple DataFrame with random data
np.random.seed(42) # For reproducibility
data = {
    'Category': ['A', 'B', 'C', 'D'],
    'Values': np.random.randint(10, 100, 4),
    'Scores': np.random.normal(50, 10, 4),
    'Quantity': np.random.randint(1, 10, 4)
}
df = pd.DataFrame(data)
print("Sample DataFrame:")
print(df)

# Step 2: Create a simple bar chart
plt.figure(figsize=(8, 6))
plt.bar(df['Category'], df['Values'], color='skyblue')
plt.title('Bar Chart of Values by Category')
plt.xlabel('Category')
plt.ylabel('Values')
plt.show()

# Step 3: Create a pie chart
plt.figure(figsize=(8, 6))
plt.pie(df['Values'], labels=df['Category'], autopct='%1.1f%%', startangle=140, colors=sns.color_palette("pastel"))
plt.title('Pie Chart of Values by Category')
plt.show()
```

Step 4: Create a line chart

```
plt.figure(figsize=(8, 6))  
plt.plot(df['Category'], df['Scores'], marker='o', color='green')  
plt.title('Line Chart of Scores by Category')  
plt.xlabel('Category')  
plt.ylabel('Scores')  
plt.show()
```

Step 5: Create a scatter plot

```
plt.figure(figsize=(8, 6))  
plt.scatter(df['Values'], df['Scores'], color='red')  
plt.title('Scatter Plot of Scores vs. Values')  
plt.xlabel('Values')  
plt.ylabel('Scores')  
plt.show()
```

Step 6: Create a histogram

```
data = np.random.randn(1000) # Normally distributed data  
plt.figure(figsize=(8, 6))  
plt.hist(data, bins=30, color='purple', alpha=0.7)  
plt.title('Histogram of Normally Distributed Data')  
plt.xlabel('Value')  
plt.ylabel('Frequency')  
plt.show()
```

Step 7: Create a box plot

```
plt.figure(figsize=(8, 6))  
sns.boxplot(x='Category', y='Scores', data=df)  
plt.title('Box Plot of Scores by Category')  
plt.xlabel('Category')  
plt.ylabel('Scores')  
plt.show()
```

Step 8: Create a heatmap

Let's create a correlation matrix as an example

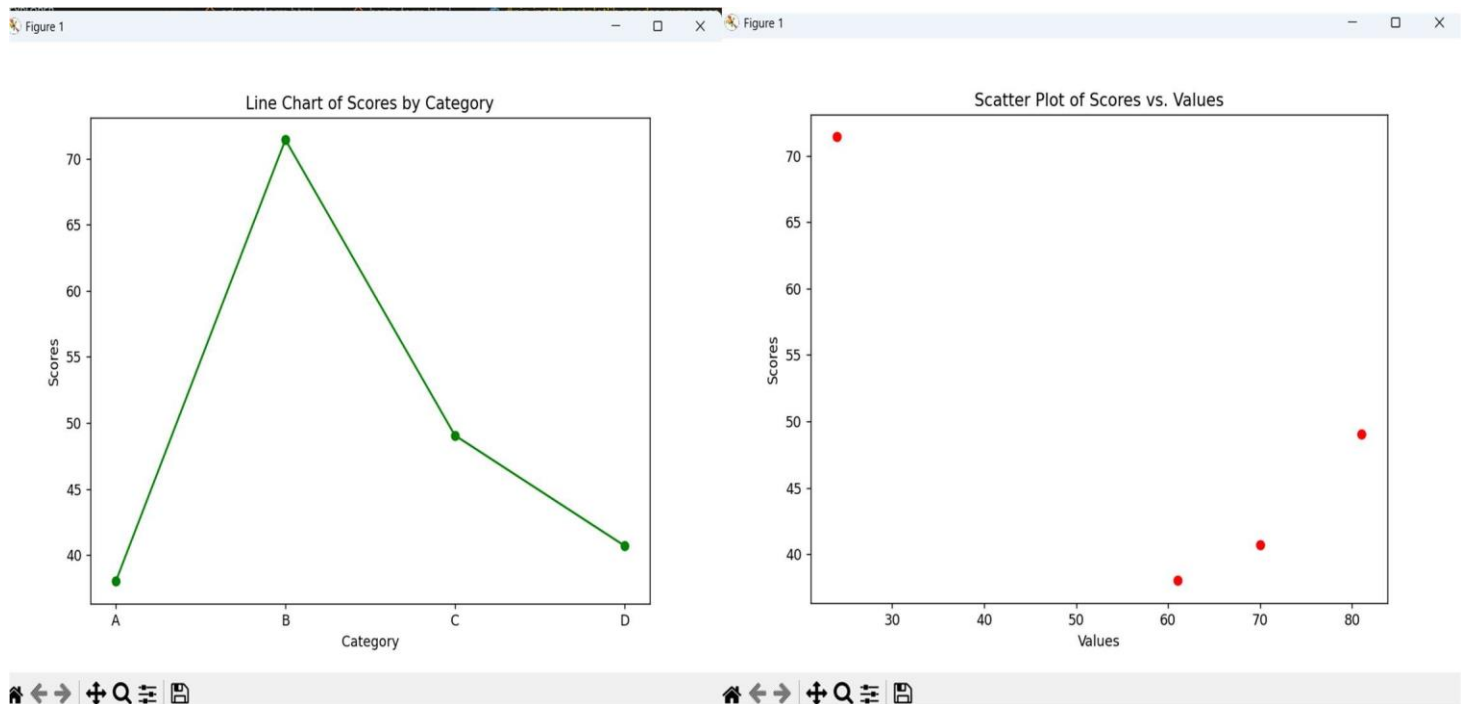
```
df_corr = df.corr()  
plt.figure(figsize=(8, 6))  
sns.heatmap(df_corr, annot=True, cmap='coolwarm', fmt=".2f")  
plt.title('Heatmap of Correlation Matrix')  
plt.show()
```

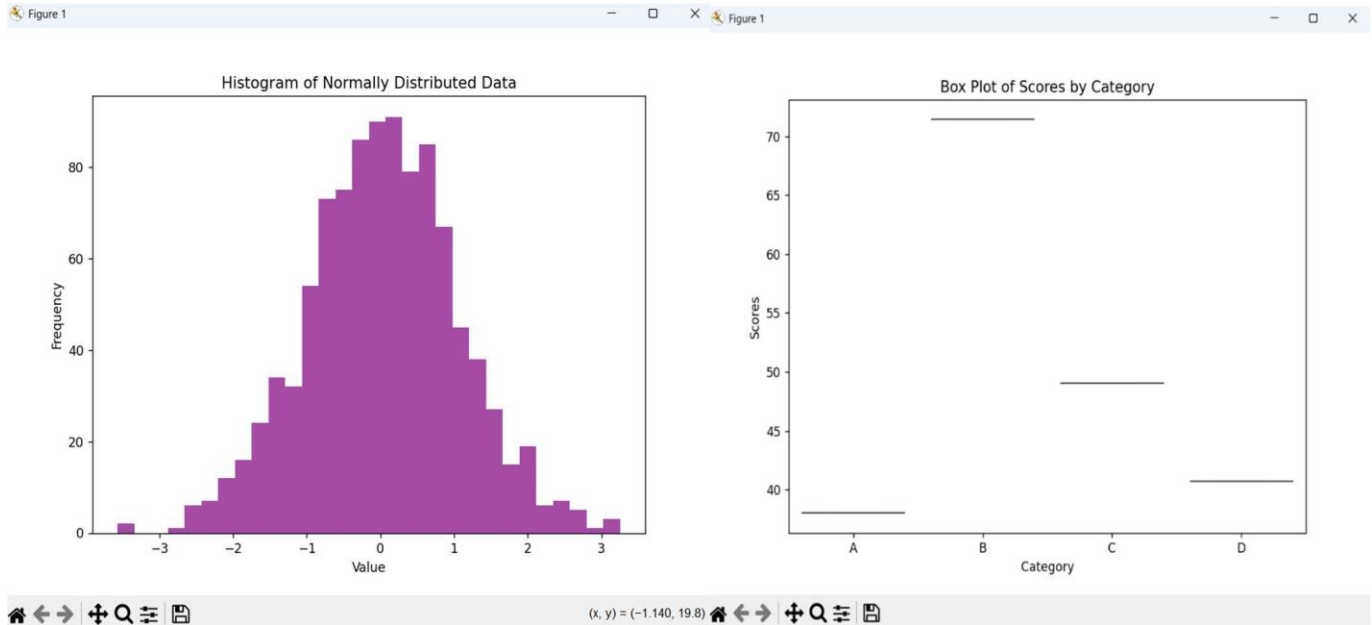
```
# Step 9: Create a pair plot
# Generating pair plots for more insights
sns.pairplot(df)
plt.suptitle('Pair Plot of the DataFrame', y=1.02)
plt.show()
# Step 10: Create a subplot (multiple plots in a single figure)
plt.figure(figsize=(12, 8))
# Subplot 1: Bar chart
plt.subplot(2, 2, 1)
plt.bar(df['Category'], df['Values'], color='blue')
plt.title('Bar Chart')
# Subplot 2: Line chart
plt.subplot(2, 2, 2)
plt.plot(df['Category'], df['Scores'], marker='o', color='orange')
plt.title('Line Chart')
# Subplot 3: Scatter plot
plt.subplot(2, 2, 3)
```

Solution:-

Sample DataFrame:

	Category	Values	Scores	Quantity
0	A	61	38.021936	5
1	B	24	71.416584	4
2	C	81	49.053790	8
3	D	70	40.711719	8





Practical-11

Aim: Write python program for advanced data visualization through Seaborn

Code:

```
#pip install seaborn pandas matplotlib numpy
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Load or create a sample dataset
# For demonstration, we'll use Seaborn's built-in datasets
# Load the 'tips' dataset, which contains data about restaurant bills and tips
tips = sns.load_dataset('tips')

# Step 2: Set Seaborn style for plots
sns.set(style='whitegrid')

# Step 3: Advanced Data Visualization
# 3.1. Violin Plot - Distribution of tips by day of the week and gender
plt.figure(figsize=(8, 6))
sns.violinplot(x='day', y='tip', hue='sex', data=tips, split=True, inner='quart', palette='muted')
plt.title('Violin Plot of Tips by Day and Gender')
plt.show()

# 3.2. Pair Plot - Pairwise relationships between variables in the dataset
sns.pairplot(tips, hue='sex', palette='coolwarm')
plt.suptitle('Pair Plot of Tips Dataset', y=1.02)
plt.show()

# 3.3. Heatmap - Correlation matrix with annotations
plt.figure(figsize=(10, 8))
corr_matrix = tips.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap of Correlation Matrix')
plt.show()

# 3.4. Box Plot - Distribution of total bill by day and time of day
plt.figure(figsize=(8, 6))
sns.boxplot(x='day', y='total_bill', hue='time', data=tips, palette='Set2')
plt.title('Box Plot of Total Bill by Day and Time of Day')
plt.show()

# 3.5. Swarm Plot - Tips by day of the week with jittered points
plt.figure(figsize=(8, 6))
sns.swarmplot(x='day', y='tip', data=tips, color='purple')
plt.title('Swarm Plot of Tips by Day')
plt.show()
```

3.6. Facet Grid - Distribution of total bill across time of day and smoker status

```
g = sns.FacetGrid(tips, col='time', row='smoker', margin_titles=True)
g.map(sns.histplot, 'total_bill', bins=10, color='teal')
g.fig.suptitle('Facet Grid of Total Bill by Time and Smoker Status', y=1.03)
plt.show()
```

3.7. Joint Plot - Relationship between total bill and tip with regression line

```
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg', color='green')
plt.suptitle('Joint Plot of Total Bill vs Tip with Regression Line', y=1.02)
plt.show()
```

3.8. Count Plot - Count of observations in each categorical bin using bars

```
plt.figure(figsize=(8, 6))
sns.countplot(x='day', hue='smoker', data=tips, palette='coolwarm')
plt.title('Count Plot of Days by Smoker Status')
plt.show()
```

3.9. KDE Plot - Kernel density estimate of total bill for different time periods

```
plt.figure(figsize=(8, 6))
sns.kdeplot(data=tips, x='total_bill', hue='time', fill=True, palette='crest', alpha=0.5)
plt.title('KDE Plot of Total Bill by Time of Day')
plt.show()
```

3.10. Strip Plot - Tips by day of the week, with jitter

```
plt.figure(figsize=(8, 6))
sns.stripplot(x='day', y='tip', data=tips, jitter=True, hue='sex', palette='Set1', dodge=True)
plt.title('Strip Plot of Tips by Day with Jitter')
plt.show()
```

Solution:

