Lab Exercises

1. Write Python script to print prime number form m to n. where m<n
2. Write Python script to create “Book” class with properties “id”, “author” and “price”. Create 4 Book objects and print details of books on console
3. Write Python script to list files and their sizes from a directory
4. Write Python script for performing simple mathematical calculations using GUI.
5. Write python script to generate Login Screen (GUI) and perform authentication using “client” and “server” as username and password respectively

# Prime Number Printer (Python)

## Aim:

The aim of this experiment is to develop a Python script that efficiently prints prime numbers within a specified range [m, n), where m is less than n. The script should utilize a loop and a primality check function to identify and print the prime numbers in the given range.

## Algorithm:

**Step 1:** Start the process

**Step 2:** Prompt the user for two integers, `m` and `n`, representing the range within which prime numbers are generated.

**Step 3:** Define a function `is\_prime` Check if a number is prime by iterating from 2 to the square root of the number and checking for divisibility.

**Step 4**: Print prime numbers within a given range by iterating through the range and using the `is\_prime` function to determine primality.

**Step 5:** Display the prime numbers within the specified range or an error message if the inputs are invalid.

**Step 6:** Stop the process

## Program:

def is\_prime(num): if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1): if num % i == 0:

return False return True

def print\_prime(m, n):

print("Prime numbers between", m, "and", n, "are:") for num in range(m, n + 1):

if is\_prime(num):

print(num)

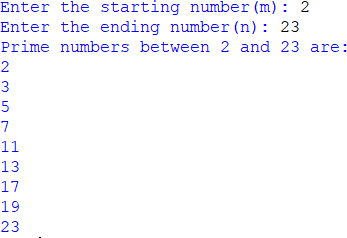
m = int(input("Enter the starting number(m): ")) n = int(input("Enter the ending number(n): "))

if m >= n:

print("Error: Invalid range, m should be less than n.") else:

print\_prime(m, n)

## Output:



**Invalid input**

## Book Class (Python)

**Aim:**

This Python script aims to define a "Book" class with properties such as "id," "author," and

"price." The script then instantiates four Book objects and prints their details, demonstrating the creation and utilization of a class in Python.

## Algorithm:

**Step 1:** Start the process

**Step 2:** Create a class named Book. Include an init method to initialize the properties of the class (book\_id, author, and price).

**Step 3:** Create four instances of the Book class with different details. **Step 4:** Display the details of each Book object using the print function. **Step 5:** Run the script to see the printed details of the four Book objects **Step 6:** Stop the process

## Program:

class Book:

def init (self, book\_id, author, price): self.id = book\_id

self.author = author self.price = price

# Create an empty list to store Book objects books = []

# Get input for creating Book objects using a for loop for i in range(4):

print("Enter details for Book", i + 1) book\_id = int(input("Enter ID: "))

author = input("Enter author: ") price = float(input("Enter price: "))

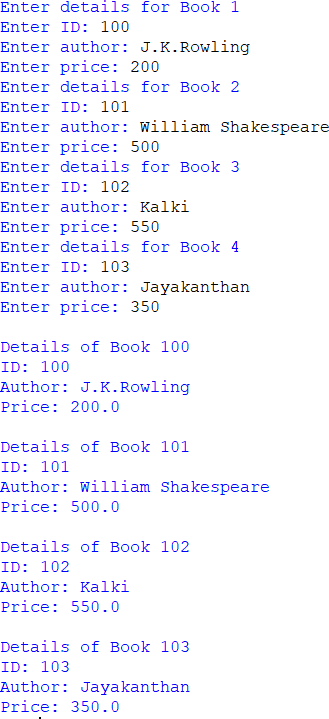
books.append(Book(book\_id, author, price))

# Print details of each book using a for loop for book in books:

print("\nDetails of Book", book.id) print("ID:", book.id)

print("Author:", book.author) print("Price:", book.price)

## Output:



* 1. **Python Script to List Files and their Sizes**

## Aim:

The aim of this script is to list the files and their sizes within a specified directory.

## Algorithm:

**Step 1:** Start the process

**Step 2**: Import the necessary module subprocess.

**Step 3:** Define the command to run the command variable holds the Ubuntu terminal command to be executed, in this case, "ls -l".

**Step 4:** Try-Except block for error handling

**Step 5:** Run the command

**Step 6:** Stop the process

**Program:**

import os

def list\_files\_and\_size(directory): if not os.path.isdir(directory):

print(f"Error:{directory}is not a valid directory:") return

file = os.listdir(directory)

print(f"listing files and size in directory:{directory}") for file\_name in file:

file\_path = os.path.join(directory,file\_name) if os.path.isfile(file\_path):

size = os.path.getsize(file\_path) print(f"{file\_name}:{size}byte")

directory\_path = r"C:\Users\Admin\Desktop\SSS\unit 1" list\_files\_and\_size(directory\_path)

## A screenshot of a computer program Description automatically generatedOutput:

* 1. **(Python) Script for GUI calculator**

## Aim:

The aim of this script for performing simple mathematical calculations using GUI.

## Algorithm:

**Step 1:** Start the process

**Step 2:**Create a Tkinter window (root) titled "Simple Calculator".

**Step 3:** Add two entry widgets to the window to allow users to input numbers.

**Step 4:**Define a list of arithmetic operations (+, -, \*, /).

**Step 5:**For each operation, create a button labeled with the corresponding symbol (+, -, \*,

/). Associate each button with the operate function using lambda functions, passing the respective operation as an argument.

**Step 6:** Inside the operate function, retrieve the values entered in the entry widgets, convert them to floats, and perform the selected operation. Handle division by zero gracefully. Update the result label with the calculated result or an error message.

**Step 7:** Start the main event loop using root.mainloop() to display the GUI and handle user interactions.

**Step 6:** Stop the process

## Program:

import **tkinter** as **tk**

from **tkinter** import **messagebox**

def **calculate**(): try:

num1 = **float**(num1\_entry.**get**()) num2 = **float**(num2\_entry.**get**()) operation = operation\_var.**get**()

if operation == '+': result = num1 + num2

elif operation == '-':

result = num1 - num2

elif operation == '\*': result = num1 \* num2

elif operation == '/': if num2 == 0:

raise **ZeroDivisionError**

result = num1 / num2

result\_label.config(text="Result: {:.2f}".**format**(result)) except **ValueError**:

**messagebox**.**showerror**("Error", "Invalid input! Please enter valid numbers.")

except **ZeroDivisionError**:

**messagebox**.**showerror**("Error", "Cannot divide by zero.")

*# Create main window*

root = **tk**.**Tk**() root.title("Simple Calculator")

*# Number 1 label and entry*

num1\_label = **tk**.**Label**(root, text="Number 1:") num1\_label.grid(row=0, column=0, padx=5, pady=5, sticky=**tk**.E)

num1\_entry = **tk**.**Entry**(root) num1\_entry.grid(row=0, column=1, padx=5, pady=5)

*# Number 2 label and entry*

num2\_label = **tk**.**Label**(root, text="Number 2:") num2\_label.grid(row=1, column=0, padx=5, pady=5, sticky=**tk**.E)

num2\_entry = **tk**.**Entry**(root) num2\_entry.grid(row=1, column=1, padx=5, pady=5)

*# Operation label and dropdown*

operation\_label = **tk**.**Label**(root, text="Operation:") operation\_label.grid(row=2, column=0, padx=5, pady=5, sticky=**tk**.E)

operation\_var = **tk**.**StringVar**(root) operation\_var.**set**('+') *# Default operation is addition*

operation\_dropdown = **tk**.**OptionMenu**(root, operation\_var, '+', '-', '\*', '/') operation\_dropdown.grid(row=2, column=1, padx=5, pady=5)

*# Calculate button*

calculate\_button = **tk**.**Button**(root, text="Calculate", command=**calculate**) calculate\_button.grid(row=3, column=0, columnspan=2, pady=10)

-

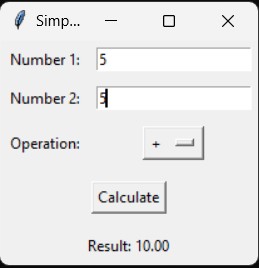
*# Result label*

result\_label = **tk**.**Label**(root, text="Result: ") result\_label.grid(row=4, column=0, columnspan=2, pady=5)

*# Run the main event loop*

root.**mainloop**()

## Output:



* 1. **Python Script for GUI Login Page**

## Aim:

To write a Python script to generate a Login Screen (GUI) and perform authentication using “client” and “server” as username and password respectively

## Algorithm:

**Step 1:** Start the process

**Step 2:** Create a tkinter window with a login form.

**Step 3:** When the user clicks the login button, check if the entered username is "client" and the password is "server".

**Step 4:** If the username and password match, display the “Welcome, client!” message and close the login window.

**Step 5 :** If the username or password is incorrect, display an error message.

**Step 6:** Allow the user to attempt login again.

**Step 7:** Stop the process

**Program:**

import tkinter as tk

from tkinter import messagebox

def authenticate():

username = username\_entry.get()

password = password\_entry.get()

if username == "client" and password == "server":

messagebox.showinfo("Login Successful", "Welcome, client!") # Here you can perform any action after successful login

else:

messagebox.showerror("Login Failed", "Invalid username or password")

# Create main window root = tk.Tk()

root.title("Login")

# Username label and entry

username\_label = tk.Label(root, text="Username:")

username\_label.grid(row=0, column=0, padx=5, pady=5, sticky=tk.E)

username\_entry = tk.Entry(root)

username\_entry.grid(row=0, column=1, padx=5, pady=5)

# Password label and entry

password\_label = tk.Label(root, text="Password:")

password\_label.grid(row=1, column=0, padx=5, pady=5, sticky=tk.E)

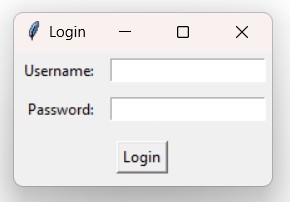
password\_entry = tk.Entry(root, show="\*") password\_entry.grid(row=1, column=1, padx=5, pady=5)

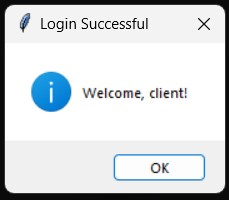
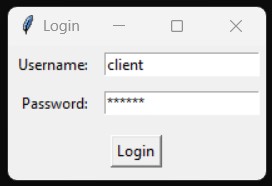
# Login button

login\_button = tk.Button(root, text="Login", command=authenticate) login\_button.grid(row=2, column=0, columnspan=2, pady=10)

# Run the main event loop root.mainloop()

**output:**





## INVALID :

