Kaziranga University
Department of Computer Science

Programming in Python Report

Name: Sivam Shorahiya
ID: CS23BCAGN033
Faculty: Nawab Wasim Rahman

Python Program Report

This report includes various Python programs showcasing arithmetic operations, linear and quadratic equations, graphical representations, mathematical functions, and a graphical user interface application using tkinter.

INDEX

Task No.	Title	Description
1	**WAP using Python Implementation of Arithmetic and Quadratic Operations**	Performs basic arithmetic operations and solves quadratic equations using standard formula
2	**Implementation of Linear Equation Solver**	Solves linear equations of the form $ax + b = 0$
3	**Graphical Representation Using Mathematical Functions**	Plots graphs like $y = x^2$ and $y = sin(x)$ using matplotlib and numpy
4	**WAP to Implement Mathematical Functions**	Implements **Factorial**, **Area of Circle**, and **Fibonacci Series** as reusable functions
5	**Tkinter Application: Memory Flash Game**	A GUI memory game using tkinter, where users repeat a flashed sequence of colors

Task 1: Arithmetic and Quadratic Operations

Program Logic and Purpose

This Python program has two main functions:

1. Arithmetic Operations:
   It takes two numbers as input and performs:
   - Addition
   - Subtraction
   - Multiplication
   - Division (with a check to avoid dividing by zero)
2. Quadratic Equation Solver:
   It takes three coefficients a, b, and c for a quadratic equation of the form:

   ax2+bx+c=0

   and uses the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   to find the roots of the equation:
   - If discriminant > 0 → Two real and distinct roots
   - If discriminant = 0 → One real root (repeated)
   - If discriminant < 0 → Two complex roots

Code

```
import math

def arithmetic_operations(a, b):
    print("Arithmetic Operations:")
    print(f"Addition: {a} + {b} = {a + b}")
    print(f"Subtraction: {a} - {b} = {a - b}")
    print(f"Multiplication: {a} * {b} = {a * b}")
    if b != 0:
        print(f"Division: {a} / {b} = {a / b}")
    else:
        print("Division: Cannot divide by zero")

def solve_quadratic(a, b, c):
    print("\nQuadratic Equation Solver:")
    discriminant = b**2 - 4*a*c

    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        print(f"Roots are real and different: {root1}, {root2}")
    elif discriminant == 0:
        root = -b / (2*a)
        print(f"Roots are real and same: {root}")
    else:
        realPart = -b / (2*a)
        imagPart = math.sqrt(-discriminant) / (2*a)
```

```
        print(f"Roots are complex: {realPart}+{imagPart}i and {realPart}-{imagPart}i")

# Sample Inputs:
a = float(input("Enter first number for arithmetic: "))
b = float(input("Enter second number for arithmetic: "))
arithmetic_operations(a, b)

print("\n--- Quadratic Equation ---")
a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
solve_quadratic(a, b, c)
```

User Manual

1. Run the program in a Python environment (IDLE, PyCharm, VS Code, etc.).
2. When prompted:
    - Enter two numbers for performing arithmetic operations.
    - The program will print the result of addition, subtraction, multiplication, and division.
3. Next, you will be asked to enter the coefficients a, b, and c of a quadratic equation.
    - The program will calculate and display the roots accordingly:
        i. Real and different
        ii. Real and same
        iii. Complex

Sample Input/Output

- **Input:**

```
Enter first number for arithmetic: 12
Enter second number for arithmetic: 4
```

- **Output:**

```
Arithmetic Operations:
Addition: 12.0 + 4.0 = 16.0
Subtraction: 12.0 - 4.0 = 8.0
Multiplication: 12.0 * 4.0 = 48.0
Division: 12.0 / 4.0 = 3.0
```

Task 2: Linear Equation Solver

Purpose

This program solves a linear equation of the standard form:

$$ax+b=0$$

It calculates the value of x by rearranging the equation:

$$x=-b/a$$

It also handles special cases:

- If $a=0$ and $b=0$: Infinite solutions
- If $a=0$ and $b\neq0$: No solution (contradiction)

Code

```
def solve_linear_equation(a, b):
    print("\nSolving Linear Equation: ax + b = 0")
    if a == 0:
        if b == 0:
            print("Infinite solutions (every x satisfies the equation).")
        else:
            print("No solution (this is a contradiction).")
    else:
        x = -b / a
        print(f"Solution: x = {-b} / {a} = {x}")

a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
solve_linear_equation(a, b)
```

User Manual

1. Run the Script in a Python IDE or terminal.
2. Enter values for the coefficients a and b when prompted.
3. The program will output:
    - A solution for x, or
    - A message saying there are infinite solutions or no solution, based on the inputs.

Sample Input and Output

- **Input:**

```
Enter coefficient a: 5
Enter coefficient b: -10
```

- **Output:**

```
Solving Linear Equation: ax + b = 0
Solution: x = 10.0 / 5.0 = 2.0
```

Task 3: Graphical Representation

Purpose

This program demonstrates the use of matplotlib and NumPy libraries in Python to plot mathematical graphs. Specifically, it plots two functions:

- y = x2 → A parabola
- y = sin(x) → A sine wave

These are displayed side by side in one window using subplot.

Code

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10, 10, 400)
y1 = x**2
y2 = np.sin(x)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(x, y1, color='blue')
plt.title("Graph of y = x²")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(x, y2, color='green')
plt.title("Graph of y = sin(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)

plt.tight_layout()
plt.show()
```

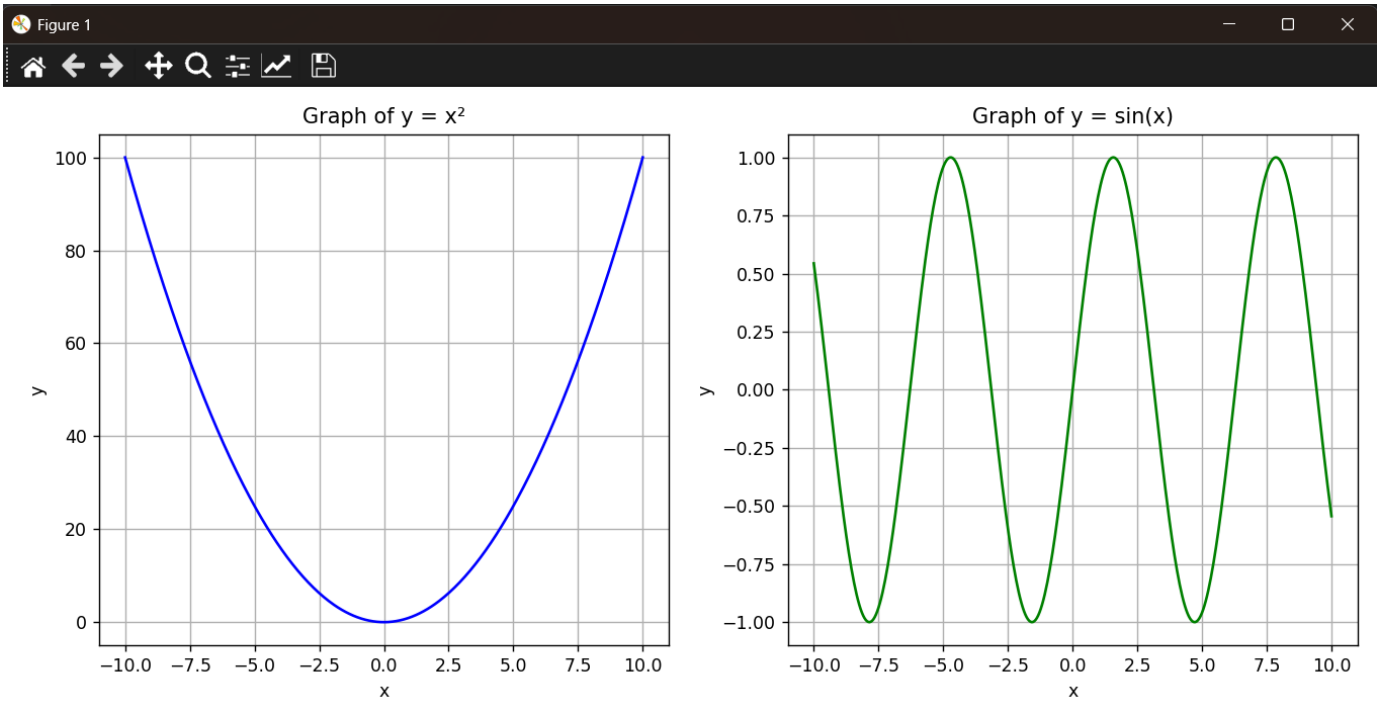
User Manual

1. Install Required Libraries (Only Once):
   Run this in your terminal or command prompt:

```
pip install matplotlib numpy
```

2. Run the Program:
   - Use any Python IDE or script runner (IDLE, VS Code, Jupyter Notebook, etc.).
   - The program will display two side-by-side plots in one window.

Sample Output



1. Left Plot: $y=x^2$
    - A parabola opening upwards.
    - Symmetric around the y-axis.
2. Right Plot: y=sin(x)
    - A wave pattern.
    - Periodic and oscillating between -1 and 1.

Both graphs have:

- Axes labels: x and y
- Titles
- Grid lines for clarity

Task 4: Function Implementations

Purpose

This program implements three commonly used mathematical functions:

1. Factorial — Computes the factorial of a number using recursion.
2. Area of a Circle — Calculates the area given a radius.
3. Fibonacci Series — Generates the first n terms of the Fibonacci sequence.

Code

```
import math

def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

def area_of_circle(radius):
    return math.pi * radius * radius

def fibonacci(n):
    fib_series = []
    a, b = 0, 1
    for _ in range(n):
        fib_series.append(a)
        a, b = b, a + b
    return fib_series

num = int(input("Enter a number for factorial: "))
print(f"Factorial of {num} = {factorial(num)}")

radius = float(input("\nEnter radius of circle: "))
print(f"Area of circle with radius {radius} = {area_of_circle(radius):.2f}")

terms = int(input("\nEnter number of terms for Fibonacci series: "))
print(f"Fibonacci series with {terms} terms: {fibonacci(terms)}")
```

User Manual

1. Run the program in a Python environment.
2. Enter:
   - A non-negative integer for factorial.
   - A positive float for radius.
   - A positive integer for the number of terms in the Fibonacci series.
3. The results of all three functions will be printed.

Sample Input and Output

- **Input:**

```
Enter a number for factorial: 5
```

- **Output:**

```
Enter a number for factorial: 5
```

Task 5: Tkinter Game - Memory Flash

Purpose

The purpose of this program is to create a simple memory game using tkinter, where the user must:

- Observe a short sequence of flashing colors.
- Repeat the sequence by clicking the corresponding color buttons.
- Get instant feedback on whether the input was correct or not.

This game helps improve attention, memory, and concentration.

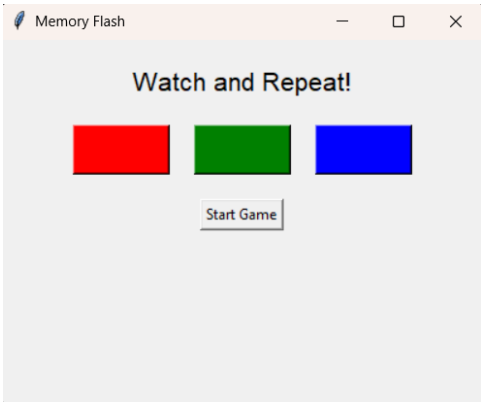
Code

```
import tkinter as tk
import random
import time

colors = ["red", "green", "blue"]
pattern = []

class MemoryGame:
    def __init__(self, root):
        self.root = root
        self.root.title("Memory Flash")
        self.root.geometry("400x300")
        self.user_pattern = []

        self.label = tk.Label(root, text="Watch and Repeat!", font=("Helvetica", 16))
        self.label.pack(pady=20)

        self.box_frame = tk.Frame(root)
        self.box_frame.pack()

        self.buttons = []
        for color in colors:
            btn = tk.Button(self.box_frame, bg=color, width=10, height=2,
                            command=lambda c=color: self.user_click(c), state="disabled")
            btn.pack(side=tk.LEFT, padx=10)
            self.buttons.append(btn)

        self.start_btn = tk.Button(root, text="Start Game", command=self.show_pattern)
        self.start_btn.pack(pady=20)

        self.result = tk.Label(root, text="", font=("Helvetica", 14))
        self.result.pack(pady=10)

    def show_pattern(self):
        self.result.config(text="")
        self.user_pattern = []
        self.start_btn.config(state="disabled")

        global pattern
        pattern = random.choices(colors, k=3)

        for i, color in enumerate(pattern):
            self.root.after(i * 700, lambda c=color: self.flash(c))
        self.root.after(2200, self.enable_buttons)

    def flash(self, color):
        for btn in self.buttons:
            if btn["bg"] == color:
                original = btn["bg"]
                btn.config(bg="white")
                self.root.after(300, lambda b=btn, c=original: b.config(bg=c))

    def enable_buttons(self):
        for btn in self.buttons:
            btn.config(state="normal")

    def user_click(self, color):
        self.user_pattern.append(color)
        if len(self.user_pattern) == 3:
            self.check_result()
```

```
    def check_result(self):
        for btn in self.buttons:
            btn.config(state="disabled")

        if self.user_pattern == pattern:
            self.result.config(text="☑ Correct! Great memory!", fg="green")
        else:
            self.result.config(text=f"❌ Oops! It was {pattern}", fg="red")

        self.start_btn.config(state="normal")

# Run game
root = tk.Tk()
game = MemoryGame(root)
root.mainloop()
```

User Manual

1. Run the script in any Python environment with tkinter support.
2. Click the “Start Game” button to begin.
3. Watch the flashing sequence of three colors.
4. After the flashing stops, click the color buttons in the same order as shown.
5. Feedback will be shown:
    - ✅ Correct if matched.
    - ❌ Oops! with the correct answer if wrong.
6. Click Start Game again to try a new pattern.



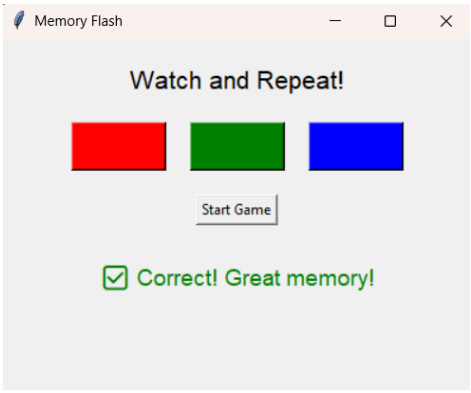
Game Output Examples

✅ **Correct Input:**

Pattern shown: ['red', 'green', 'blue']

User clicks: ['red', 'green', 'blue']

Output: ✅ Correct! Great memory!



❌ Incorrect Input:

Pattern shown: ['blue', 'green', 'red']

User clicks: ['blue', 'red', 'green']

Output: ❌ Oops! It was ['blue', 'green', 'red']