## ST. JOSEPH’S COLLEGE PILATHARA

**(Affiliated to Kannur University)**

**Pilathara (P.O.), Kannur (Dt.), Kerala – 670504** [**www.stjosephscollege.ac.in**](http://www.stjosephscollege.ac.in/)



**Department of Computer Application**

# BCA Practical Record

**Reg. No.** :

**Name** :

**Semester** : SIX

**Subject** : 6B22BCA Lab VI: Python Programming

## ST. JOSEPH’S COLLEGE PILATHARA



**Department of Computer Application**

**BCA Practical Record**

**CERTIFICATE**

This is to certify that this is a bonafide record of practical work done by Mr./Ms. , Reg. No. of 6th Semester BCA in “Python Programming” as prescribed by Kannur University during the period to

**Examiner 1 Lecturer in Charge**

**Examiner 2 Head of the Department**

**Place :**

**Date :**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | [Largest Number](#_bookmark0) | 1 |
|  | [Perfect Numbers](#_bookmark1) | 2 |
|  | [Binary Search](#_bookmark2) | 4 |
|  | [Bisection Method](#_bookmark3) | 6 |
|  | [Fibonacci Series](#_bookmark4) | 8 |
|  | [LCM and GCD](#_bookmark5) | 10 |
|  | [Merge Sort](#_bookmark6) | 11 |
|  | [File Handling](#_bookmark7) | 13 |
|  | [Prime Numbers](#_bookmark8) | 15 |
|  | [Database Management](#_bookmark9) | 17 |
|  | [GUI Programming](#_bookmark10) | 21 |
|  | [Quadratic Function Graph](#_bookmark11) | 23 |

## Program 1

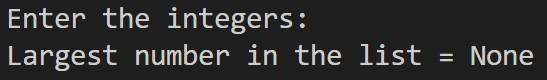
**Write a program to find the largest from a list of numbers.**

nums = map(int, input("Enter the integers: ").split()) ans = max(nums, default=None)

print("Largest number in the list =", ans)

# OUTPUT

Sample1



Sample2



## Program 2

**Write a program to generate first n perfect numbers.**

def is\_perfect\_number(num):

if num < 2:

return False

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

def generate\_perfect\_numbers(limit):

num, count = 1, 0

while count < limit:

num += 1

if is\_perfect\_number(num):

yield num

count += 1

n = int(input("Enter the limit: "))

if n < 1:

print("INVALID limit...")

else:

print(f"Generating the First {n} Perfect Numbers...")

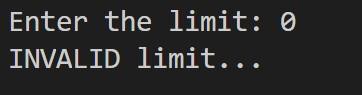
for c, x in enumerate(generate\_perfect\_numbers(n), start=1):

print(f"{c}. {x}")

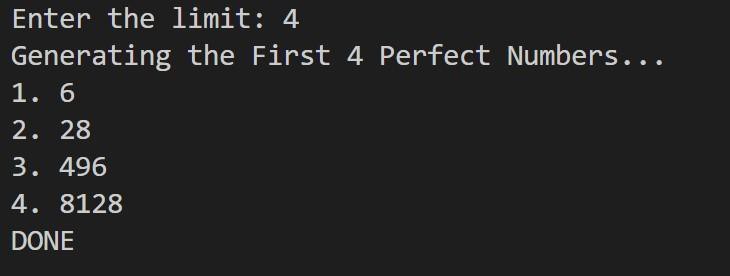
print("DONE")

# OUTPUT

Sample1



Sample2



## Program 3

**Write a program to perform the binary search.**

def binary\_search(arr, key):

l, u = 0, len(arr) - 1

while l <= u:

mid = (l + u) // 2

if arr[mid] == key:

return mid

elif arr[mid] > key:

u = mid - 1

else:

l = mid + 1

return -1

arr = input("Enter the sorted array elements: ").split()

key = input("Enter the element to search: ")

print(f"...Searching for '{key}' in {arr}")

idx = binary\_search(arr, key)

if idx == -1:

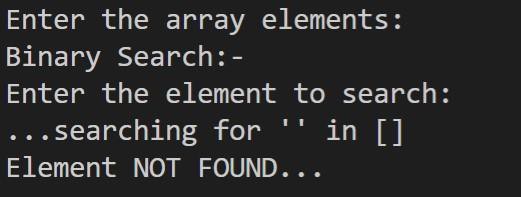
print("Element NOT FOUND...")

else:

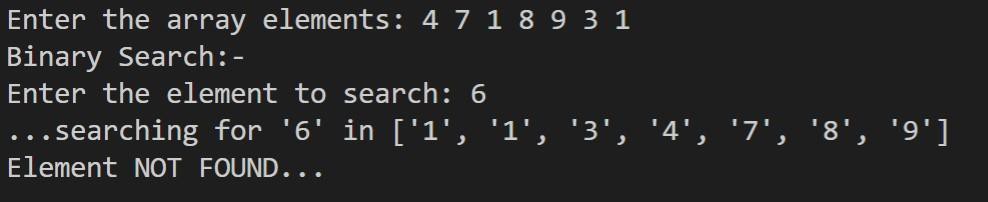
print(f"{key} FOUND at position {idx + 1}")

# OUTPUT

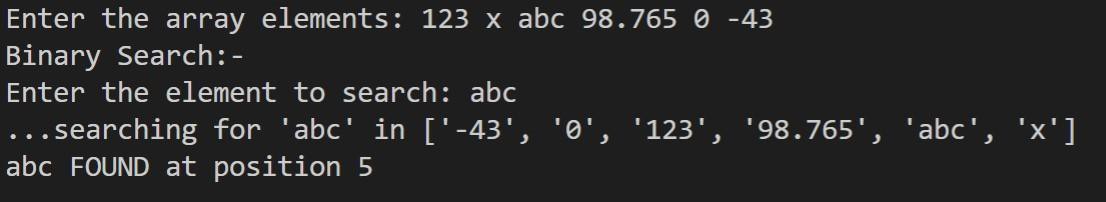
Sample1



Sample2



Sample3



## Program 4

**Write a program to find the square root of a number using bisection search method.**

def bisection\_method(num, tol=0.00001):

x, y = 0, num

while True:

sqrt = (x + y) / 2.0

if abs(sqrt\*\*2 - num) <= tol:

return sqrt

elif sqrt\*\*2 < num:

x = sqrt

else:

y = sqrt

n = float(input("Enter the number: "))

if n < 0:

print("Complex roots...")

else:

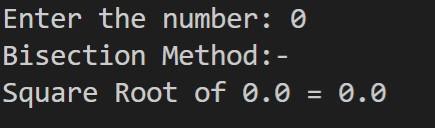
print("Bisection Method:-")

ans = bisection\_method(n)

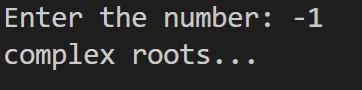
print(f"Square Root of {n} = {ans:.5f}")

# OUTPUT

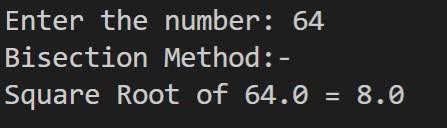
Sample1



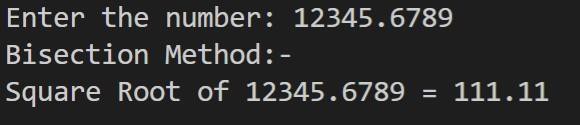
Sample2



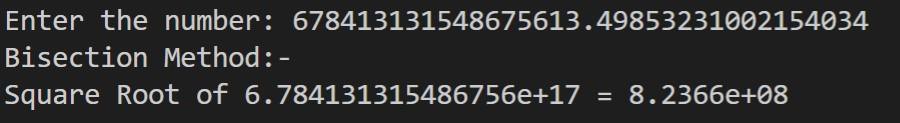
Sample3



Sample4



Sample5



## Program 5

**Write a program to generate Fibonacci series using recursion.**

def fibonacci(n):

if n <= 1:

return n

return fibonacci(n - 1) + fibonacci(n - 2)

def generate\_fibonacci\_series(limit):

return (fibonacci(i) for i in range(limit))

n = int(input("Enter the limit: "))

if n < 1:

print("INVALID limit...")

else:

print(f"Generating the First {n} Fibonacci Numbers using recursion...")

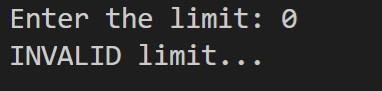
for i, num in enumerate(generate\_fibonacci\_series(n), start=1):

print(f"{i}. {num}")

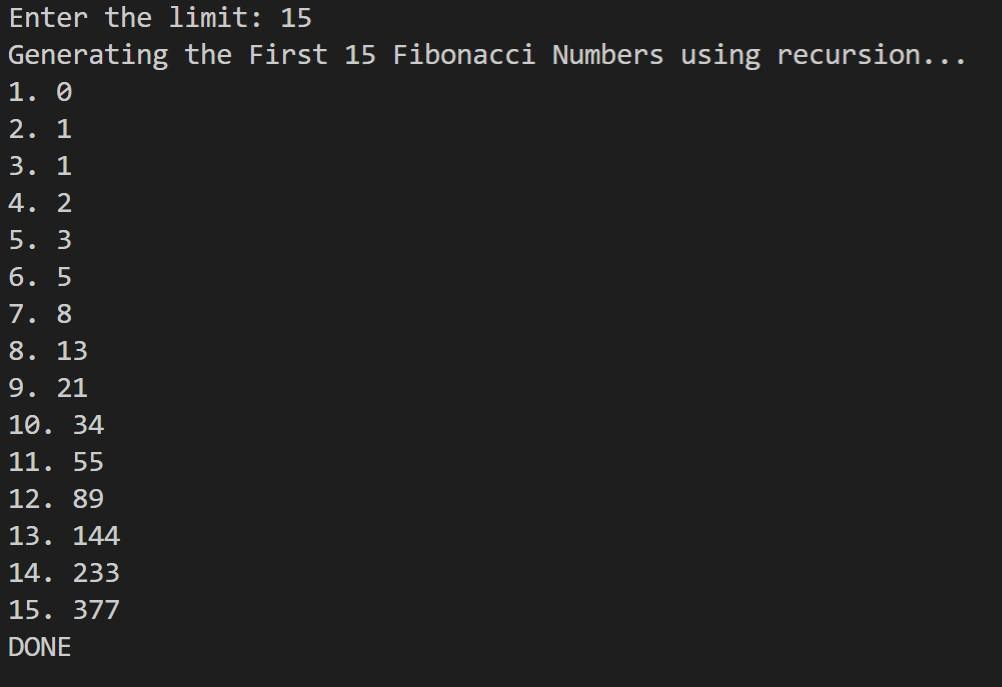
print("DONE")

# OUTPUT

Sample1



Sample2



## Program 6

**Write a program to find the LCM and GCD of 2 numbers.**

import math

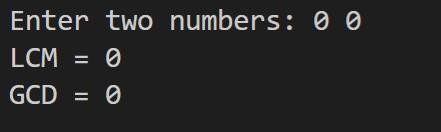
a, b = map(int, input("Enter two numbers: ").split()) GCD = math.gcd(a, b)

LCM = int(a \* b / GCD) if GCD != 0 else 0 print("LCM =", LCM)

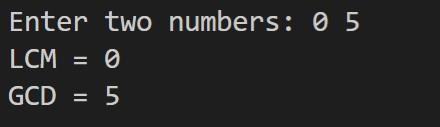
print("GCD =", GCD)

# OUTPUT

Sample1



Sample2



## Program 7

**Write a program to perform merge sort.**

def merge(left, right):

result = []

i = j = 0

while i < len(left) and j < len(right):

if left[i] < right[j]:

result.append(left[i])

i += 1

else:

result.append(right[j])

j += 1

result.extend(left[i:])

result.extend(right[j:])

return result

def merge\_sort(arr):

if len(arr) < 2:

return arr

mid = len(arr) // 2

left = merge\_sort(arr[:mid])

right = merge\_sort(arr[mid:])

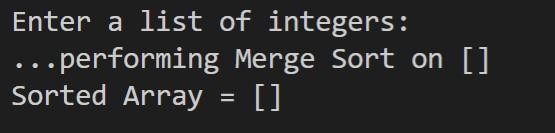
return merge(left, right)

arr = list(map(int, input("Enter a list of integers: ").split()))

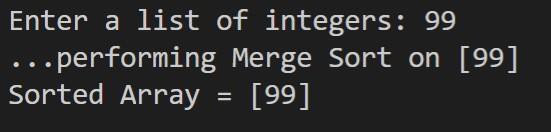
print("Sorted Array =", merge\_sort(arr))

# OUTPUT

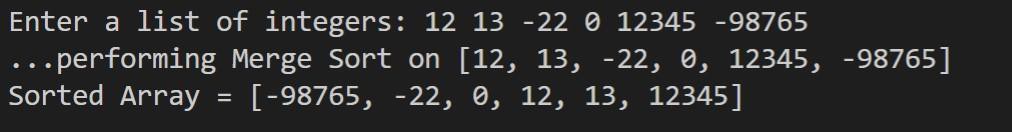
Sample1



Sample2



Sample3



## Program 8

**Write a program which reads the contents of a file and copy the contents to another file after changing all the letter to upper case. Exceptions should be handled.**

import sys

error = None try:

filename = input("Enter the filename: ") with open(filename, mode="rt") as infile:

text = infile.read()

with open("output\_file.txt", mode="wt") as outfile: uppercase\_text = text.upper() print("...conversion DONE") outfile.write(uppercase\_text) print("<output\_file.txt> created SUCCESSFULLY")

except EOFError as e: error = e

sys.stderr.write("! INVALID filename...") except OSError as e:

error = e

sys.stderr.write("! CANNOT open file...") except BaseException:

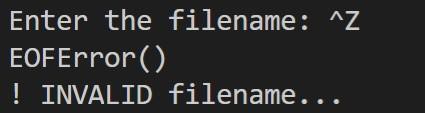
print(sys.exc\_info()) finally:

if error:

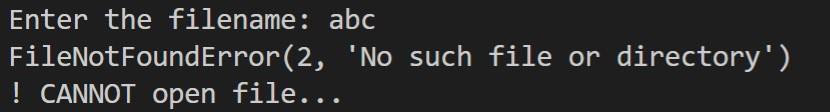
print(repr(error))

# OUTPUT

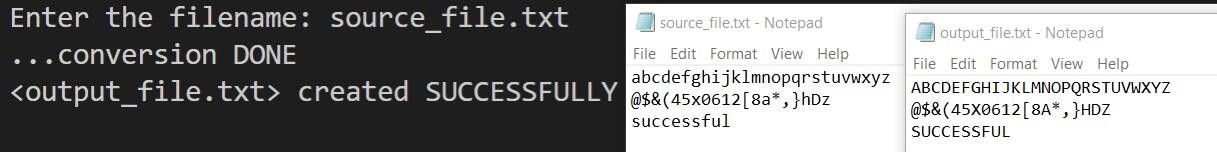
Sample1



Sample2



Sample3



## Program 9

**Write a program to find the prime numbers in a list of numbers.**

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

return False i = 2

while i\*i <= n:

if n % i == 0: return False

i += 1

return True

nums = map(int, input("Enter a list of numbers: ").split()) primes\_found = False

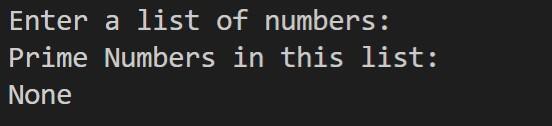
print("Prime Numbers in this list:") for num in filter(is\_prime, nums):

print(num) primes\_found = True

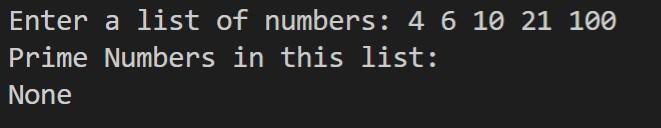
if not primes\_found: print(None)

# OUTPUT

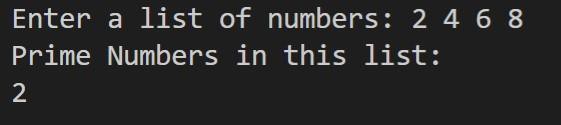
Sample1



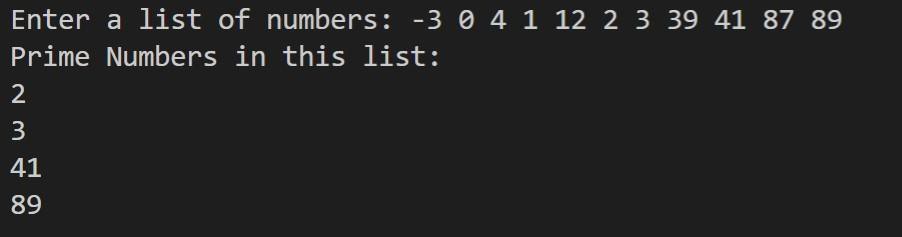
Sample2



Sample3



Sample4



## Program 10

**Write a python program to perform the following**

1. **Create table students with fields name,sex,rollno,marks**
2. **Insert some rows into the table**
3. **Update the marks of all students by adding 2 marks**
4. **Delete a student with a given rollno**
5. **Display the details of a student with a given rollno**

import pymysql

connection = pymysql.connect(

host="localhost",

user="student",

password="123",

autocommit=True

)

cursor = connection.cursor()

try:

cursor.execute("CREATE DATABASE IF NOT EXISTS bca\_practical\_pymysql")

print("Database created/exists.")

except Exception as e:

print("Error creating database:", repr(e))

cursor.execute("USE bca\_practical\_pymysql")

try:

cursor.execute('''

CREATE TABLE IF NOT EXISTS students (

name VARCHAR(255) NOT NULL,

sex ENUM('M', 'F', 'U') NOT NULL,

rollno INT NOT NULL PRIMARY KEY,

marks DECIMAL(5,2) NOT NULL CHECK(marks >= 0)

)

''')

print("Table 'students' created/exists.")

except Exception as e:

print("Error creating table:", repr(e))

cursor.execute("DELETE FROM students")

dummy\_data = [

("Alice", "F", 101, 85.00),

("Bob", "M", 102, 78.00),

("Charlie", "M", 103, 90.00)

]

try:

for i, record in enumerate(dummy\_data, start=1):

sql = "INSERT INTO students VALUES (%s, %s, %s, %s)"

cursor.execute(sql, record)

print(f"Record {i} inserted successfully.")

except Exception as e:

print("Error during insertion:", repr(e))

try:

cursor.execute("UPDATE students SET marks = marks + 2")

if cursor.rowcount == 0:

raise RuntimeError("No records to update.")

print("Updated marks for all students.")

except Exception as e:

print("Error during update:", repr(e))

rollno\_to\_delete = 102

try:

cursor.execute("DELETE FROM students WHERE rollno = %s", (rollno\_to\_delete,))

if cursor.rowcount == 0:

raise KeyError(f"No student with rollno={rollno\_to\_delete}")

print("Student deleted successfully.")

except Exception as e:

print("Error during deletion:", repr(e))

rollno\_to\_display = 103

try:

cursor.execute('''

SELECT CAST(name AS CHAR) AS Name,

CAST(sex AS CHAR) AS Sex,

CAST(rollno AS CHAR) AS Rollno,

CAST(marks AS CHAR) AS Marks

FROM students

WHERE rollno = %s

''', (rollno\_to\_display,))

result = cursor.fetchone()

if result is None:

raise KeyError(f"No student with rollno={rollno\_to\_display}")

print("Student details:")

for col, val in zip(("Name", "Sex", "Rollno", "Marks"), result):

print(f"{col:<8}: {val}")

except Exception as e:

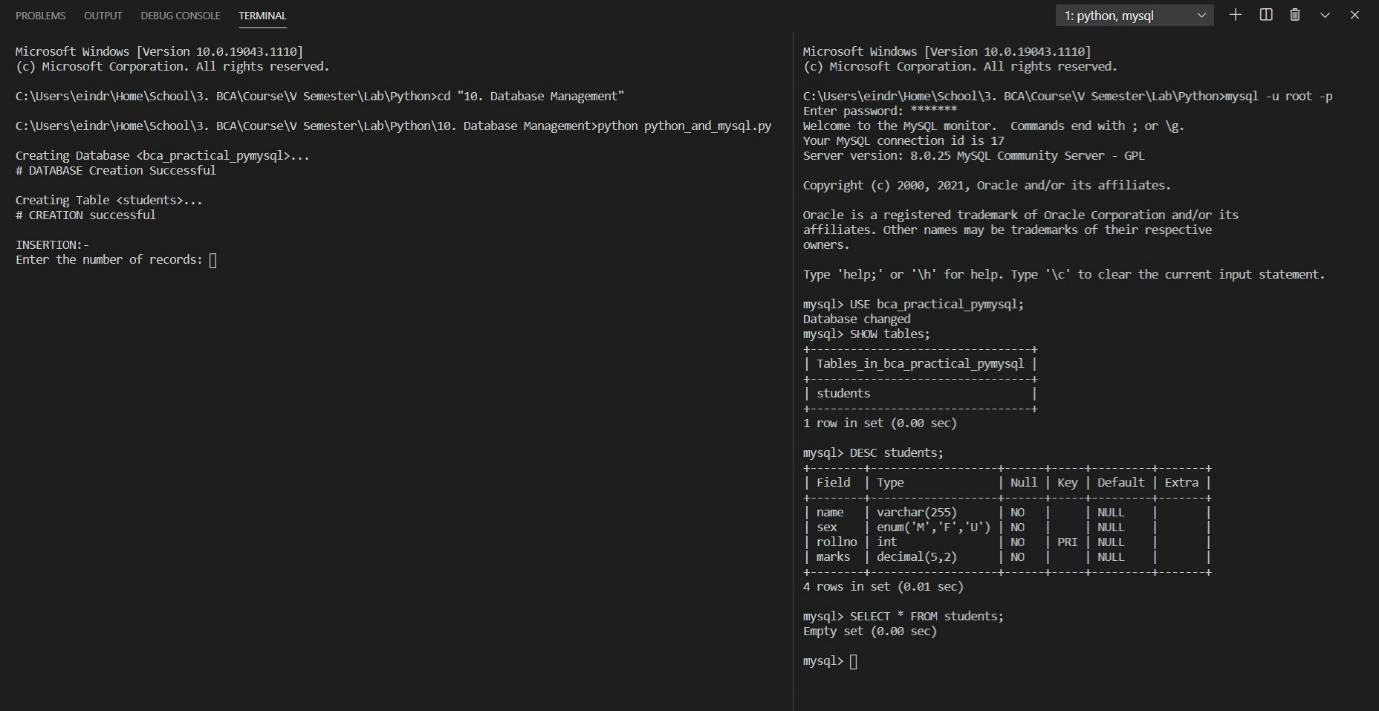
print("Error during display:", repr(e))

connection.close()

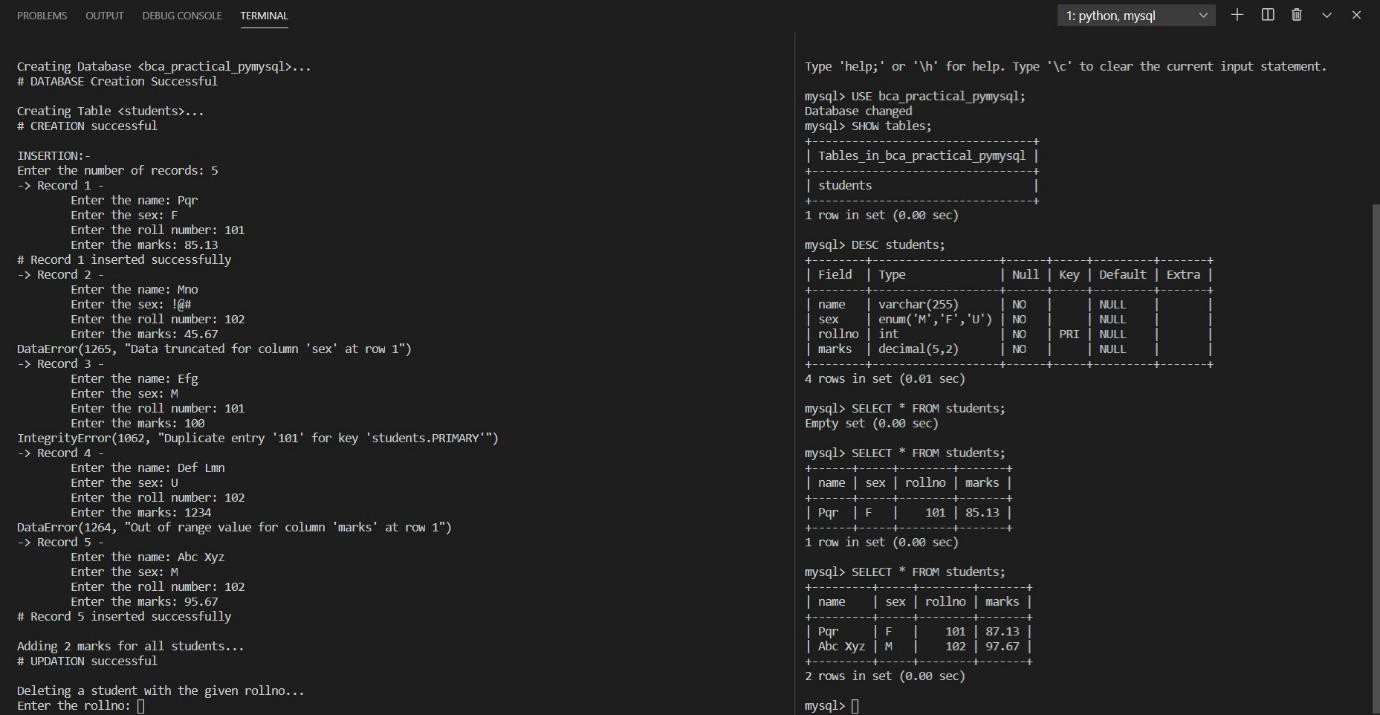
print("Program Execution SUCCESSFUL")

# OUTPUT

Segment 1



Segment 2



## Program 11

**Create a simple Login window using Tkinter.**

import tkinter as tk

import tkinter.font as tk\_font

root = tk.Tk()

root.title("Login Window using Tkinter") root.geometry('600x300')

text\_font = tk\_font.Font(family="Courier New", size=12, weight="bold")

input\_font = tk\_font.Font(family="Lucida Console", size=10)

username\_label = tk.Label(root, text="Username", font=text\_font) password\_label = tk.Label(root, text="Password", font=text\_font) username\_entry = tk.Entry(root, width=50, font=input\_font) password\_entry = tk.Entry(root, show="\*", width=50, font=input\_font) login\_button = tk.Button(root, text="LOGIN", font=text\_font)

username\_label.grid(row=0, column=0, padx=20, pady=10, ipadx=5, ipady=20)

password\_label.grid(row=1, column=0, padx=20, pady=10, ipadx=5, ipady=20)

username\_entry.grid(row=0, column=1, padx=20, pady=10, ipadx=5, ipady=10)

password\_entry.grid(row=1, column=1, padx=20, pady=10, ipadx=5, ipady=10)

login\_button.grid(row=2, column=1, padx=20, pady=20, ipadx=100, ipady=10)

def login():

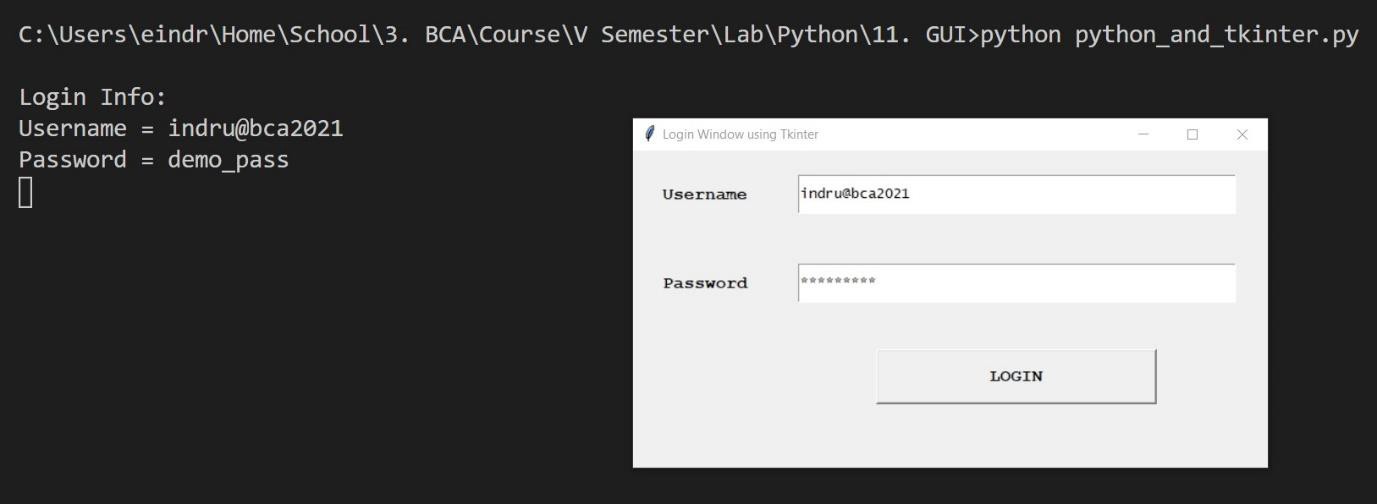
print("\nLogin Info:")

print("Username =", username\_entry.get()) print("Password =", password\_entry.get())

login\_button.config(command=login)

root.mainloop()

# OUTPUT



## Program 12

**Create a plot for the mathematical function x2. The title of the plot and the axes should be labelled.**

import matplotlib.pyplot as plt import numpy as np

l = int(input("Enter the inclusive lower bound: ")) u = int(input("Enter the inclusive upper bound: "))

print("Plotting...")

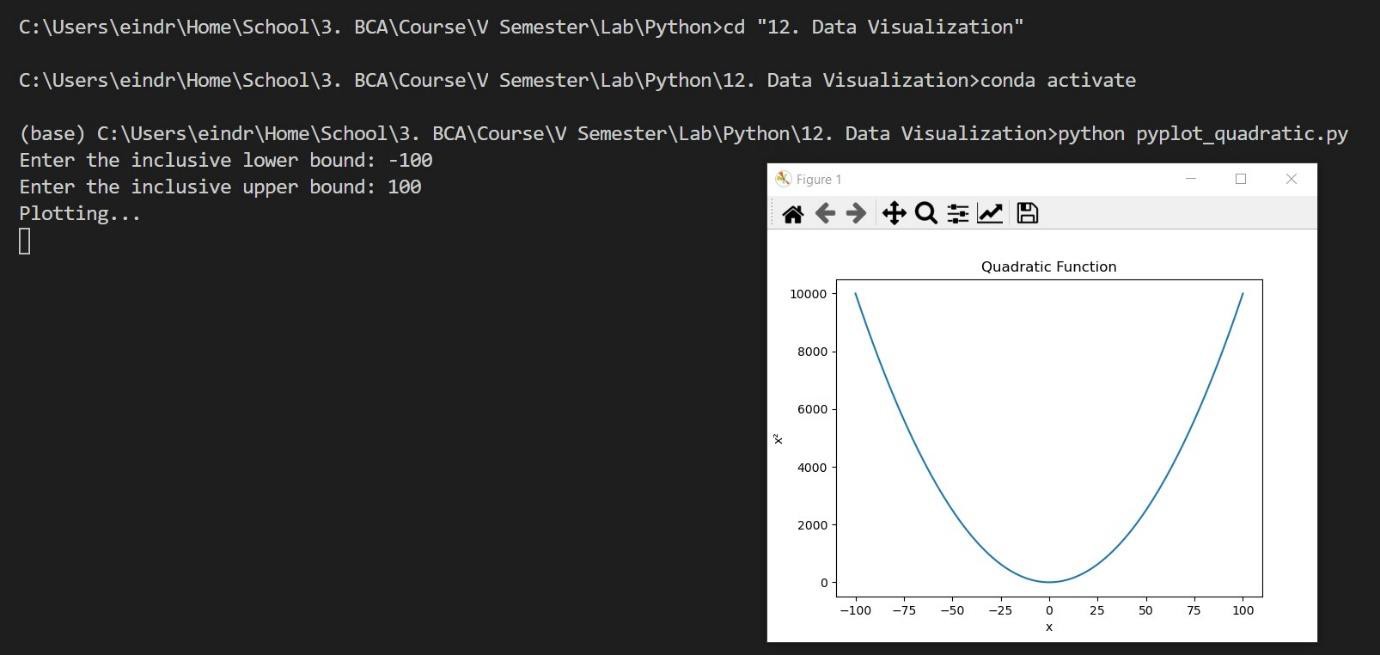
x = np.arange(l, u, 0.1) y = x\*\*2

plt.title("Quadratic Function") plt.xlabel("x")

plt.ylabel("x²")

plt.plot(x, y) plt.show()

# OUTPUT



\*-\*-\*-\*-\*