

VIT<sup>®</sup>  
B H O P A L  
[www.vitbhopal.ac.in](http://www.vitbhopal.ac.in)

**Fall Semester 2025-26**

**Project Report**

**Project Title:** Scientific Calculator

**Student Name:** Arpit Singh

**Registration Number:** 25BCE10726

**Course:** Introduction to Problem Solving and Programming.

**Course Instructor:** Dr.Preetam Suman

**Slot:**C14+E11+E12

## **1. Introduction:-**

This is the program for a scientific calculator that has been developed using python. It is a command line interface based program that takes input from the user and performs the necessary calculations and displays the output. The primary utility of the program lies in the combination of the variety of functions that it offers ranging from basic arithmetic to scientific to data visualization tools.

## **2. Problem Statement:-**

Modern scientific work and data analysis often require switching between different tools: a standard calculator for arithmetic calculations, a scientific calculator for complex functions (like log or trig. etc), and specialized devices or software for basic charting.

The problem this project addresses is the lack of a unified, utility that efficiently combines the three core functionalities—algebraic calculations, scientific functions, and simple data visualization—into one interactive program thereby streamlining the workflow for students, engineers, and analysts.

### **3. Functional Requirements:-**

The system must perform the following functions accurately:

#### **1. User Interface:**

- Accept the user's name upon startup.
- Display a clear, categorized menu of available functions (Algebraic, Scientific, Data Visualization).
- Accept user choice for the desired function (e.g., Addition, Pie Chart, Factorial,etc).

#### **2. Algebraic Functions:**

- Addition, Subtraction, Multiplication, Division.
- Random Number Generator (within a specified lower and upper limit).

#### **3. Scientific Functions:**

- Exponential, Logarithm, Factorial.
- Fibonacci sequence generation (up to a specified term).
- Trigonometric functions: sine, cosine, tangent, cosecant, secant, cotangent.

#### **4. Data Visualization Functions (requires matplotlib):**

- Numerical Line Graph: Plotting a series of numerical data points.
- Bar Graph: Plotting categorical data with associated values.
- Pie Chart: Visualizing proportional data, requiring labels, sizes, and optional colors.

#### **5. Input Handling:**

- Prompt the user for all necessary parameters for the chosen function (e.g., numbers for arithmetic, list of values for charts,etc).

## 4. Non-functional Requirements:-

1. **Usability:** The command-line interface must be intuitive, with clear instructions.
2. **Performance:** Calculations and graph generation must complete in near real-time.
3. **Reliability:** Mathematical functions must return mathematically correct results and the data visualization tools must accurately work.
4. **Extensibility:** The code structure should allow for easy addition of new functions (e.g. statistical tools) in the future.
5. **Dependencies:** The application must rely only on widely available Python packages (math, random, matplotlib).

## 5. System Architecture:-

The system employs a Command-Line Interface in which the user provides the input and the output is displayed on the screen.

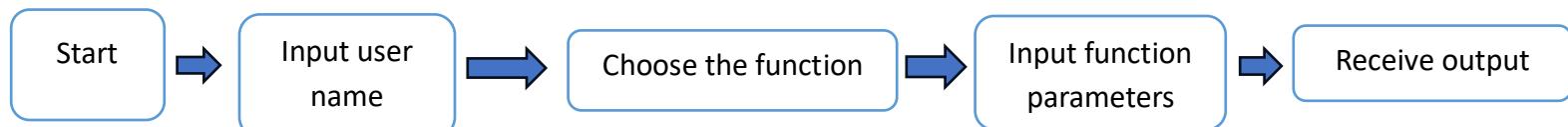
### Structure:

1. **Initialization:** The program starts, inputs the user name, greets the user, and initializes variables (like the user's name).
2. **Function Loop:** The main menu is presented and the program waits for the user's function choice. A simple loop asks for a valid function choice.
3. **Function Code blocks:** Based on the user's choice, the control flow is directed to a specific function code block (e.g. Factorial, Pie Chart, Logarithim,etc).
4. **Execution & Output:** The chosen function code executes, handles necessary data input, performs the core logic, and prints the result or displays a graph.

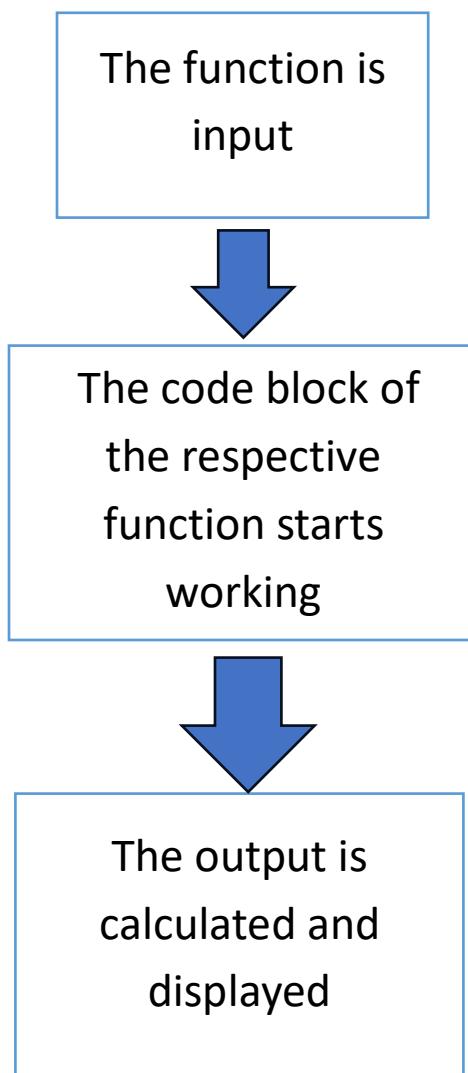
The system is designed for single user use only.

## 6.Design Diagram:-

### 1 Use Case Diagram:-



### 2. Work flow diagram:-



## 7. Design Decisions & Rationale:-

Aspect	Decision	Rationale
Language & Environment	Python (CLI)	Rapid prototyping, excellent built-in math and scientific libraries, and ease of access.
Function Organization	Categorical Menu (Algebraic, Scientific,etc)	Improves user experience and navigation by logically grouping related features, making the system easier to understand and navigate.
Data Visualization	Use of matplotlib.pyplot	matplotlib is the standard library for plotting in Python, offering robust and customizable charting features for the three required graph types.
Inputs for functions.	Sequential input() prompts	Keeps the command-line interface simple and direct, guiding the user step-by-step through the required inputs for complex functions like charts.
Error Handling (Basic)	Conditional checks (e.g. if lower_limit > upper_limit for Random Generator etc)	Provides immediate, user-friendly feedback for invalid logical inputs, improving reliability.

# 8. Implementation Details

The core implementation is a python (.py) file which runs sequentially.

1. **Libraries:** math, random, and matplotlib.pyplot are imported for specialized functionality.
2. **Scientific Functions:** Trigonometric functions use the math library, expecting input in radians.
3. **Visualization Implementation:** The data visualisation functions dynamically collect various inputs like values, names, sizes, colors, etc from the user via the input() function, then display the data visually.
4. **User Input:** All user input is captured using the input() function, with type conversion (float, int, list,etc) performed immediately after collection.

# 9. Screenshots / Results:-

The screenshot shows a Jupyter Notebook interface running on a Windows operating system. The title bar indicates the browser is Microsoft Edge, and the URL is `localhost:8888/notebooks/Project-1%20Scientific%20Calculator%20.ipynb`. The notebook tab is titled "Project-1 Scientific Calculator".

The code cell contains the following Python script:

```
186     plt.show()
187
188 elif ans=="Random number generator" or ans=="random number generator":
189     lower_limit=float(input('The random number should be greater than:'))
190     upper_limit=float(input('The random number should be lesser than:'))
191     if lower_limit>upper_limit:
192         print('The lower limit, cannot be greater than the upper limit of the range. Please re-enter the values.')
193     elif lower_limit==upper_limit:
194         random_num=random.uniform(lower_limit,upper_limit)
195         print('The random number is:',random_num)
```

The output cell shows the following interaction:

```
Please Enter your name : Medhavrat Saini
Nice to meet you, Medhavrat Saini
Choose a Function from the following --->
Algebraic-->
Addition, Subtraction, Multiplication, Division, Random number generator.

Scientific-->
Exponential, Logarithim, Factorial, Fibonacci, sine(), cosine(), tangent(), cosecant(), secant(), cotangent().

Data Visualization-->
Numerical Line Graph, Bar Graph, Pie Chart.
Logarithim
Enter the argument of logarithim (+ve real no.): 57
Enter the base of logarithim (+ve real no. other than 1): 10
The Logarithim of 57 with the base 10 is 1.755874856724912
```

The screenshot shows a Jupyter Notebook interface running on a Windows operating system. The title bar indicates the notebook is titled "Project-1 Scientific Calculator" and is located at "localhost:8888/notebooks/Project-1%20Scientific%20Calculator%20.ipynb". The notebook content displays a Python script for a scientific calculator. The script includes a function to generate a random number between two user-specified limits. It also provides a menu for various mathematical operations: Algebraic (Addition, Subtraction, Multiplication, Division), Scientific (Exponential, Logarithm, Factorial, Fibonacci, sine(), cosine(), tangent(), cosecant(), secant(), cotangent()), and Data Visualization (Numerical Line Graph, Bar Graph, Pie Chart). A Fibonacci sequence generator is also included. The user has entered their name as "Medhavrat Saini" and chosen to see the first 8 terms of the Fibonacci sequence.

```
ans== random number generator or ans== random number generator:
lower_limit=float(input('The random number should be greater than:'))
upper_limit=float(input('The random number should be lesser than:'))
if lower_limit>upper_limit:
    print('The lower limit, cannot be greater than the upper limit of the range. Please re-enter the values.')
elif lower_limit==upper_limit:
    random_num=random.uniform(lower_limit,upper_limit)
    print('The random number is:',random_num)

Please Enter your name : Medhavrat Saini
Nice to meet you, Medhavrat Saini
Choose a Function from the following --->

Algebraic-->
Addition, Subtraction, Multiplication, Division, Random number generator.

Scientific-->
Exponential, Logarithim, Factorial, Fibonacci, sine(), cosine(), tangent(), cosecant(), secant(), cotangent().

Data Visualization-->
Numerical line Graph, Bar Graph, Pie Chart.
Fibonacci
Enter the number of terms you would like to see: 8
0, 1, 1, 2, 3, 5, 8, 13,
```

This screenshot shows the same Jupyter Notebook interface as the previous one, but with a different timestamp in the bottom right corner (9:47 PM) and a different date (11/23/2025). The notebook content is identical, displaying the same Python script for the scientific calculator, the user's name "Medhavrat Saini", and the requested Fibonacci sequence of 13 terms. The output shows the sequence starting from 0 up to 13.

```
ans== random number generator or ans== random number generator:
lower_limit=float(input('The random number should be greater than:'))
upper_limit=float(input('The random number should be lesser than:'))
if lower_limit>upper_limit:
    print('The lower limit, cannot be greater than the upper limit of the range. Please re-enter the values.')
elif lower_limit==upper_limit:
    random_num=random.uniform(lower_limit,upper_limit)
    print('The random number is:',random_num)

Please Enter your name : Medhavrat Saini
Nice to meet you, Medhavrat Saini
Choose a Function from the following --->

Algebraic-->
Addition, Subtraction, Multiplication, Division, Random number generator.

Scientific-->
Exponential, Logarithim, Factorial, Fibonacci, sine(), cosine(), tangent(), cosecant(), secant(), cotangent().

Data Visualization-->
Numerical line Graph, Bar Graph, Pie Chart.
Factorial
Enter the Number: 13
The factorial of the given number is: 6227020800
```



## 10. Testing Approach:-

The testing approach was primarily focused on validating functional correctness across all three categories.

- **Numerical Testing (Algebraic & Scientific):**

**Test Cases:** Test cases were run against known correct values (e.g.,  $\sin(\pi/2)=1$ ,  $5!=120$ ,  $\log(10)=2.302\dots$ , etc).

**Boundary Cases:** Boundary conditions were tested, such as division by zero, log of zero/negative (expected error), and negative numbers for Factorial.

- **Data Visualization testing:**

The visualization functions were tested to ensure the input data was correctly passed to the matplotlib library and resulted in the desired visual output (e.g., confirming the correct number of slices in a pie chart and accurate percentage calculations,etc).

- **User Interaction Testing:**

Testing the menu selection to ensure the correct function is called for each input string, including case-insensitive matching.

## 11. Challenges Faced:-

1. **Correct Input for Visualization:** The challenge was robustly handling space-separated lists of numerical and string data (names, sizes, colors, etc) via the single `input()` command, requiring careful use of `split()` and `map()` functions.
2. **Trigonometric Units:** Python's math library works in radians, which is often counter-intuitive for users expecting degrees. Therefore the input is first taken in degrees and then converted to radians and then the final value of the trigonometric function is calculated.

3. **Function input loop:** If an incorrect function was input, a message is displayed asking the user for a correct input and then input is asked again.

## 12. Learnings & Key Takeaways:-

- **Procedural Design:** The project reinforced the principles of writing procedural Python code with clear conditional logic (if/elif/else) for directing program flow.
- **Library Usage:** Gained practical experience in easily using standard libraries (math, numpy, etc) into a unified program.
- **User-Centric Input:** Learned the importance of clear instructions when gathering multiple pieces of data in a commandline especially for data structures like lists and arrays.

## 13. Future Enhancements:-

1. **Graphical User Interface:** Migrate the application from command line interface to a graphical user interface for enhanced usability.
2. **Statistics Module:** Add functions for common statistical analysis such as mean, median, standard deviation, variance, etc.
3. **Complex Numbers:** Implement support for calculations involving complex numbers.
4. **Degree/Radian Toggle:** Introduce a feature allowing the user to select whether trigonometric inputs should be interpreted as degrees or radians.
5. **Robust Error Handling:** Implement more formal method to solve runtime errors (like ValueError for invalid input types) rather than relying solely on logical checks.

## 14. References:-

- Python Standard Library Documentation (for math and random modules).

- Matplotlib Documentation (for Pie Chart, Bar Graph, and Line Graph implementation).

