lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with '' A'' Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

MASTER LABORATORY MANUAL

OF

II B.TECH I SEMESTER (R20)

PYTHON PROGRAMMING

Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with '' A'' Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

INSTITUTE

VISION

Producing globally competent and quality technocrats with human values for the holistic needs of industry and society

MISSION

- Creating an outstanding infrastructure and platform for enhancement of skills, knowledge and behaviour of students towards employment and higher studies
- Providing a healthy environment for research, development and entrepreneurship, to meet the expectations of industry and society.
- Transforming the graduates to contribute to the socio-economic development and welfare of the society through value based education.

lendi Institute of Engineering & Technology
An Autonomous Institution
Accredited by NAAC with '' A'' Grade, Accredited by NBA (ECE, CSE.EEE & MECH)
Approved by A.I.C.T.E. & Permanently Affiliated to J. N. T. U. Gurajada, VIZIANAGARAM
Via 5th APSP Battalion, Jonnada (V), Denkada (M), NH-3, Vizianagaram Dist - 535005, A.P. Website : www.lendi.org
Ph : 08922-241111, 241666, Cell No : 9490344747, 9490304747,e-mail : lendi_2008@yahoo.com

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

VISION

To excel in the computing arena and to produce globally competent computer science and Information Technology graduates with Ethical and Human values to serve society.

MISSION

- To impart strong theoretical and practical background in computer science and information technology discipline with an emphasis on software development.
- To provide an open environment to the students and faculty that promotes professional growth
- To inculcate the skills necessary to continue their education and research for contribution to society.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates of Computer Science and Information Technology will acquire strong knowledge to analyze, design, and develop computing products and solutions for real-life problems utilizing the latest tools, techniques, and technologies.

PEO2: Graduates of Computer Science and Information Technology shall have interdisciplinary approach, professional attitude and ethics, communication, teamwork skills and leadership capabilities to solve social issues through their Employment, Higher Studies and Research.

PEO3: Graduates will engage in life-long learning and professional development to adapt to dynamically computing environment.

PROGRAM OUTCOMES (POs)

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design & Development: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

PO4: Investigations: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tools: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: Engineer & Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment & Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice

PO9: Individual & Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings

PO10: Communication Skills: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions

PO11: Project mgt. & Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments

PO12: Life Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Ability to solve contemporary issues utilizing skills

PSO2: To acquire knowledge of the latest tools and technologies to provide technical solutions

PSO3: To qualify in national and international competitive examinations for successful higher studies and Employment

Course Code	Subject Name	L	T	P	C
R20CIT-PC2102	Python Programming LAB	0	0	3	1.5

Course Outcomes:

1. Understand the working environment of Python and its program structure.
2. Implement conditional and iterative statements.
3. Create custom modules and functions to handle different operations.
4. Implement Object oriented concepts through real time scenarios and handle errors.

Experiment 1: Basic I/O

a. Demonstrate the python script by running in Interactive and Script Mode.
b. Write a python script to read using input() and display using print() functions.
c. Write a python script to make use of all conversion functions.

Experiment 2: Decision Making

a. Write a python script to take five subject marks and print the grade for the student.
b. Write the python script to print whether the roots are equal, distinct or complex for given coefficients a, b and c for quadratic equation.

Experiment 3: Loops

a. Write a program to take input as integer N and check whether N is Pronic Number or not. (Product of two consecutive numbers is pronic N(N+1): Eg 110 = 10*11)
b. Write a python script to take input as amount in rupees R and find out the least number of notes N that can be possible to store in a Wallet.(Hint Notes: 2000,500,200,100,50,20,10) Eg: R=2589, N=5
c. Write a program to check whether given number N is N-Series(Disarium) number or not**.** (Eg. 135 is N-Series Number because $1^1+3^2+5^3 = 135$ and some others are 89, 175, 518 etc)

Experiment 4: Nested Loops

a. Write a python script to take input as String S="LENDI", print the following:

```
L
LEL
LENEL
LENDNEL
LENDIDNEL
```

b. Write a python script to print the any alphabet shape using *s.

```
* * *
*     *
* * * *
*     *
*     *
```

Experiment 5: Modules & Functions

a. Using Recursion, Write a program to take input as vehicle Number N and check whether N is Fancy number or not. (Folding of digits of number should be 9)
b. Create a module named “Lendi” and create functions addStudent, removeStudent, searchStudent. Access the above module using import statement.
c. Write a python script using lambdas, to take input as String, and sort the string SS in descending/ascending order according to their frequency of its occurrences of characters.(Eg.S='missisippi', SS=ispm)

Experiment 6: Permutations & Combinations

a. Write a python script to take input as number N, and find out the largest number L , that can be formed with N.Eg. N=679, P={679,697,769,796,967,976}, L = 976

b. Write a python script to take input as list, L and print output as largest number L and total combinations C for given N digit number formed by the combination of L.(Eg. L=[1,2,1,4], N=3, L=421,C=12).

Experiment 7: String &Regular Expressions

a. Write a python script to take two string S1 and S2 and do the following:
   i) Check S1 and S2 are anagrams or not.
   ii) Check S1 is Sub string of S2 or not.
   iii) S1 is palindrome or not
b. Write a python script to take input as multi-line string and find the sum of all numbers in that string using re module. (Eg. S="he11o they are 40students in97 room of 4th line", Sum= 152)

Experiment 8: Lists & Dictionary

a. Write a program to take input as String S and print frequency of each character in S using List data structure.
b. Write a program to take input as String S contains characters and special symbols, reverse the String S such that special symbols remains at same position. (Eg. S="m@d#u" , Output="u@d#m").
c. Write a python script to take input as String sentence S and print each word count using dictionary.

Experiment 9: OOPS

a. Using Python OOPS, create a class, constructor, method, __str__ and __repr__ for:
   i. Employee
   ii. Student

Experiment 10: Exceptions

a. Write a python program to implement Exceptions hierarchy.
b. Create a user defined Exception named "FundsLessException" and raise the exception when there are no enough funds in the bank account.

Experiment 11: Data Analysis

a. Using NumPy, implement different matrix operations in python.
b. Using pandas, read the data from anytext files.

Experiment 12: Plotting

a. Using MatPlotlib library, plot the graph with all different plot types.(Pie Chart, Area Plot, Scatter Plot, Histogram and Bar Graph)

APPLICATIONS:

- Web Application Development and Scraping
- Designing Games
- Machine Learning and AI based applications
- Data Science and Visualization
- Embedded and CAD Application

COURSE OUTCOMES VS POs MAPPING (DETAILED; HIGH:3; MEDIUM:2; LOW:1**):**

SNO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
C204.1	3	2	2	2	3			2	2			1	2	2	2
C204.2	3	3	2	2	3			2	2			2	2	2	3
C204.3	3	3	2	2	3			1	2		2	2	3	2	3
C204.4	3	3	3	2	3	1	1	1	2		2	2	3	2	3
C204.5	3	3	3	2	3	1		1	2		2	2	3	2	3
C204.*	3	3	3	2	3	1	1	2	2	1	2	2	3	2	3

COs VS POs MAPPING JUSTIFICATION:

The students will be able to:

S.NO	PO/PSO MAPPED	LEVEL OF MAPPING	JUSTIFICATION
C204.1	PO1	3	Acquire the knowledge of working environment of python.
	PO2	2	Understand the basics of variables.
	PO3	2	Understand the structure of python scripts
	PO4	2	Identify the python Shell and its data stack.
	PO5	3	Use the REPL shell to run basic python scripts.
	PO8	2	Apply the programming skills for healthy practices only by following ethics
	PO9	1	Involve as a team to understand Interactive Development Learning Environment.
	PO12	1	Understand the concepts of python and apply to real time.
	PSO1	2	Write complex scripts in REPL shell.
	PSO2	2	Analyze the data through assignment of values to variables.
	PSO3	2	Face interviews with ease and comfort.
C204.2	PO1	3	Acquire the knowledge of decision and iterative statements
	PO2	3	Analyze the data using different methods like string and number.
	PO3	2	Develop the simple python scripts that handle text processing.
	PO4	2	Identify the usage of iterations and decisions in strings and numbers.
	PO5	3	Write the python scripts in latest tools like Miniconda.

	PO8	2	Apply the programming skills for healthy practices only by following ethics
	PO9	2	Creating basic applications using the decision making statements.
	PO12	2	Apply decision statements and loops in real world data.
	PSO1	2	Enhance the script using modules to handle web data.
	PSO2	2	Analyze the data using new modules of python.
	PSO3	3	Resolve the complex problems in competitive exams.
C204.3	PO1	3	Acquire the knowledge of modules and packages and their usability.
	PO2	3	Analyze the data with new python APIs using PIP.
	PO3	2	Design user defined modules and packages to customize the script.
	PO4	2	Identify new modules and their usage in documenting data.
	PO5	3	Use modern tools like Anaconda to handle new packages in python 3.x version.
	PO8	1	Apply the programming skills for healthy practices only by following ethics
	PO9	2	Creating basic applications using the functions and modules
	PO11	2	Create basic projects to deal contemporary issues.
	PO12	2	Understand python usage in smart applications like IOT devices and mobile phones.
	PSO1	3	Write advanced script in python using latest packages of python.
	PSO2	2	Analyze the data in network or from internet using latest modules.
	PSO3	3	Solve various problems during interviews and exams.
C204.4	PO1	3	Acquire knowledge of organizing data using data structures.
	PO2	3	Solve contemporary issues using data objects.
	PO3	3	Highly Develop a data structure to solve complex problems.
	PO4	2	Identify the complex data structures and their usage.
	PO5	3	Use latest tools like Pycharm to make script much easier.
	PO6	1	Applying contextual knowledge for handling societal needs through coding skills
	PO7	1	Understand and apply the essence of oops concepts for sustainable development

	PO8	1	Apply the programming skills for healthy practices only by following ethics
	PO9	2	Creating basic applications using the functions and modules
	PO11	2	Demonstrate the knowledge and utilize for multidisciplinary project implementations.
	PO12	2	Use data structures in their regular life to solve complex problems.
	PSO1	3	Resolve the problems using enhanced modules of python apis.
	PSO2	2	Analyze the data from web documents and store into data structures.
	PSO3	3	Face the interviews easily through data processing in python.
C204.5	PO1	3	Acquire the knowledge of object oriented concepts and their advantages.
	PO2	3	Solve the real world problems that are related to objects.
	PO3	3	Highly Design new classes and import them in python standard library.
	PO4	2	Identify new error handling mechanisms to solve complex issues.
	PO5	3	Use latest tools to develop projects in python such as Eclipse and NetBeans.
	PO6	1	Applying contextual knowledge for handling societal needs through coding skills
	PO8	1	Apply the programming skills for healthy practices only by following ethics
	PO9	3	Involve as team to fix bugs and raise exceptions.
	PO11	2	Demonstrate the knowledge and utilize for multidisciplinary project implementations.
	PO12	2	Apply concepts of objects in daily life and solve contemporary issues.
	PSO1	3	Understand advanced APIs to handle different data analytics operations.
	PSO2	2	Using latest techniques in python like numpy to handle mathematical solutions.
	PSO3	3	Face App challenges and data science competitions.

Experiment1: BasicI/O

a) Demonstrate the python script by running in Interactive and Script Mode.

1. Interactive Mode:

Interactive Mode is when you run Python code line-by-line in a Python shell or interpreter.

You can do this in**:**

- Python's REPL(just type python or python 3 in the terminal)
- Jupyter Notebook
- Python shell in an IDLE

DESCRIPTION :

- Python Interactive Mode allows users to execute Python commands one at a time directly in the command-line interface or terminal.
- It is useful for testing small code snippets and debugging quickly.
- The prompt is represented by >>> where users can type and run Python expressions.
- It provides immediate feedback by displaying the result of each command instantly.
- This mode is ideal for learning, experimenting, and prototyping Python code.

ALGORITHM :

1. Interactive Mode:

STEP-1: Open the Python interpreter:

Open a terminal or command prompt and type python (or python3 on some systems) to start an interactive Python session.

STEP-2: Execute commands one by one:

In this mode, you can type Python code line by line, and it will execute immediately after you press Enter.

STEP-3: Test or experiment with code:

You can use this mode to test small snippets of code or explore Python features quickly. It's great for debugging or trying out new concepts.

STEP-4: Exit Interactive Mode:

To exit the interactive mode, you can type exit() or simply press Ctrl + Z (Windows) or Ctrl + D (Mac/Linux).

2. Script Mode

Script Mode is when you write Python code in a.py file and then run the file all at once.

- Create a file called square_calculator.py:

DESCRIPTION :

- Python Script Mode allows users to write and save code in a file with a .py extension.
- The entire script is executed at once, making it suitable for larger programs.
- It enables code reuse, organization, and debugging of complex logic.
- Scripts can be run from the command line or an IDE like VS Code or PyCharm.
- This mode is ideal for developing full applications and automation tasks.

ALGORITHM :

2. Script Mode:

STEP-1: Write the Python script:

In script mode, you write your Python code in a file (typically with a .py extension).

STEP-2: Save the script:

Once the code is written, save it with a .py extension (e.g., my_script.py).

STEP-3: Run the script:

To run the script, open the terminal/command prompt, navigate to the folder where the script is saved, and type python my_script.py (or python3 my_script.py on some systems).

STEP-4: See the output:

When the script is executed, Python processes the entire file and displays the output (if any) all at once.

STEP-5: Edit and re-run:

You can edit the script and rerun it as needed. This is ideal for larger programs that are intended to be reused or shar

b) Write a python script to read using input() and display using print() functions.

DESCRIPTION:

- This Python script reads user input using the **input()** function.
- It captures the data entered by the user as a string.
- The script then processes or directly uses the input value.
- It displays the output or message using the **print()** function.
- This is commonly used for interactive programs that take user data and respond accordingly.

ALGORITHM

STEP-1: Start

STEP-2: Read Input:

Use the input() function to get input from the user.

STEP-3: Process Input (Optional):

You can perform some processing on the input if needed (e.g., convert the input to an integer or manipulate it).

STEP-4: Display Output:

Use the print() function to display the result or the input back to the user.

STEP-5: End

C) Write a python script to make use of all conversion functions.

DESCRIPTION:

- This Python script demonstrates the use of built-in type conversion functions.
- It converts data between types such as int(), float(), str(), bool(), and complex().
- The script takes input values and converts them to different types using these functions.
- It displays the converted results using the print() function.
- This helps in understanding how data types interact and change in Python.

ALGORITHM :

STEP-1: **Start**

STEP-2: **Initialize some variables**:

Choose values of different types like integers, floats, strings, lists, etc.

STEP-3: **Convert data types**:

Use the int() function to convert to an integer.

Use the float() function to convert to a float.

Use the str() function to convert to a string.

Use the list() function to convert a string or tuple to a list.

Use the tuple() function to convert a list or string to a tuple.

Use the set() function to convert a list or tuple to a set.

STEP-4: **Print the results**:

Display the converted values using the print() function.

STEP-5: End

VIVA QUESTIONS :

1. What is the difference between input() and print() functions in Python?
2. How would you take an integer input from the user in Python?
3. What will happen if you try to convert a non-numeric string to an integer using int()?
4. Can you explain how to take multiple inputs from a user in one line in Python?
5. What is the output of the following code? print("Hello " + input("Enter your name: "))

Experiment2: Decision Making

a.Write a python script to take five subject marks and print the grade for the student.

DESCRIPTION:

- This Python script takes input for marks in five different subjects from the user.
- It calculates the total and average of the entered marks.
- Based on the average, it determines the student's grade using conditional statements.
- Grades are assigned using a standard grading scale (A, B, C, D, F).
- The script then displays the total, average, and the final grade to the user.

ALGORITHM :

STEP-1: Start

STEP-2: Input Marks:

Prompt the user to input marks for 5 subjects using the input() function.

Convert the input marks to float type.

STEP-3: Calculate Total Marks:

Add all the marks together to get the total.

STEP-4: Calculate Average:

Divide the total marks by 5 to get the average.

STEP-5: Determine Grade:

Check the average score:

If the average is greater than or equal to 90, assign grade 'A'.

If the average is greater than or equal to 80 but less than 90, assign grade 'B'.

If the average is greater than or equal to 70 but less than 80, assign grade 'C'.

If the average is greater than or equal to 60 but less than 70, assign grade 'D'.

If the average is less than 60, assign grade 'F'.

STEP-6: Display Results:

Print the total marks, average, and grade.

STEP-7: End

b.Write the python script to print whether the roots are equal, distinct or complex for given coefficients a, b and c for quadratic equation.

DESCRIPTION :

- This Python script takes input for the coefficients **a**, **b**, and **c** of a quadratic equation.
- It calculates the discriminant (**$D = b^2 - 4ac$**) to determine the nature of the roots.
- Based on the discriminant, the script checks if the roots are equal, distinct, or complex.
- It uses conditional statements to classify and print the type of roots.
- This helps in understanding how the discriminant affects the solution of quadratic equations.

ALGORITHM :

STEP-1: Start

STEP-2: **Input** the coefficients a, b, and c.

STEP-3: Calculate the discriminant: D = b^2 - 4ac.

STEP-4: **If D > 0**:

Calculate real and distinct roots using the quadratic formula:

root1=−b+D2a, root2=−b−D2a\text{root1} = \frac{-b + \sqrt{D}}{2a}, \quad \text{root2} = \frac{-b - \sqrt{D}}{2a}root1=2a−b+D,root2=2a−b−D

Print "Roots are real and distinct".

STEP-5: Else If D == 0:

Calculate the real and equal root:

root=−b2a\text{root} = \frac{-b}{2a}root=2a−b

Print "Roots are real and equal".

STEP-6: **Else**:

Calculate complex roots using cmath.sqrt(D):

root1=−b+cmath.sqrt(D)2a,

root2=−b−cmath.sqrt(D)2a\text{root1} = \frac{-b + \text{cmath.sqrt}(D)}{2a},

\quad \text{root2} = \frac{-b - \text{cmath.sqrt}(D)}{2a}root1=2a−b+cmath.sqrt(D), root2=2a−b−cmath.sqrt(D)

Print "Roots are complex".

STEP-7: **End**

VIVA QUESTIONS:

1. What is a conditional statement in Python? Can you name the types of conditional statements available?
2. How does the if statement work in Python? Provide an example.
3. What is the purpose of the else and elif statements in Python? How are they used in decision making?
4. What happens if you write an if statement without an else part? Is it mandatory to include else in every if statement?
5. Explain the difference between if and elif statements in Python with an example

Experiment3: Loops

a.Write a program to take input as integer N and check whether N is Pronic Number or not. (Product of two consecutive numbers is pronic N(N+1):Eg110=10*11)

DESCRIPTION:

- This Python program takes an integer **N** as input from the user.
- It checks whether the number is a **Pronic Number** (i.e., the product of two consecutive integers).
- The script iteratively multiplies consecutive numbers to see if any match the input **N**.
- If a match is found, it prints that **N** is a Pronic Number; otherwise, it is not.
- This helps in identifying numbers that follow the Pronic number pattern (e.g., 6 = 2×3, 12 = 3×4).

ALGORITHM :

STEP-1: **Start**

STEP-2: Input the integer N from the user.

STEP-3: Initialize a boolean variable is_pronic as False.

STEP-4: **Loop** through integers i from 0 to N (inclusive):

Check if i * (i + 1) == N:

If true, set is_pronic to True and exit the loop.

STEP-5: If is_pronic is True:

Print that N is a Pronic number.

STEP-6: **Else**:

Print that N is NOT a Pronic number.

STEP-7: End

b.Write a python script to take input as amount in rupees R and find out the least number of notes N that can be possible to store in a Wallet.(Hint Notes: 2000,500,200,100,50,20,10) Eg: R=2589, N=5

DESCRIPTION :

- This Python script takes an input amount **R** in rupees from the user.
- It calculates the **least number of currency notes** needed to make up the amount.
- The script uses available denominations: 2000, 500, 200, 100, 50, 20, and 10.
- It applies a greedy approach by subtracting the highest possible note value first.
- Finally, it prints the total number of notes required to store the amount efficiently.

ALGORITHM :

STEP-1: Start

STEP-2: **Input** the amount R in rupees.

STEP-3: Initialize a list notes with the available denominations:

[2000, 500, 200, 100,50, 20,10].

STEP-4: Initialize a counter count to 0 (to keep track of the total number of notes).

STEP-5: For each note in the notes list:

If R >= note:

Calculate how many notes of that denomination are needed using integer division: n = R // note.

Update R to the remainder using modulus: R = R % note.

Add the number of notes n to the counter count.

STEP-6: Output the value of count, which represents the minimum number of notes needed.

STEP-7: End

c.Write a program to check whether given number N is N- Series (Disarium) number or not. (Eg.135 is N-Series Numberbecause11+32+53=135andsomeothersare89,175,518 etc)

DESCRIPTION:

- This Python script takes an integer **N** as input from the user.
- It checks whether the number is an **N-Series (Disarium) Number**, where the sum of its digits raised to the power of their positions equals the number itself.
- The script extracts each digit and raises it to the power of its position (starting from 1).
- It sums these powered digits and compares the result with the original number.
- If they match, it prints that **N** is a Disarium Number; otherwise, it is not.

ALGORITHM :

STEP-1: Start

STEP-2: Input the number N.

STEP-3: Convert the number N to a string num_str to easily access its digits.

STEP-4: Initialize sum_pow to 0 (this will hold the sum of each digit raised to the power of its position).

STEP-5: For each digit in num_str (loop through each character and its index i):

Convert the digit to an integer.

Raise the digit to the power of i+1 (since the position starts from 1).

Add the result to sum_pow.

STEP-6: If sum_pow is equal to N:

Print that N is a Disarium number.

STEP-6: Else:

STEP-7: Print that N is NOT a Disarium number.

STEP-8: End

VIVA QUESTIONS:

1. What is the difference between a while loop and a for loop in Python?
2. How does a while loop work in Python? Provide an example.
3. How does a for loop work in Python? Can you explain how it iterates over a range of numbers?
4. What is the purpose of the break statement in a loop? Provide an example of how it can be used.
5. What is the purpose of the continue statement in a loop? How does it differ from break?

Experiment4: Nested Loops

a. **Write a python script to take input as String S="LENDI", print the following :**

L

LEL

LENEL

LENDNEL

LENDIDNEL

DESCRIPTION :

- This Python script takes a string **S = "LENDI"** and generates a specific pattern of output.
- It iterates through the string and prints each character with increasing length in the center, creating a mirrored structure.
- In each line, the substring is gradually extended from the beginning, while the reverse of the substring is appended symmetrically.
- The script uses string slicing and concatenation to build the desired pattern.
- Finally, it prints each line to display the formatted output in the specified pattern.

ALGORITHM :

STEP-1: Start

STEP-2: Initialize the string S (e.g., S = "LENDI").

STEP-3: For each index i from 1 to len(S) (inclusive):

Extract the left part of the string as left = S[:i] (substring from the start to index i-1).

Extract the right part of the string as right = S[i-2::-1] (substring starting from i-2 and reversed).

Print the concatenation of left and right.

STEP-4: End

b.Write a python script to print the any alphabet shape using*s.

```
 ***
*   *
 ** **
*   *
*   *
```

DESCRIPTION :

- This Python script prints a specific alphabet shape (e.g., letter "A") using asterisks (*).
- It uses loops and conditional statements to print each row of the shape.
- The script handles the placement of the stars in each line to form the structure of the alphabet.
- Spacing is managed using spaces to align the stars correctly and maintain the shape.
- Finally, it outputs the alphabet shape row by row, formatted with asterisks.

ALGORITHM :

STEP-1: Start

STEP-2: Print the first line: " * * *".

STEP-3: Print the second line: "* *".

STEP-4: Print the third line: "****".

STEP-5: Print the fourth line: "* *".

STEP-6: **Print** the fifth line: "* *".

STEP-7: End

VIVA QUESTIONS:

1. What is a nested loop in Python? Can you explain it with an example?
2. How does a nested for loop work? Provide an example where a nested for loop is used.
3. What is the difference between a single loop and a nested loop in terms of execution and performance?
4. How do you iterate over a two-dimensional list (matrix) using nested loops? Provide an example.
5. What happens if you use a while loop inside a for loop? Can you give an example?

Experiment5: Modules & Functions

a.Using Recursion, Write a program to take input as vehicle Number N and check Whether N is Fancy number or not.(Folding of digits of number should be 9)

DESCRIPTION:

- This Python script uses recursion to check if a given vehicle number **N** is a Fancy number.
- A Fancy number is defined as a number where the sum of the digits, when recursively folded (added together) until a single digit remains, equals 9.
- The script breaks the number into individual digits and recursively sums them.
- Once the sum is reduced to a single digit, it checks if the result equals 9.
- If the final single digit is 9, the script prints that **N** is a Fancy number; otherwise, it is not.

ALGORITHM :

STEP-1: **Start**

STEP-2: Define a function fold_digits(n):

If n < 10, return n (base case: the number is already a single digit).

Else, return the recursive call fold_digits(sum of digits of n).

STEP-3: Input a vehicle number num from the user.

STEP-4: Check if the result of fold_digits(num) is equal to 9:

If true, print that num is a Fancy Number.

Else, print that num is NOT a Fancy Number.

STEP-5: End

Program:

```
deffold_digits(n): if n < 10:
    returnn else:
    returnfold_digits(sum(int(d)fordinstr(n)))
    num = int(input("Enter vehicle number: "))
    if fold_digits(num) == 9: print(f"{num}isaFancyNumber!")
    else:
        print(f"{num}isNOTaFancyNumber.")
```

Output:

```
Entervehiclenumber:936 936 is a Fancy
Number!
```

b.Create a module named “Lendi” and create functions add Student, remove Student, search Student. Access the above module using import statement.

DESCRIPTION:

- This Python script defines a module named **"Lendi"** containing three functions: **add_student**, **remove_student**, and **search_student**.
- The **add_student** function adds a student's details to a list or database.
- The **remove_student** function removes a student's details from the list using their unique identifier.
- The **search_student** function searches for a student by name or ID and returns their details.
- The module is accessed using the import statement in another script to manage student data efficiently

ALGORITHM :

STEP-1: Create a file named Lendi.py

STEP-2: Start

STEP-3: Create a list students to store the names of the students.

STEP-4: Define function addStudent(name):

Append the given name to the students list.

Print a message confirming the student was added.

STEP-5: Define function removeStudent(name):

Check if name is in the students list.

If yes, remove name from the list and print a message confirming the removal.

If no, print a message indicating the student was not found.

STEP-6: Define function searchStudent(name):

Check if name is in the students list.

If yes, print a message confirming the student is in the list.

If no, print a message indicating the student was not found.

STEP-7: End

ALGORITHM :

Step 2: Use the module in another script (manage_students.py)

STEP-1: Start

STEP-2: Import the Lendi module in the manage_students.py script.

STEP-3: Call the addStudent() function with "Alice" and "Bob" as arguments.

STEP-4: Call the searchStudent() function with "Alice" to check if Alice is in the list.

STEP-5: Call the removeStudent() function with "Bob" to remove Bob from the list.

STEP-6: Call the searchStudent() function with "Bob" to check if Bob is still in the list.

STEP-7: End

c.Write a python script using lambdas, to take input as String, and sort the string SS in descending/ascending order according to their frequency of its occurrences of characters.(Eg.S='missisippi', SS=ispm)

DESCRIPTION:

- This Python script uses a **lambda function** to sort the characters of a string **S** based on their frequency of occurrence.
- It takes input as a string and counts the frequency of each character.
- The characters are then sorted in either **ascending or descending** order based on their frequency, using a lambda function as the key for sorting.
- The script utilizes Python's sorted() function, along with a custom sorting criteria defined by the lambda.
- Finally, the sorted characters are printed as a new string, with characters ordered according to their frequency.

ALGORITHM:

STEP-1: Start

STEP-2: Input the string S.

STEP-3: Initialize an empty dictionary freq to store the frequency of each character.

STEP-4: Loop through each character char in the string S:

Update the frequency of char in the dictionary freq using freq.get(char, 0) + 1.

STEP-5: Sort the characters by their frequency in descending order. If frequencies are the same, sort alphabetically:

Use sorted() to sort freq.items(), with a custom sorting key:

First, sort by frequency in descending order (-x[1]).

Second, sort alphabetically (x[0]).

STEP-6: Create a new string SS by concatenating characters, repeated by their frequency, from the sorted result.

STEP-7: Print the sorted string SS.

STEP-8: End

VIVA QUESTIONS :

1. What is the purpose of using functions in Python? How do functions help in improving code modularity?
2. How do you define a function in Python? Provide an example of a simple function.
3. What is the difference between a function with parameters and a function without parameters? Provide an example for each.
4. What is the `return` statement in Python functions? How is it different from `print()`?
5. Can a function return multiple values in Python? If yes, how can you achieve that? Provide an example.

Experiment-6: Permutations & Combinations

a.Write a python script to take input as number N, and find out the largest number L,that can be formed with N. Eg. N=679, P={679,697,769,796,967,976}, L = 976}

DESCRIPTION :

- This Python script takes an integer **N** as input and extracts its digits.
- It generates all possible permutations of the digits to form different numbers.
- The script uses the itertools. permutations() function to generate all permutations of the digits.
- Each permutation is converted back to an integer and the maximum value is selected.
- Finally, the script prints the largest number **L** that can be formed from the digits of **N**.

ALGORITHM :

STEP-1: Start

STEP-2: Input the number N as a string (to handle individual digits).

STEP-3: **Sort** the digits of the string N in descending order using the sorted() function.

STEP-4: Join the sorted digits together to form the largest number using ".join().

STEP-5: Print the largest number formed.

STEP-6: **End**

b.Write a python script to take input as list, L and print output as largest number L and total combinations C for given N digit number formed by the combination of L.(Eg. L=[1,2,1,4], N=3, L=421,C=12)

DESCRIPTION :

- This Python script takes a list **L** and a number **N** as input.
- It generates all possible combinations of **N** digits from the list **L** using the itertools.combinations() function.
- The script then calculates the largest number **L** that can be formed from the combinations.
- It also counts the total number of combinations **C** that can be formed for the given **N** digit number.
- Finally, it prints the largest number and the total combinations formed.

ALGORITHM :

STEP-1: Start

STEP-2: Input the list L as space-separated integers and convert it into a list.

STEP-3: Input the number N, which specifies the length of the permutations.

STEP-4: Generate all permutations of length N from the list L using the permutations() function from the itertools module.

Convert the result into a set to remove duplicates.

STEP-5: Convert each permutation (which is a tuple) into an integer and find the largest number using the max() function.

STEP-6: Calculate the total number of unique permutations using len() on the set of permutations.

STEP-7: Print the largest number formed and the total number of combinations.

STEP-8: End

VIVA QUESTIONS :

1. What is the difference between permutations and combinations?
2. How do you calculate the number of permutations of n objects taken r at a time? Provide the formula.
3. How does the formula for combinations differ from the formula for permutations?
4. What Python module can you use to calculate permutations and combinations, and how do you import it?
5. How would you use Python to calculate the number of permutations of 5 objects taken 3 at a time? Provide the code.

Experiment7: String & Regular Expressions

a.Write a python script to take two string S1 and S2 and do the following:

i) Check S1and S2 are anagrams or not.

ii) Check S1 is Substring of S2 or not.

iii) S1 is palindrome or not.

DESCRIPTION:

- This Python script takes two strings **S1** and **S2** as input.
- It first checks if **S1** and **S2** are **anagrams** by comparing the sorted versions of both strings.
- Next, it checks if **S1** is a **substring** of **S2** using the in operator.
- Finally, it checks if **S1** is a **palindrome** by comparing the string to its reverse.
- The script prints the results of all three checks: anagram, substring, and palindrome status.

ALGORITHM :

STEP-1: Start

STEP-2: Define a function is_anagram(s1, s2):

Sort both s1 and s2.

Return True if the sorted strings are equal, otherwise return False.

STEP-3: Define a function is_substring(s1, s2):

Return True if s1 is found within s2, otherwise return False.

STEP-4: Define a function is_palindrome(s1):

Return True if s1 is equal to its reverse (s1[::-1]), otherwise return False.

STEP-5: Input the first string s1 and the second string s2 from the user.

Remove spaces and convert both strings to lowercase.

STEP-6: Print Results:

Check if s1 and s2 are anagrams using is_anagram(s1, s2).

Check if s1 is a substring of s2 using is_substring(s1, s2).

Check if s1 is a palindrome using is_palindrome(s1).

STEP-7: End

b.Write a python script to take input as multi-line string and find the sum of all numbers in that string using re module. (Eg. S=”he11otheyare40studentsin97roomof4thline”,Sum=152)

ALGORITHM

STEP-1: Start

STEP-2: Input a multiline string containing both text and numbers (e.g., multiline_string).

STEP-3: Use Regular Expression (re.findall):

Extract all sequences of digits (\d+) from the string.

This function returns a list of numbers found in the string as strings.

STEP-4: Convert the list of string numbers to integers using map(int, numbers).

STEP-5: Calculate the sum of all the numbers using the sum() function.

STEP-6: Print the extracted numbers and their total sum.

STEP-7: End

VIVA QUESRTIONS:

1. What is a string in Python? How do you define a string in Python?
2. What are the common string methods available in Python for string manipulation? Provide examples of at least three methods.
3. How can you concatenate two strings in Python? Provide an example.
4. What is slicing in Python strings? How would you extract a substring from a string using slicing?
5. What is the difference between mutable and immutable objects in Python? Is a string mutable or immutable?

Experiment8: Lists & Dictionary

a.Write a program to take input as String S and print frequency of each character in S using List data structure.

DESCRIPTION:

- This Python script takes a multi-line string **S** as input, containing both letters and numbers.
- It uses the re (regular expression) module to find all the numbers within the string.
- The script extracts all the numbers using the regular expression pattern r'\d+' to match sequences of digits.
- It then converts the extracted numbers to integers and calculates their sum.
- Finally, the script prints the total sum of all the numbers found in the string.

ALGORITHM :

STEP-1: Start

STEP-2: Define function character_frequency(s):

Initialize two empty lists: characters (to store unique characters) and frequencies (to store the frequency of each corresponding character).

Iterate over each character char in the string s:

If char is not in characters, add char to the characters list and append 1 to the frequencies list.

If char is already in characters, find its index and increment its corresponding frequency in the frequencies list.

STEP-3: Input a string s from the user.

STEP-4: Call the character_frequency(s) function to calculate and print the frequencies of characters.

STEP-5: End

b.Write a program to take input as String S contains characters and special symbols, reverse the String S such that special symbols remains at same position. (Eg. S=”m@d#u” , Output=”u@d#m”)

DESCRIPTION :

- This Python script takes a string **S** as input, containing characters and special symbols.
- It extracts the letters and stores them in a list while skipping the special symbols.
- The script then reverses the list of letters and places them back in the original positions, leaving the special symbols in their original spots.
- It iterates through the string, replacing the alphabetic characters with the reversed ones while keeping special symbols unchanged.
- Finally, the script prints the modified string where only the characters are reversed, and special symbols remain in place.

ALGORITHM :

STEP-1: **Start**

STEP-2: Define function reverse_with_special_symbols(s):

Convert the input string s into a list of characters (s_list).

Create a list chars containing only alphanumeric characters from the string s (ignores special symbols).

Reverse the list chars to prepare for inserting characters in reverse order.

Initialize an empty list result and a variable char_index to track the position of characters in the reversed list.

Iterate through each character char in s_list:

If char is alphanumeric, append the next character from chars to result and increment char_index.

If char is not alphanumeric (i.e., a special symbol), append it directly to result.

After processing all characters, return the result as a string by joining the list result.

STEP-3: Input a string s with alphanumeric characters and special symbols.

STEP-4: Call the function reverse_with_special_symbols(s) to process the string.

STEP-5: Print the output string.

STEP-6: **End**

c.Write a python script to take input as String sentence S and print each word count using dictionary.

DESCRIPTION:

- This Python script takes a string **S** as input, which contains a sentence of words.
- It splits the sentence into individual words using the split() method.
- The script then creates a dictionary to store each word as a key and its count as the corresponding value.
- It iterates through the list of words and updates the dictionary with the frequency of each word.
- Finally, the script prints the dictionary, displaying each word and its count in the sentence.

ALGORITHM :

STEP-1: **Start**

STEP-2: Define function word_count(sentence):

Convert the input sentence to lowercase and split it into individual words.

Initialize an empty dictionary word_dict to store the frequency of each word.

Iterate through each word in the list of words:

If the word is already in word_dict, increment its count.

Otherwise, add the word to word_dict with a count of 1.

Print the word count by iterating through the dictionary word_dict.

STEP-3: **Input** a sentence from the user.

STEP-4: Call the function word_count(sentence) to calculate and display the word count.

STEP-5: End

VIVA QUESTIONS:

1. What is a list in Python, and how do you define a list? Provide an example.
2. How can you access elements in a list using indices? What happens if you try to access an index that is out of range?
3. What are some of the common methods available for lists in Python? Provide examples of at least three methods (e.g., append(), remove(), sort()).
4. How can you add an element to a list at a specific index in Python? Provide an example.
5. What is the difference between a list and a tuple in Python? Can you modify the elements of a tuple?

Experiment9: OOPS

a.Using Python OOPS, create a class, constructor, method,
_st_and _repr_for:

i. Employee

ii. Student

DESCRIPTION:

- This Python script defines two classes: **Employee** and **Student**.
- Each class includes a **constructor** (__init__) to initialize attributes like name, ID, and other relevant details.
- Both classes contain a method to display information about the employee or student, and the __str__ and __repr__ methods are used to define how objects should be represented as strings.
- The __str__ method provides a user-friendly string output, while __repr__ gives a more detailed, developer-friendly string for debugging.
- Finally, the script demonstrates creating objects of both classes and printing them to show how the string representation works.

ALGORITHM for employee:

STEP-1: Start

STEP-2: Define the Employee class:

Initialize method (__init__):

Define the constructor with parameters: name, id, and position.

Set the instance variables self.name, self.id, and self.position to the passed arguments.

Method to display employee details (display_details):

Return a string that displays the employee's ID, name, and position.

Override __str__ method:

Return a string representation of the Employee object in the form: "Employee(name, position)".

Override __repr__ method:

Return a detailed string representation of the Employee object in the form: "Employee(name='name', id=id, position='position')".

STEP-3: Create an instance of Employee:

Instantiate an employee object emp with a name, ID, and position.

STEP-4: Display employee details:

Call the display_details method to print the employee details.

STEP-5: Print string and representation of the object:

Print the string representation of emp by calling print(emp).

Print the detailed representation of emp by calling print(repr(emp)).

STEP-6: End

ALGORITHM for student:

STEP-1: Start

STEP-2: Define the Student class:

Initialize method (__init__):

Define the constructor with parameters: name, roll_no, and course.

Set the instance variables self.name, self.roll_no, and self.course to the passed arguments.

Method to display student details (display_details):

Return a string that displays the student's roll number, name, and course.

Override __str__ method:

Return a simple string representation of the Student object in the form: "Student(name, course)".

Override __repr__ method:

Return a detailed string representation of the Student object in the form: "Student(name='name', roll_no=roll_no, course='course')".

STEP-3: Create an instance of Student:

Instantiate a student object student with a name, roll number, and course.

STEP-4: Display student details:

Call the display_details method to print the student details.

STEP-5: Print string and representation of the object:

Print the string representation of student by calling print(student).

Print the detailed representation of student by calling print(repr(student)).

STEP-6: End

VIVA QUESTIONS:

1. What is Object-Oriented Programming (OOP)? Can you explain its basic concepts?
2. What is a class in Python? How do you define a class, and what is its purpose?
3. What is an object in Python? How is an object created from a class?
4. Can you explain the concept of encapsulation in OOP? How do you implement it in Python?
5. What is inheritance in Python? How does it help in code reuse? Provide an example.

Experiment10: Exceptions

a.Write a python program to implement Exceptions hierarchy

DESCRIPTION:

- This Python script demonstrates the creation of a custom exception hierarchy by defining a base exception class.
- It then defines multiple subclasses that inherit from this base exception class, each representing different types of errors.
- The script handles specific exceptions using try, except, and finally blocks to catch and manage errors at various levels of the hierarchy.
- It showcases how exceptions can be raised and caught using the custom hierarchy, allowing for fine-grained error handling.
- Finally, the script prints appropriate messages based on the type of exception raised, demonstrating how to manage different error conditions effectively.

ALGORITHM :

STEP-1: Start

STEP-2: Define custom exception classes:

Define MyBaseException:

This class inherits from the built-in Exception class.

Initialize it with a custom message and call the base class constructor with the message.

Define MyCustomException:

This class inherits from MyBaseException.

Initialize it with a custom message and call the base class constructor with the message.

Define AnotherCustomException:

This class also inherits from MyBaseException.

Initialize it with a custom message and call the base class constructor with the message.

STEP-3: Use the custom exceptions in a try-except block:

First try block:

Raise the Another Custom Exception with a specific error message.

In the except block, catch the MyBaseException and print the message associated with the exception.

Second try block:

Raise the My Custom Exception with a specific error message.

In the except block, catch the My Custom Exception and print the message associated with the exception.

STEP-4: End

b.Create a user defined Exception named "Funds Less Exception"and raise the exception when there are no enough funds in the bank account.

DESCRIPTION:

- This Python script defines a user-defined exception called **"FundsLessException"** that inherits from the base Exception class.
- The exception is raised when an attempted withdrawal from a bank account exceeds the available balance.
- The script uses a **BankAccount** class with methods for deposit, withdrawal, and balance checking.
- If a withdrawal request exceeds the balance, the **FundsLessException** is raised with a custom error message.
- Finally, the script handles the exception using a try-except block and prints a message indicating insufficient funds.

ALGORITHM :

STEP-1: Start

STEP-2: Define the custom exception class FundsLessException:

This class inherits from the built-in Exception class.

It initializes with an optional custom message ("Insufficient funds in the account" by default).

The constructor calls the base Exception class constructor to pass the message.

STEP-3: Define the Bank Account class:

Constructor (_init_):

Accept a parameter balance to set the initial balance of the bank account.

Initialize the account with the given balance.

Method withdraw(self, amount):

If the requested withdrawal amount is greater than the current balance, raise the FundsLessException with a custom message indicating insufficient funds.

Otherwise, subtract the withdrawal amount from the balance and print a success message with the updated balance.

STEP-4: Test the BankAccount class with exception handling:

First try-except block:

Create a bank account with an initial balance of 500.

Attempt to withdraw 600, which should raise the FundsLessException because the balance is insufficient.

Catch the exception and print the error message.

Second try-except block:

Create a bank account with an initial balance of 1000.

Attempt to withdraw 400, which should be successful.

Print the updated balance after the successful withdrawal.

STEP-5: End.

VIVA QUESTIONS:

1. What is an exception in Python, and why are exceptions used in programming?
2. How do you handle exceptions in Python using try and except? Provide a simple example.
3. What is the purpose of the else and finally blocks in exception handling? How are they used?
4. What happens if an exception occurs inside the try block, but there is no corresponding except block to catch it?
5. What is the difference between Exception and BaseException in Python? When should each be used?

Experiment11: Data Analysis

a.Using NumPy, implement different matrix operations in python.

DESCRIPTION:

- This Python script uses the **NumPy** library to perform various matrix operations such as addition, subtraction, multiplication, and division.
- It demonstrates creating matrices using np.array() and performing operations like element-wise addition, scalar multiplication, and matrix multiplication.
- The script also calculates the **transpose**, **determinant**, and **inverse** of matrices using np.transpose(), np.linalg.det(), and np.linalg.inv().
- Matrix operations are applied to demonstrate their mathematical properties and functions.
- Finally, the script prints the results of these operations to show how matrix calculations can be performed efficiently with **NumPy**.

ALGORITHM :

STEP-1: Start

STEP-2: Import the NumPy module:

Import NumPy as np to enable matrix operations.

STEP-3: Define two 2x2 matrices A and B:

Matrix A is defined as a 2x2 array [[1, 2], [3, 4]].

Matrix B is defined as a 2x2 array [[5, 6], [7, 8]].

STEP-4: Matrix Addition (A + B):

Add matrix A and matrix B element-wise to get the result matrix_addition.

Print the result of matrix addition.

STEP-5: Matrix Subtraction (A - B):

Subtract matrix B from matrix A element-wise to get the result matrix_subtraction.

Print the result of matrix subtraction.

STEP-6: Matrix Multiplication (A * B):

Use the np.dot() function to perform matrix multiplication between matrix A and matrix B to get the result matrix_multiplication.

Print the result of matrix multiplication.

STEP-7: Matrix Transposition (A^T):

Use .T to get the transpose of matrix A, resulting in matrix_transpose.

Print the result of matrix transposition.

STEP-8: Matrix Inversion (A^-1):

Use np.linalg.inv() to calculate the inverse of matrix A.

If A is invertible, print the result of the matrix inversion.

If A is not invertible (raises LinAlgError), print a message indicating that matrix A is not invertible.

STEP9: Element-wise Multiplication (A * B):

Perform element-wise multiplication between matrix A and matrix B using the * operator.

Print the result of element-wise multiplication.

STEP-10: End

b.**Using pandas, read the data from any text files.**

DESCRIPTION:

- This Python script uses the **NumPy** library to perform various matrix operations such as addition, subtraction, multiplication, and division.
- It demonstrates creating matrices using np.array() and performing operations like element-wise addition, scalar multiplication, and matrix multiplication.
- The script also calculates the **transpose**, **determinant**, and **inverse** of matrices using np.transpose(), np.linalg.det(), and np.linalg.inv().
- Matrix operations are applied to demonstrate their mathematical properties and functions. Finally, the script prints the results of these operations to show how matrix calculations can be performed efficiently with **NumPy**.

ALGORITHM :

STEP-1: Start

STEP-2: Import the pandas library:

Import pandas as pd to work with DataFrame operations.

STEP-3: Read the data from the file:

Read the data from a comma-separated values (CSV) file using pd.read_csv('data.txt') if the file contains comma-separated values (CSV).

Alternatively, if the file contains space-separated values, read the file using pd.read_csv('data.txt', delim_whitespace=True).

STEP-4: Display the DataFrame:

Print the DataFrame to display the data read from the file.

STEP-5: End

VIVA QUESTIONS:

1. What is data analysis, and how is it relevant in Python programming?
2. Which Python libraries are commonly used for data analysis? Can you explain the purpose of libraries like pandas, numpy, and matplotlib?
3. How do you import a CSV file into Python for data analysis? Provide an example using pandas.
4. What is a DataFrame in pandas, and how does it differ from a regular Python list or dictionary?
5. How can you access a specific column or row in a pandas DataFrame? Provide an example.

Experiment12: Plotting

a.Using Mat Plot lib library, plot the graph with all different plot types.(Pie Chart, Area Plot, Scatter Plot, Histogram and Bar Graph)

DESCRIPTION:

- This Python script uses the **Matplotlib** library to create and display various types of plots, including **Pie Chart**, **Area Plot**, **Scatter Plot**, **Histogram**, and **Bar Graph**.
- It generates a **Pie Chart** to show proportions, an **Area Plot** for cumulative data, and a **Scatter Plot** to represent data points on a two-dimensional graph.
- The script also creates a **Histogram** to display data distribution and a **Bar Graph** to visualize categorical data.
- Each plot type is created using specific Matplotlib functions such as plt.pie(), plt.plot(), plt.scatter(), plt.hist(), and plt.bar().
- Finally, the script uses plt.show() to display the plots in a single figure window

ALGORITHM

STEP-1: Import the required libraries:

Import matplotlib.pyplot as plt to create plots.

Import numpy as np to handle data generation and manipulation.

STEP-2 : Define Data for the Plots:

Define a list labels containing categories for the pie chart (['A', 'B', 'C', 'D']).

Define a list sizes containing numerical values for the pie chart ([25, 35, 20, 20]).

Define x as an array of values (e.g., x = np.arange(1, 6)).

Define y as a random array of integers (e.g., y = np.random.randint(1, 10, 5)).

Define height as random heights for the bar graph (e.g., height = np.random.randint(1, 10, 5)).

STEP-3: Create Subplots Layout:

Use plt.subplot(231) to define a 2x3 grid, and select the first position for the Pie Chart.

Use plt.subplot(232) for the second position in the grid for the Area Plot.

Use plt.subplot(233) for the third position in the grid for the Scatter Plot.

Use plt.subplot(234) for the fourth position in the grid for the Histogram.

Use plt.subplot(235) for the fifth position in the grid for the Bar Graph.

STEP-4: Plot the Pie Chart:

Call plt.pie() to generate the pie chart.

Pass the sizes and labels to plt.pie().

Set the autopct parameter for displaying the percentage and startangle for rotation.

Add a title using plt.title().

STEP-5: Plot the Area Plot:

Call plt.fill_between() to create the area plot.

Call plt.plot() to overlay the line on the area plot.

Add a title using plt.title().

STEP-6: Plot the Scatter Plot:

Call plt.scatter() to generate a scatter plot.

Add labels for the X and Y axes using plt.xlabel() and plt.ylabel().

Add a title using plt.title().

STEP-7: Plot the Histogram:

Call plt.hist() to generate a histogram.

Set the number of bins, color, and edge color.

Add labels for the X and Y axes using plt.xlabel() and plt.ylabel().

Add a title using plt.title().

STEP-8: Plot the Bar Graph:

Call plt.bar() to generate the bar graph.

Pass x and height to create the bars.

Set the width of the bars and add labels for the axes using plt.xlabel() and plt.ylabel().

Add a title using plt.title().

STEP9: Adjust Layout:

Use plt.tight_layout() to adjust the spacing between the plots and ensure they fit into the figure.

STEP-10: Display the Plots:

Call plt.show() to display all the plots.

VIVA QUESTIONS :

1. What is the purpose of using the matplotlib library in Python for plotting?
2. How do you plot a simple line graph using matplotlib.pyplot?
3. What is the difference between plot(), scatter(), and bar() functions in matplotlib?
4. How can you add labels to the x-axis and y-axis in a plot?
5. What is the role of the legend() function in a plot?