*Python Projects: Mathematical Computations, Graphs, and GUIs*

**SUBMITTED BY: ANANYA BORTHAKUR**

**ROLL NO: CS23BCAGN043**

**SUBMITTED TO: MR.NAWAB WASIM RAHMAN**

**THE ASSAM KAZIRANGA UNIVERSITY**

**DATE:19/05/2025**

**Acknowledgement**

I would like to sincerely thank Mr. Nawab Wasim Rahman for his invaluable guidance and continuous support throughout the preparation of this Python Projects Report. His insightful feedback and encouragement have been essential in enhancing my understanding and successfully completing the assignments. I deeply appreciate his dedication to fostering a productive learning environment and providing expert advice that greatly contributed to this work.

**Declaration**

I hereby declare that this Python Projects Report is my own original work. It has not been submitted to any other institution or for any other purpose. All sources and references used in compiling this report have been properly acknowledged and cited.

**TABLE OF CONTENTS**

Acknowledgment

Declaration

Abstract

Introduction

* Why this project matters

1. Objective
2. Software & Tools Used
3. Program 1: Arithmetic and Quadratic Operations
4. Program 2: Linear Equation Solver
5. Program 3: Graph Plotting
6. Program 4: Python Function
7. Program 5: GUI Application using Tkinter
8. Conclusion

**Abstract**

This report documents a series of Python programming projects focusing on mathematical computations, graph plotting, recursive functions, and GUI application development using Tkinter. Each project demonstrates Python’s capability to solve arithmetic and algebraic problems, visualize data through graphs, implement recursive algorithms, and build user-friendly graphical interfaces. The objective is to enhance problem-solving skills while providing practical experience in applying Python to varied tasks such as equation solving, data visualization, and interface design. Emphasizing the use of Spyder IDE within the Anaconda distribution, the report highlights Python’s versatility and effectiveness in scientific computing and software development, making it a valuable learning resource for students and academics interested in mathematical and application-oriented programming.

Introduction

Python is a high-level, interpreted programming language renowned for its simplicity, readability, and versatility. Its design philosophy emphasizes code clarity and ease of learning, making it ideal for both beginners and experienced developers. This lab assignment aims to leverage Python’s strengths to solve various mathematical problems and develop simple applications. The key tasks undertaken include performing arithmetic operations, solving linear and quadratic equations, visualizing mathematical functions via graph plotting, implementing recursive algorithms such as factorial calculation, and creating basic graphical user interfaces (GUIs) using the Tkinter library.

All coding exercises in this project were conducted using the **Spyder Integrated Development Environment (IDE)**, which is part of the widely adopted **Anaconda distribution**. This environment is especially popular in scientific computing and data science communities due to its powerful debugging, testing, and code management features that facilitate efficient and organized programming workflows.

Q.Why This Project Matters:

This project is not just about learning to write code. It teaches how to:

* Solve problems step by step.
* Write clean and readable programs.
* Use Python libraries like NumPy, Matplotlib, and Tkinter.
* Connect math with coding in a creative way.

## Objective

The main objectives of this project report are to:

* Implement basic arithmetic and algebraic expressions using Python.
* Solve and visualize linear and quadratic equations with graphical plots.
* Understand and apply recursive functions, such as calculating factorials.
* Create simple GUI applications leveraging the Tkinter module.
* Present these implementations clearly and systematically within a comprehensive report format.

## Software & Tools Used

This project utilized several key software and tools to facilitate effective development and execution of Python programs focused on mathematical computations and GUI applications.

* **Anaconda Distribution:** A comprehensive platform designed for scientific computing and data science, providing a managed environment with numerous pre-installed libraries and tools.
* **Spyder IDE:** An integrated development environment tailored for scientific Python programming, offering powerful code editing, debugging, and testing capabilities to streamline development.
* **Python 3.x:** The primary programming language employed, known for its clear syntax and extensive standard library supporting mathematical operations and application development.
* **Matplotlib & NumPy:** Essential libraries used respectively for plotting mathematical graphs and performing efficient numerical computations and array manipulations.
* **Tkinter:** Python’s standard GUI library, used to develop simple graphical user interfaces that enhance user interaction.
* **GitHub (optional):** A platform for version control and collaborative code management, facilitating project sharing and maintenance.

## Program 1: Arithmetic and Quadratic Operations

**CODE:-**

**import math  # Import math module for square root**

**# Variables for arithmetic  
a = 10  
b = 5**

**# Performing basic arithmetic operations  
print("Arithmetic Operations ")  
print(f"{a} + {b} = {a + b}")       # Addition  
print(f"{a} - {b} = {a - b}")       # Subtraction  
print(f"{a} \* {b} = {a \* b}")       # Multiplication  
print(f"{a} / {b} = {a / b if b != 0 else 'Undefined'}")  # Division with zero check**

**# Solving a quadratic equation of the form ax^2 + bx + c = 0  
print("\n Quadratic Equation Solver ")  
a = 1  
b = -3  
c = 2**

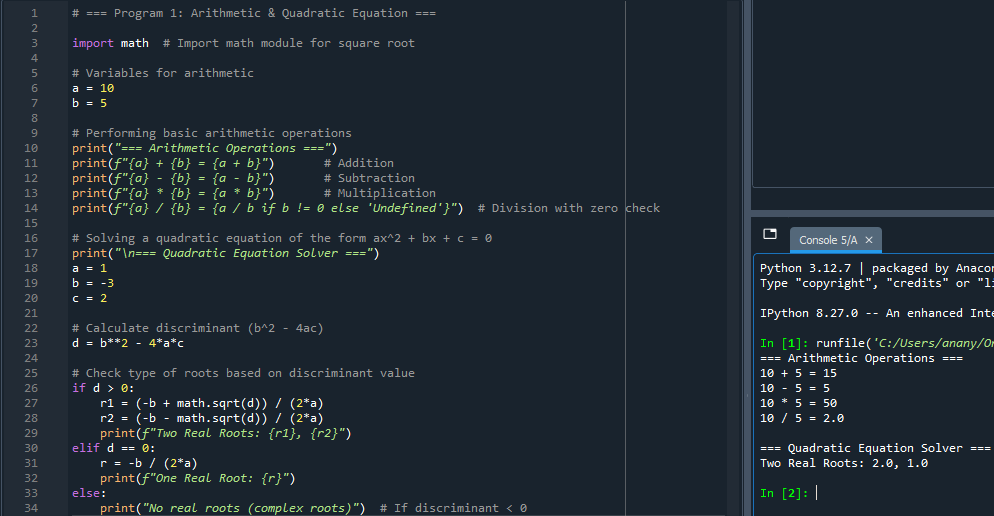
**# Calculate discriminant (b^2 - 4ac)  
d = b\*\*2 - 4\*a\*c**

**# Check type of roots based on discriminant value  
if d > 0:  
   r1 = (-b + math.sqrt(d)) / (2\*a)  
   r2 = (-b - math.sqrt(d)) / (2\*a)  
   print(f"Two Real Roots: {r1}, {r2}")  
elif d == 0:  
   r = -b / (2\*a)  
   print(f"One Real Root: {r}")  
else:  
   print("No real roots (complex roots)")  # If discriminant < 0**

**OBJECTIVE**

To perform basic math operations like add, subtract, multiply, and divide. It checks for zero before dividing to avoid errors. It also solves a quadratic expression for a given x. The results are shown clearly for easy understanding.

**OUTPUT:**



## Program 2: Linear Equation Solver

**CODE:-**

**# Given coefficients  
a = 2  
b = 4**

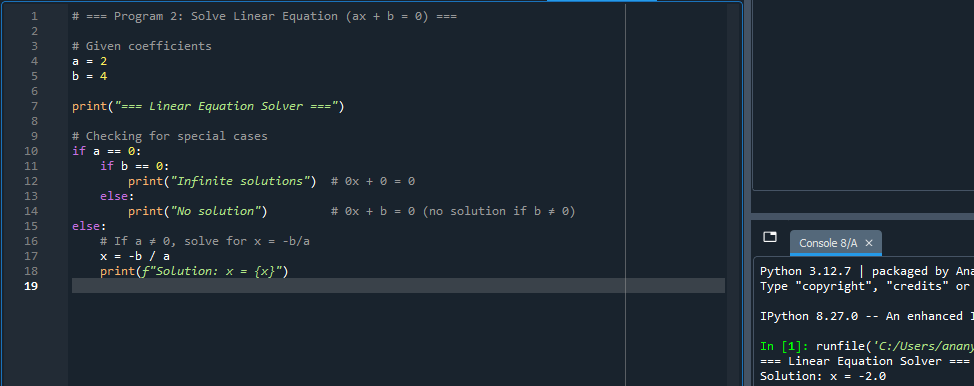
**print("=== Linear Equation Solver ===")**

**# Checking for special cases  
if a == 0:  
   if b == 0:  
       print("Infinite solutions")  # 0x + 0 = 0  
   else:  
       print("No solution")         # 0x + b = 0 (no solution if b ≠ 0)  
else:  
   # If a ≠ 0, solve for x = -b/a  
   x = -b / a  
   print(f"Solution: x = {x}")**

**OBJECTIVE**

To solve the linear equation *y = mx + c* by taking inputs for *m*, *x*, and *c*, then displaying the equation and result clearly.

**OUTPUT:**



## Program 3: Graph Plotting

**CODE:-**

**import numpy as np           # For numerical array and linspace  
import matplotlib.pyplot as plt  # For plotting**

**# Create an array of 100 points from -10 to 10  
x = np.linspace(-10, 10, 100)**

**# Equation: y = x^2  
y = x \*\* 2**

**# Create the plot  
plt.plot(x, y, label="y = x^2", color="blue")**

**# Adding title and labels  
plt.title("Graph of y = x^2")  
plt.xlabel("X-axis")  
plt.ylabel("Y-axis")**

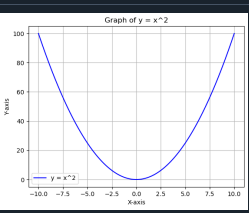
**# Add grid and legend  
plt.grid(True)  
plt.legend()**

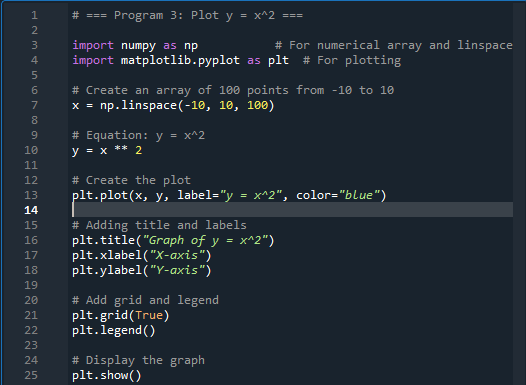
**# Display the graph  
plt.show()**

**OBJECTIVE**

To plot a quadratic equation using NumPy and Matplotlib by calculating y-values for a range of x-values and displaying the graph with labels, a title, grid, and legend.

**OUTPUT:**





## Program 4: Python Function – Factorial

**CODE:-**

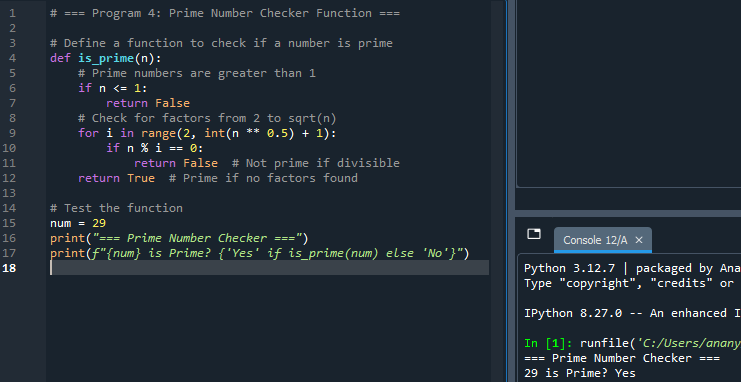
**# Define a function to check if a number is prime  
def is\_prime(n):  
   # Prime numbers are greater than 1  
   if n <= 1:  
       return False  
   # Check for factors from 2 to sqrt(n)  
   for i in range(2, int(n \*\* 0.5) + 1):  
       if n % i == 0:  
           return False  # Not prime if divisible  
   return True  # Prime if no factors found**

**# Test the function  
num = 29  
print("=== Prime Number Checker ===")  
print(f"{num} is Prime? {'Yes' if is\_prime(num) else 'No'}")**

**OBJECTIVE**

To show how recursion works by finding the factorial of a number, where the function keeps calling itself until it reaches 1

**OUTPUT:**



## Program 5: GUI Application using Tkinter

**CODE:-**

**import tkinter as tk  # Import tkinter module**

**# Function to run when the button is clicked  
def submit():  
   name = name\_entry.get()  # Get name from entry box  
   roll = roll\_entry.get()  # Get roll number from entry box  
   print(f"Name: {name}")  
   print(f"Roll No: {roll}")**

**# Create the main window  
window = tk.Tk()  
window.title("Student Information Form")  # Window title**

**# Create labels for input fields  
tk.Label(window, text="Name").grid(row=0, column=0)  
tk.Label(window, text="Roll No").grid(row=1, column=0)**

**# Create entry fields for user input  
name\_entry = tk.Entry(window)  
roll\_entry = tk.Entry(window)**

**# Place entry fields in grid layout  
name\_entry.grid(row=0, column=1)  
roll\_entry.grid(row=1, column=1)**

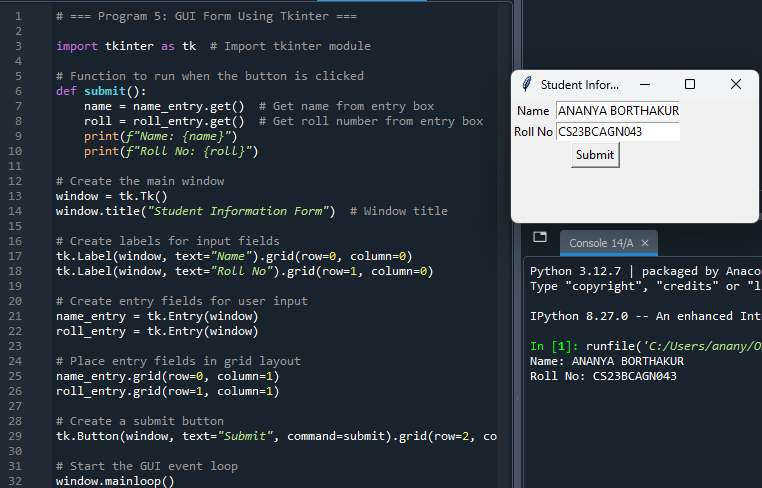
**# Create a submit button  
tk.Button(window, text="Submit", command=submit).grid(row=2, column=0, columnspan=2)**

**# Start the GUI event loop  
window.mainloop()**

**OBJECTIVE**

This program presents a simple graphical user interface application built using Python’s **Tkinter** library. It create a simple GUI form using Tkinter that collects and displays student name and roll number upon submission.

**OUTPUT:**



## Conclusion

This report successfully demonstrates the application of Python programming in diverse areas including mathematical computation, data visualization, recursion, and GUI development. Through implementing arithmetic operations, equation solving, graph plotting, recursive factorial functions, and interactive GUI applications, practical programming skills have been enhanced.

The use of **Spyder IDE** within the **Anaconda distribution** significantly streamlined the coding, testing, and debugging processes, offering an efficient and integrated development environment. These tools facilitated a smooth workflow, enabling clear and organized code development.

Overall, Python proves to be a versatile language that effectively bridges mathematical concepts with interactive design, making it accessible for beginners while powerful enough for experienced programmers engaged in scientific and application development tasks.

***THANK YOU***