EX NO: 1.a **BASIC PROGRAM IN PYTHON**

AIM:

To write a Python program that prints “Hello, World!”

ALGORITHM:

Step 1:Start the program.

Step 2: Use the print() function in Python to display a message.

Step 3: Inside the print() function, write the string “Hello, World”

Step 4: Execute the program.

RESULT:

Thus the above basic python program is executed successfully and the output is verified.

EX NO : 1.b **SIMPLE ADDITION**

AIM:

To write a Python program that print the sum of two numbers

ALGORITHM:

Step 1:Start the program.

Step 2: Declare a variable a and assign the value 10 to it.

Step 3: Declare a variable b and assign the value 20 to it.

Step 4: Add a and b, and store the result in the variable sum.

Step 5: Use the print() function to display the message “Sum:”; followed by the value of sum.

Step 6: Stop the program.

RESULT:

Thus the above simple addition python program is executed successfully and the output is verified.

EX NO: 1.c **PROMPT INPUT FROM USER**

AIM:

To write a Python program that asks the user for their name and prints a greeting.

ALGORITHM:

Step 1:Start the program.

Step 2: Use input() to prompt the user to enter their name.

Step 3: Store the entered name in a variable.

Step 4: Use print() to display a greeting with the name.

Step 5: Stop the program.

RESULT:

Thus the above Prompt User Input python program is executed successfully and the output is verified.

EX NO : 1.d **BASIC CALCULATOR OPERATIONS**

AIM:

To write a Python program that takes two numbers as input and displays their sum, difference, product, and quotient.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter the first number using input() and convert it to an integer using int().

Step 3: Prompt the user to enter the second number using input() and convert it to an integer using int().

Step 4: Calculate and display the sum of the two numbers using print().

Step 5: Calculate and display the difference of the two numbers using print().

Step 6: Calculate and display the product of the two numbers using print().

Step 7: Calculate and display the quotient of the two numbers using print().

Step 8: Stop the program.

RESULT:

Thus the above basic calculator operations program is executed successfully and the output is verified.

EX NO : 1.e **AREA OF CIRCLE**

AIM:

To write a Python program that calculates and displays the area of a circle based on the given radius.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter the radius of the circle using input() and convert it to

a float using float().

Step 3: Calculate the area using the formula: area = Step 3:14159 × radius × radius.

Step 4: Display the area using the print() function.

Step 5: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 2.a **CHECK IF NUMBER IS ODD OR EVEN**

AIM:

To determine whether the given number is even or odd.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter a number.

Step 3: Convert the input to an integer.

Step 4: Use the modulus operator % to check if the number is divisible by Step 2:

Step 5: If num % 2 == 0, print "Even number".

Step 6: Else, print “Odd number”

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 2.b **PROGRAM TO FIND GREATEST AMONG THREE NUMBERS**

AIM:

To find and display the greatest number among three user-input numbers.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter three numbers: a, b, and c.

Step 3: Convert the inputs to integers.

Step 4: Check if a= b and & a= c.

Step 5: If true, print a as the greatest; else check b= a and b= c.

Step 6: If true, print b; otherwise, print c as the greatest.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 2.c **CHECK LEAP YEAR**

AIM:

To determine whether a given year is a leap year.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter a year.

Step 3: Convert the input to an integer.

Step 4: Check if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0).

Step 5: If the condition is true, print that it is s a leap year.

Step 6: Else, print that it is not a leap year.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 2.d **CHECK THE SIGN OF A NUMBER**

AIM:

To check if a number is positive, negative, or zero.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter a number.

Step 3: Convert the input to a float.

Step 4: If the number is greater than 0, print Positive number.

Step 5: Else if the number is equal to 0, print Zero.

Step 6: Else, print Negative number.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 2.e **SIMPLE GRADING SYSTEM**

AIM:

To write a python program for calculating grade based on the student marks.

ALGORITHM:

Step 1:Start the program.

Step 2: Prompt the user to enter their marks.

Step 3: Convert the input to an integer.

Step 4: If marks ≥ 90, assign grade A else if ≥ 80, assign B &

Step 5: Else if marks ≥ 70, assign grade C else if ≥ 60, assign D

Step 6: If marks less than 60, assign grade F and print the grade.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 3.a **FINDING A FACTORIAL NUMBER USING FUNCTION**

AIM:

To compute the factorial of a given number using a user-defined function.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a function factorial(n) to calculate factorial.

Step 3: Initialize result = 1.

Step 4: Use a loop from 1 to n and multiply result by each value.

Step 5: Return the final result.

Step 6: Accept input from the user, call the function, and print the result.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 3.b **FINDING PRIME NUMBER USING FUNCTION**

AIM:

To check whether a number is prime using a function.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a function is_prime(n) to check if a number is prime.

Step 3: If n = 1, return False.

Step 4: Use a loop from 2 to √n to check for divisibility.

Step 5: If any number divides n, return False; else return True.

Step 6: Get input from user, call the function, and print result.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 3.c **FINDING GREATEST COMMON DIVISOR USING FUNCTION**

AIM:

To find the GCD of two numbers using a function.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a function gcd(a, b) using a while loop.

Step 3: Loop while b is not zero.

Step 4: Update values: a, b = b, a % b.

Step 5: When loop ends, return a as the GCD.

Step 6: Get two numbers from the user, call the function, and print the result.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 3.d **FINDING FIBONACCI SERIES USING FUNCTION**

AIM:

To generate the Fibonacci series up to n terms using a function.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a function fibonacci(n) to print series.

Step 3: Initialize two variables: a = 0, b = 1.

Step 4: Use a loop to print n terms: print a, then update a = b, b = a + b.

Step 5: Get the number of terms from the user.

Step 6: Call the function to display the series.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 3.d **SUM OF DIGITS USING FUNCTION**

AIM:

To calculate the sum of digits of a number using a function.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a function sum_of_digits(n) to add digits.

Step 3: Initialize total = 0.

Step 4: Use a while loop: add n % 10 to total.

Step 5: Remove last digit using n = n // 10.

Step 6: Repeat until n becomes 0, then return total and print it.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 4.a **STACK IMPLEMENTATION USING LIST**

AIM:

To implement stack operations (push and pop) using a Python list.

ALGORITHM:

Step 1:Start the program.

Step 2: Create an empty list stack = [].

Step 3: Use append() to push elements like 10, 20, and 30.

Step 4: Display the stack after push operations.

Step 5: Use pop() to remove the top element from the stack.

Step 6: Display the popped element and the updated stacK

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 4.b **QUEUE IMPLEMENTATION USING LIST**

AIM:

To implement queue operations using a Python list.

ALGORITHM:

Step 1:Start the program.

Step 2: Create an empty list queue = [].

Step 3: Use append() to enqueue elements 10, 20, and 30.

Step 4: Display the queue after enqueue operations.

Step 5: Use pop(0) to dequeue the front element of the queue.

Step 6: Display the dequeued element and the updated queue.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 4.c **STACK OPERATION WITH USER INTERACTION**

AIM:

To implement a menu-driven stack using list with push, pop, and display operations.

ALGORITHM:

Step 1:Start the program and initialize an empty list stack = [].

Step 2: Continuously display a menu with choices: Push, Pop, Display, Exit.

Step 3: On choosing Push, get input from the user and append() it to the stack.

Step 4: On choosing Pop, check if stack is empty; if not, pop() and show the top element.

Step 5: On choosing Display, print the current stack contents.

Step 6: If Exit is selected, break the loop.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 4.d **QUEUE OPERATION WITH USER INTERACTION**

AIM:

To implement a menu-driven queue using list with enqueue, dequeue, and

display operations.

ALGORITHM:

Step 1:Start the program and initialize an empty list queue = [].

Step 2: Continuously display a menu with choices: Enqueue, Dequeue, Display, Exit.

Step 3: On choosing Enqueue, get input from the user and append() it to the queue.

Step 4: On choosing Dequeue, check if queue is empty; if not, use pop(0) to remove

the front item.

Step 5: On choosing Display, print the current queue contents.

Step 6: If Exit is selected, break the loop.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO: 5.a **WORKING WITH TUPLES**

AIM:

To demonstrate creation, indexing, slicing, and length calculation of tuples.

ALGORITHM:

Step 1:Start the program.

Step 2: Create a tuple my_tuple with sample values.

Step 3: Access and print the first element using index my_tuple[0].

Step 4: Perform slicing to extract a sub-part my_tuple[1:3].

Step 5: Use len() function to find the length of the tuple.

Step 6: Display all results.

Step 7: Stop the program

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 5.b **WORKING WITH LIST**

AIM:

To perform indexing, slicing, and reversing operations on a list.

ALGORITHM:

Step 1:Start the program.

Step 2: Create a list my_list with integer elements.

Step 3: Access and print the second element using index my_list[1].

Step 4: Slice the list from the third element onward using my_list[2:].

Step 5: Reverse the list using slicing my_list[::-1].

Step 6: Display all results.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 5.c **WORKING WITH SET**

AIM:

To demonstrate union, intersection, and difference of sets.

ALGORITHM:

Step 1:Start the program.

Step 2: Create two sets set1 and set2

Step 3: Perform union operation and display the result.

Step 4: Perform intersection and display the result.

Step 5: Perform difference operation set1 - set2 and display the result.

Step 6: Show all outputs to the user.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 5.d **WORKING WITH DICTIONARIES**

AIM:

To demonstrate dictionary creation, access, update, and iteration.

ALGORITHM:

Step 1:Start the program.

Step 2: Create a dictionary student with keys like name, age, course.

Step 3: Access and print the value of the “name” key.

Step 4: Add a new key-value pair: grade: A.

Step 5: Use a for loop to iterate through all items in the dictionary.

Step 6: Print each key and value in formatted output.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :6.a **CLASS AND OBJECT**

AIM:

To create a class with a constructor and method, and access it using an object.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a class Student with a constructor _init_ to initialize name and age.

Step 3: Define a method display() to print the student name and age.

Step 4: Create an object s1 of class Student with sample data.

Step 5: Call the display() method using the object.

Step 6: Print the values as output.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO: 6.b **CLASS WITH METHODS AND CONSTRUCTOR**

AIM:

To calculate the area of a rectangle using a class with a constructor and method.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a class Rectangle with a constructor to initialize length and breadth.

Step 3: Define a method area() that returns length * breadth.

Step 4: Create an object r of class Rectangle with sample dimensions

Step 5: Call the area() method using the object.

Step 6: Print the area as output.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 6.c **SINGLE INHERITANCE**

AIM:

To demonstrate single inheritance using a base class and a derived class.

ALGORITHM:

Step 1:Start the program.

Step 2: Define a base class Person with a constructor to initialize name and method

show().

Step 3: Define a derived class Student inheriting from Person, with an additional

attribute course.

Step 4: Use init_() to call base class constructor from the derived class.

Step 5: Define method show_course() to display course.

Step 6: Create an object s of the derived class and call both methods.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :6.d **MULTIPLE INHERITANCE**

AIM:

To demonstrate multiple inheritance using three classes.

ALGORITHM:

Step 1:Start the program.

Step 2: Define class Father with method skills() printing skills.

Step 3: Define class Mother with method skills() printing skills.

Step 4: Define class Child inheriting from both Father and Mother.

Step 5: In Child, override the skills() method and call Father.skills(self) and Mother.skills(self).

Step 6: Add additional child skills and display all.

Step 7: Stop the program

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :6.e **MULTILEVEL INHERITANCE**

AIM:

To demonstrate multilevel inheritance using three classes in hierarchy.

ALGORITHM:

Step 1:Start the program.

Step 2: Define base class Animal with method sound().

Step 3: Create class Dog derived from Animal with method bark().

Step 4: Create class Puppy derived from Dog with method weep().

Step 5: Create an object p of the Puppy class.

Step 6: Call all three methods using the object p.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :7.a **WRITING TO FILE**

AIM:

To write multiple lines of text to a file using Python.

ALGORITHM:

Step 1:Start the program.

Step 2: Open a file.

Step 3: Use the write() method to write two lines to the file.

Step 4: Close the file using file.close().

Step 5: Print a confirmation message.

Step 6: Check the file to verify the content.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO : 7.b **READING FROM FILE**

AIM:

To read and display the content of a file using Python.

ALGORITHM:

Step 1:Start the program.

Step 2: Open the file example.txt in read mode.

Step 3: Read the entire content using read() method.

Step 4: Store the content in a variable.

Step 5: Print the content.

Step 6: Close the file using file.close().

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :7.c **READING FILE LINE BY LINE**

AIM:

To read a file one line at a time using a loop.

ALGORITHM:

Step 1:Start the program.

Step 2: Open the file using with open() in read mode.

Step 3: Use a for loop to iterate over each line.

Step 4: Strip extra spaces or newline characters using strip().

Step 5: Print each line one by one.

Step 6: File automatically closes after the block ends.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO :7.d **APPENDING TO FILE**

AIM:

To add new lines to an existing file without deleting its content.

ALGORITHM:

Step 1:Start the program.

Step 2: Open the file example.txt in append mode using with open().

Step 3: Use write() to add a new line to the file.

Step 4: Automatically close the file using with block.

Step 5: Print a success message.

Step 6: Check the file to confirm the new line was added.

Step 7: Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO: 7.e WRITING AND READING NUMBERS

AIM:

To writing and reading a number in a file using Python.

ALGORITHM:

Step 1: Start the program.

Step 2: Open a file named "numbers.txt" in write mode ("w").

Step 3: Use a loop to write numbers from 1 to 5 into the file, each on a new line.

- Convert each number to a string before writing.
- Add a newline character \n after each number.

Step 4: Close the file automatically using the with statement.

Step 5: Open the "numbers.txt" file again, but this time in read mode ("r").

Step 6: Read each line from the file and print the number after stripping the newline character.

Step 7: Stop the program.

Result:

Thus, the above program Writing and Reading Numbers from a File is executed successfully and the output is verified.

EX NO:8.a BUILT-IN MODULE

AIM:

To learn how to use the built-in Python math module and utilize its functions and constants.

ALGORITHM:

1. Start the program.
2. Import the math module.
3. Use the sqrt() function to calculate the square root of 1Step 6:
4. Access the constant pi from the math module.
5. Print the square root of 1Step 6:
6. Print the value of pi.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:8.b **CREATING AND USING A CUSTOM MODULE**

AIM:

To create a custom Python module and use its functions by importing it in another program.

ALGORITHM:

1. Start the program.
2. Create a file named mymodule.py.
3. Define a function greet(name) inside mymodule.py that prints a greeting message.
4. In another file, import the custom module mymodule.
5. Call the function greet() from mymodule with a name argument (e.g., "Alice").
6. Observe the greeting message printed.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:8.c **PACKAGES**

AIM:

To learn how to organize modules into packages and import modules from a package.

ALGORITHM:

1. Start the program.
2. Create a folder named mypackage and add an empty __init__.py file inside it.
3. Inside mypackage, create two files: mod1.py and modStep 2:py.
4. Define a function hello() in mod1.py that prints a message.
5. Define a function info() in modStep 2:py that prints a message.
6. In the main program, import mod1 and mod2 from mypackage and call their respective functions.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:9.a **EXCEPTION HANDLING BY ZERO DIVISION ERROR**

AIM:

To handle division by zero error in Python using exception handling.

ALGORITHM:

1. Start the program.
2. Initialize variable a with 10 and b with 0.
3. Use a try block to perform division a / b.
4. Use an except block to catch ZeroDivisionError.
5. Print an error message if division by zero occurs.
6. Continue program execution without crashing.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:9.b **HANDLING MULTIPLE EXCEPTIONS**

AIM:

To handle multiple exceptions such as division by zero and invalid input errors using multiple except blocks.

ALGORITHM:

1. Start the program.
2. Take user input and convert it to integer inside a try block.
3. Attempt to divide 10 by the user input.
4. Catch ZeroDivisionError if user enters zero and print an appropriate message.
5. Catch ValueError if user enters invalid input and print an error message.
6. Ensure the program handles these exceptions gracefully.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:9.c **EXCEPTION HANDLING FOR INPUT VALIDATION**

AIM:

To demonstrate the use of else and finally blocks along with exception handling for input validation.

ALGORITHM:

---

1. Use a try block to take integer input from the user.
2. Raise a ValueError manually if the input number is negative.
3. Catch the ValueError and print the error message inside the except block.
4. Use an else block to print the entered number if no exception occurs.
5. Use a finally block to print the completion message regardless of exceptions.
6. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:10.a **MATH MODULE**

AIM:

To use the math module for performing mathematical operations like calculating square root, factorial, and accessing constants.

ALGORITHM:

1. Start the program.
2. Import the math module.
3. Use math.sqrt() to find the square root of 49.
4. Use math.factorial() to calculate factorial of Step 5:
5. Access the constant math.e (Euler's number).
6. Print all the results.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:10.b **RANDOM MODULE**

AIM:

To use the random module to generate random numbers and select random elements from a list.

ALGORITHM:

1. Start the program.
2. Import the random module.
3. Generate a random integer between 1 and 10 using random.randint().
4. Select a random element from a list using random.choice().
5. Print the generated random number and selected choice.
6. Observe the randomness in outputs.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO:10.c **DATETIME MODULE**

AIM:

To use the datetime module to retrieve and manipulate current date and time information.

ALGORITHM:

1. Start the program.
2. Import the datetime module.
3. Get the current date and time using datetime.datetime.now().
4. Extract the year and month from the current date and time object.
5. Print the current date and time, year, and month.
6. Understand how to access date/time attributes.
7. Stop the program.

RESULT:

Thus the above program is executed successfully and the output is verified.

EX NO: 11 **CREATING VIRTUAL ENVIRONMENTS AND MANAGING PACKAGES WITH PIP**

AIM:

To learn how to create a Python virtual environment and manage packages using pip.

ALGORITHM:

1. Start the procedure.
2. Open the command prompt (Windows) or terminal (Linux/Mac).
3. Navigate to your project directory using cd path_to_your_project_directory.
4. Create a virtual environment using python -m venvmyenv.
5. Activate the virtual environment:
    - On Windows: myenv\Scripts\activate
6. Install required packages inside the virtual environment using pip install package_name (e.g., pip install requests).
7. Deactivate the virtual environment after use with the command deactivate.
8. Stop the program.

RESULT:

Thus the above creating virtual Environments and managing packages is created successfully.

EX NO: 12 **INSTALLING AND EXPLORING NUMPY, PANDAS & MATPLOTLIB FOR REAL WORLD PROBLEMS**

AIM:

To install and explore the basic features of NumPy, Pandas, and Matplotlib libraries used for solving real-world data problems.

ALGORITHM:

1. Start the procedure.
2. Open Command Prompt
3. Verify Python and pip installation by typing python --version and pip --version.
4. Install NumPy, Pandas, and Matplotlib libraries using:

   pip install numpy pandas matplotlib
5. Verify the installation by running pip list and checking the libraries are listed.
6. Create a NumPy array and calculate mean and standard deviation.
7. Create a Pandas DataFrame and calculate average age.
8. Plot a simple line graph using Matplotlib.
9. Stop the program.

RESULT:

Thus the above installation process and python code for packages are executed successfully and the output is verified.