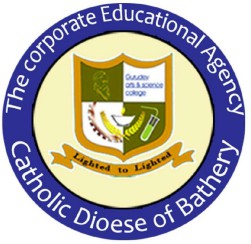
**GURUDEV ARTS AND SCIENCE COLLEGE**

MATHIL, PAYYANUR, KANNUR DIST.

(Affiliated to Kannur University)



**BACHELOR OF COMPUTER APPLICATION**

**PYTHON PROGRAMMING**

**PRACTICAL RECORD**

:

:

**Name**

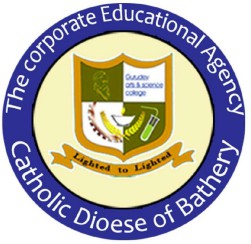
**Register No**



**GURUDEV ARTS AND SCIENCE COLLEGE**

MATHIL, PAYYANUR, KANNUR DIST.

(Affiliated to Kannur University)



**DEPARTMENT OF COMPUTER SCIENCE**

Certified that this is the bonafide record of practical work done by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of III BCA at Gurudev Arts and College, Mathil for the year 2024 to 2025.

Lecture Charge:



Examiner: 1.

2.

Head Of Dept. Computer Science

Submitted for University Examinations 2025

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Program** | **Page No.** |
| 1 | Write a program to find the largest number from a list of numbers. | 1 |
| 2 | Write a program to generate the first n perfect numbers. | 2 |
| 3 | Write a program to perform a binary search. | 3 |
| 4 | Write a program to find the square root of a number using the bisection search method. | 4 |
| 5 | Write a program to generate the Fibonacci series using recursion. | 5 |
| 6 | Write a program to find the LCM and GCD of two numbers. | 6 |
| 7 | Write a program to perform merge sort. | 7–8 |
| 8 | Write a program that reads the contents of a file and copies them to another file after converting all letters to uppercase. Exceptions should be handled. | 9 |
| 9 | Write a program to find the prime numbers in a list of numbers. | 10 |
| 10 | Write a Python program to perform the following database operations:   1. Create a 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. | 11–13 |
|  |  |  |
|  |  |  |
| 11 | Create a simple login window using Tkinter. | 14-15 |
| 12 | Create a plot in Python with a title and labeled axes. | 16 |



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

def find\_largest(numbers):

if not numbers:

return None # Return None if the list is empty

largest = numbers[0] # Assume the first number is the largest

for number in numbers:

if number > largest:

largest = number # Update largest if the current number is bigger

return largest

# Example usage:

numbers = [10, 24, 5, 37, 42, 18, 4]

largest\_number = find\_largest(numbers)

print("The largest number is:", largest\_number)

**OUTPUT:**

The largest number is: 42

**Q:Write a program to generate first n perfect numbers**

def is\_perfect\_number(number):

if number < 2:

return False

divisors\_sum = sum(divisor for divisor in range(1, number) if number % divisor == 0)

return divisors\_sum == number

def generate\_perfect\_numbers(n):

perfect\_numbers = []

candidate = 2 # Start checking from 2

while len(perfect\_numbers) < n:

if is\_perfect\_number(candidate):

perfect\_numbers.append(candidate)

candidate += 1

return perfect\_numbers

# Example usage:

n = 4

perfect\_numbers = generate\_perfect\_numbers(n)

print(f"The first {n} perfect numbers are:", perfect\_numbers)

**OUTPUT:**

The first 4 perfect numbers are: [6, 28, 496, 8128]

**Q: Write a program to perform the binary search**

def binary\_search(sorted\_list, target):

left = 0

right = len(sorted\_list) - 1

while left <= right:

mid = (left + right) // 2

mid\_value = sorted\_list[mid]

if mid\_value == target:

return mid # Target found, return its index

elif mid\_value < target:

left = mid + 1 # Continue search in the right half

else:

right = mid - 1 # Continue search in the left half

return -1 # Target not found

# Example usage:

sorted\_list = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

target = 7

result = binary\_search(sorted\_list, target)

if result != -1:

print(f"Target {target} found at index {result}.")

else:

print(f"Target {target} not found in the list.")

**OUTPUT:**

Target 7 found at index 3.

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

def bisection\_sqrt(x, epsilon=1e-10):

if x < 0:

raise ValueError("Cannot compute the square root of a negative number.")

if x == 0 or x == 1:

return x

low = 0

high = max(1, x)

guess = (low + high) / 2.0

while abs(guess\*\*2 - x) > epsilon:

if guess\*\*2 < x:

low = guess

else:

high = guess

guess = (low + high) / 2.0

return guess

# Example usage:

number = 25

result = bisection\_sqrt(number)

print(f"The square root of {number} is approximately {result}.")

**OUTPUT:**

The square root of 25 is approximately 4.9999999999954525.

**Q: Write a program to generate Fibonacci series using recursion**

def fibonacci(n):

if n <= 0:

raise ValueError("The input number must be a positive integer.")

if n == 1:

return 0

if n == 2:

return 1

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

def generate\_fibonacci\_series(length):

if length <= 0:

raise ValueError("The length of the series must be a positive integer.")

series = []

for i in range(1, length + 1):

series.append(fibonacci(i))

return series

# Example usage:

length = 10

series = generate\_fibonacci\_series(length)

print(f"The first {length} numbers in the Fibonacci series are: {series}")

**OUTPUT:**

The first 10 numbers in the Fibonacci series are: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

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

def gcd(a, b):

while b:

a, b = b, a % b

return a

def lcm(a, b):

return a \* b // gcd(a, b)

# Example usage:

num1 = 12

num2 = 18

gcd\_result = gcd(num1, num2)

lcm\_result = lcm(num1, num2)

print(f"The GCD of {num1} and {num2} is {gcd\_result}.")

print(f"The LCM of {num1} and {num2} is {lcm\_result}.")

**OUTPUT:**

The GCD of 12 and 18 is 6.

The LCM of 12 and 18 is 36.

**Q: Write a program to perform merge sort**

def merge\_sort(arr):

if len(arr) <= 1:

return arr

# Find the middle point to divide the array into two halves

mid = len(arr) // 2

# Call merge\_sort for the first half

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

# Call merge\_sort for the second half

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

# Merge the two halves sorted in step 2 and 3

return merge(left\_half, right\_half)

def merge(left, right):

merged = []

left\_index, right\_index = 0, 0

# Traverse both arrays and insert smaller of both elements in merged array

while left\_index < len(left) and right\_index < len(right):

if left[left\_index] < right[right\_index]:

merged.append(left[left\_index])

left\_index += 1

else:

merged.append(right[right\_index])

right\_index += 1

# Collect remaining elements (if any)

merged.extend(left[left\_index:])

merged.extend(right[right\_index:])

return merged

# Example usage:

arr = [38, 27, 43, 3, 9, 82, 10]

sorted\_arr = merge\_sort(arr)

print(f"Sorted array: {sorted\_arr}")

**OUTPUT:**

Sorted array: [3, 9, 10, 27, 38, 43, 82]

**Q: 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.**

try:

    # Open the source file in read mode

    with open("input.txt", "r") as f1:

        # Read the contents of the source file

        content = f1.read()

    # Convert content to uppercase

    upper\_content = content.upper()

    # Open the destination file in append mode and write the uppercase content

    with open("output.txt", "a") as f2:

        f2.write(upper\_content)

    # Open the destination file in read mode and print its contents

    with open("output.txt", "r") as f2:

        for x in f2.read():

            print(x, end="")

except IOError:

    print("Error: can't find file or read data")

**OUTPUT:**

THE THINGS WE COULD HAVE SHARED,BECAUSE IN THAT MOMENT

I KNEW THAT OF ALL THE PEOPLE IN THE WORLD,

THERE WAS ONLY ONE I WANTED TO BE WITH, AND IT WAS HER.

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

numberList = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

ansList = []

for num in numberList:

# Check for numbers less than 2

if num <= 1:

continue

# Assume num is prime until proven otherwise

is\_prime = True

# Check for factors from 2 to sqrt(num)

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

if num % i == 0:

is\_prime = False

break

# If no factors were found, num is prime

if is\_prime:

ansList.append(num)

# Print the result

if ansList:

print("Prime Numbers:")

for ans in ansList:

print(ans)

else:

print("No number in the given list is Prime")

**OUTPUT:**

Prime Numbers:

2

3

5

7

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

**a) Create table students with fields name,sex,rollno,marks.**

**b) Insert some rows into the table**

**c) Update the marks of all students by adding 2 marks.**

**d) Delete a student with a given rollno.**

**e) Display the details of a student with a given rollno.**

import mysql.connector

# Establish a connection to the database

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

password="yourpassword",

database="mydatabase"

)

mycursor = mydb.cursor()

**# a) Create table students with fields name, sex, rollno, marks**

mycursor.execute("""

CREATE TABLE IF NOT EXISTS students (

rollno INT PRIMARY KEY,

name VARCHAR(30),

sex VARCHAR(1),

marks INT

)

""")

**# b) Insert some rows into the table**

sql = "INSERT INTO students (rollno, name, sex, marks) VALUES (%s, %s, %s, %s)"

val = [

(1, 'Peter', 'M', 49),

(2, 'Diya', 'F', 40),

(3, 'Manu', 'M', 39),

(4, 'Anjali', 'F', 45),

(5, 'Manas', 'M', 48)

]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) inserted.")

**# c) Update the marks of all students by adding 2 marks**

sql = "UPDATE students SET marks = marks + 2"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) updated.")

# Display all records after the update

mycursor.execute("SELECT \* FROM students")

myresult = mycursor.fetchall()

print("Students after update:")

for x in myresult:

print(x)

**#d) Delete a student with a given rollno**

rno = int(input('Enter the roll number of student to delete: '))

sql = "DELETE FROM students WHERE rollno = %s"

mycursor.execute(sql, (rno,))

mydb.commit()

print(mycursor.rowcount, "record(s) deleted.")

**# e) Display the details of a student with a given rollno**

rno = int(input('Enter the roll number of student to display the details: '))

sql = "SELECT \* FROM students WHERE rollno = %s"

mycursor.execute(sql, (rno,))

myresult = mycursor.fetchall()

print("Student details:")

if myresult:

for x in myresult:

print(x)

else:

print("No student found with the given roll number.")

# Close the cursor and connection

mycursor.close()

mydb.close()

**OUTPUT:**

5 record(s) inserted.

5 record(s) updated.

Students after update:

(1, 'Peter', 'M', 51)

(2, 'Diya', 'F', 42)

(3, 'Manu', 'M', 41)

(4, 'Anjali', 'F', 47)

(5, 'Manas', 'M', 50)

Enter the roll number of student to delete: 3

1 record(s) deleted.

Enter the roll number of student to display the details: 2

Student details:

(2, 'Diya', 'F', 42)

**Q: Create a simple Login window using Tkinter**

import tkinter as tk

from tkinter import messagebox

# Function to handle the login logic

def login():

    username = entry\_username.get()

    password = entry\_password.get()

    # Simple validation logic (you can replace this with actual authentication)

    if username == "user" and password == "password":

        messagebox.showinfo("Login Successful", "Welcome, " + username + "!")

    else:

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

# Create the main application window

root = tk.Tk()

root.title("Login Window")

# Set window size

root.geometry("300x150")

# Create and place the username label and entry field

label\_username = tk.Label(root, text="Username:")

label\_username.pack(pady=5)

entry\_username = tk.Entry(root)

entry\_username.pack(pady=5)

# Create and place the password label and entry field

label\_password = tk.Label(root, text="Password:")

label\_password.pack(pady=5)

entry\_password = tk.Entry(root, show="\*")

entry\_password.pack(pady=5)

# Create and place the login button

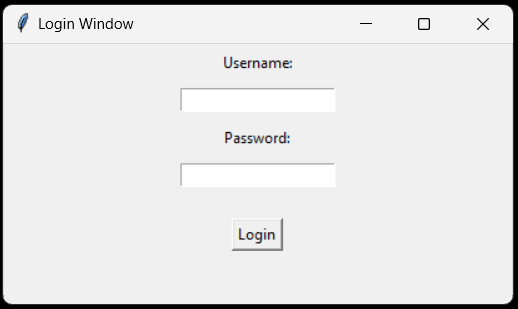
login\_button = tk.Button(root, text="Login", command=login)

login\_button.pack(pady=20)

# Run the application

root.mainloop()

**OUTPUT:**



**Q: Create a plot in Python. The title of the plot and the axes should be labelled.**

import matplotlib.pyplot as plt

# Sample data

x = [0, 1, 2, 3, 4, 5]

y = [0, 1, 4, 9, 16, 25]

# Create a plot

plt.plot(x, y)

# Add title and labels

plt.title("Sample Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

# Show the plot

plt.show()

**OUTPUT:**

