SNS COLLEGE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)
COIMBATORE–641 035.
Approved by AICTE New Delhi & Affiliated to Anna University Chennai
Accredited by NBA & Accredited by NAAC with 'A+' Grade, Recognized by UGC

DEPARTMENT OF BIOMEDICAL ENGINEERING

PRACTICAL LAB RECORD

23ITP204 - PROGRAMMING IN PYTHON

II B.E. BME / IV SEMESTER

Name ……..

Register No. ……..

Year/ Semester ……..

Branch ……..

Academic Year : 2024-2025 (Even Semester)

SNS COLLEGE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

Coimbatore – 35

DEPARTMENT OF BIOMEDICAL ENGINEERING

23ITP204 - PROGRAMMING IN PYTHON

Name: …….. Roll No: ……

Class: …….. Semester: ……

Register No: ……

Certified that this is the bonafide record of work done by the above student of the "**23ITP204 - PROGRAMMING IN PYTHON"** during the year 2024-2025 (Even Semester).

Signature of Lab In-charge **Head of the Department**

Submitted for the practical examination held on ……

Internal Examiner **External Examiner**

SYLLABUS

23ITP204	PROGRAMMING IN PYTHON	L	T	P	C
(Common to All B.E/B.Tech Programme Except CSE, IT and AIML)		0	0	4	2

COURSE OBJECTIVES:

- To train the student to the basic concepts of python programming language.
- To develop correct and efficient Python programs to solve problems spanning multiple disciplines.

LIST OF EXPERIMENTS

1. Program for various base conversion functions.
2. Programs to demonstrate the usage of operators and conditional statements
3. Programs to demonstrate usage of control structures
4. Program using array operation
5. Programs to demonstrate the usage of String functions
6. Program using classes and functions
7. Program to implement recursive function.
8. Program to implement lambda function.
9. Program on file manipulation
10. Programs to demonstrate the usage of lists, sets, dictionaries and tuples.
11. Program to implement function template.
12. Program to implement class template

L:0	T:0	P:60	Total: 60 Periods

COURSE OUTCOMES

At the end of the course students should be able to

CO 1	Write simple programs using built-in data types of Python.
CO 2	Apply the conditional statements and loops for solving problems.
CO 3	Implement arrays, strings and functions in Python
CO 4	Identify the commonly used operations involving lists, sets, dictionaries, tuples and file handling in real time applications.
CO 5	Implement exemplary applications related to templates for solving real time problems.

SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Coimbatore – 35

DEPARTMENT OF BIOMEDICAL ENGINEERING

VISION

To provide world class education with Centre of Excellence in the field of Biomedical Engineering to cater the need of Medical Industries, research and technology development for the benefit of society.

MISSION

- To offer quality education of international acclaim by imbibing critical and creative analysis in designing Biomedical Engineering solutions
- To provide opportunities and conducive environment to the faculty members to enhance their skills and expertise in teaching, research and consultancy activities
- To translate scientific discovery in medical technology for better health care
- To foster the students to understand ethical, social and economic implication of their work for the improvement of society

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Graduate will demonstrate their acquired knowledge in technical competence and professional skills to solve wide range of challenges in Biomedical Engineering and advanced contemporary areas.

PEO 2: Graduate will communicate with multidisciplinary teams and engage in research, contribute to the society.

PEO 3: Graduate will pursuit knowledge in the field of Biomedical Engineering to contribute to the profession and employability.

PEO 4: Graduate will design and develop the products using modern tools for the advancement of Biomedical Engineering and generate the employment through entrepreneurship.

PEO 5: Graduate will apply their professional knowledge in the human, social, ethical and environmental context.

PROGRAMME OUTCOMES (POs)

At the end of the program, graduate will be able to:

PO1	**Engineering Knowledge**	Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	**Problem Analysis**	Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO3	**Design/Development of Solutions**	Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs

		with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO4	**Conduct Investigations of Complex Problems**	Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	**Engineering Tool Usage**	Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
PO6	**The Engineer and The World**	Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO7	**Ethics**	Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
PO8	**Individual and Collaborative Team work**	Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO9	**Communication**	Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences
PO10	**Project Management and Finance**	Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO11	**Life-Long Learning**	Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

PROGRAMME SPECIFIC OBJECTIVES

At the end of this program, graduate will be able to:

PSO 1 : Analyze, design and develop the systems to supplement and/ or assist the physiology of the human body.

PSO 2: Develop the mathematical model to understand the inter-relation among various Physiological systems

INDEX

S. No	Date	Name of the Experiments	Page No	Marks	Staff Signature

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	BASE CONVERSION FUNCTIONS

AIM:

To develop a Python program that implements various base conversion functions using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter the base of the input number.
Step 3: Prompt the user to input the number to be converted.
Step 4: Convert the input number to the other base types.
Step 5: Print the converted values for the different base types.
Step 6: Stop the program.

PROGRAM:

```
input_base = int(input("Enter the input base (e.g., 2, 8, 10, 16): "))
input_number = input("Enter the number to be converted: ")

if input_base == 10:
    num = int(input_number)

    # Convert Decimal to Binary
    binary_result = bin(num)[2:]
    print(f"Decimal to Binary: {binary_result}")

    # Convert Decimal to Octal
    octal_result = oct(num)[2:]
    print(f"Decimal to Octal: {octal_result}")

    # Convert Decimal to Hexadecimal
    hexadecimal_result = hex(num)[2:].upper()
    print(f"Decimal to Hexadecimal: {hexadecimal_result}")

elif input_base == 2:
    # Convert Binary to Decimal
    decimal_result = int(input_number, 2)
    print(f"Binary to Decimal: {decimal_result}")

    # Convert Binary to Octal
    octal_result = oct(decimal_result)[2:]
    print(f"Binary to Octal: {octal_result}")

    # Convert Binary to Hexadecimal
    hexadecimal_result = hex(decimal_result)[2:].upper()
    print(f"Binary to Hexadecimal: {hexadecimal_result}")
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
elif input_base == 8:
    # Convert Octal to Decimal
    decimal_result = int(input_number, 8)
    print(f"Octal to Decimal: {decimal_result}")

    # Convert Octal to Binary
    binary_result = bin(decimal_result)[2:]
    print(f"Octal to Binary: { binary_result}")

    # Convert Octal to Hexadecimal
    hexadecimal_result = hex(decimal_result)[2:].upper()
    print(f"Octal to Hexadecimal: {hexadecimal_result}")

elif input_base == 16:
    # Convert Hexadecimal to Decimal
    decimal_result = int(input_number, 16)
    print(f"Hexadecimal to Decimal: {decimal_result}")

    # Convert Hexadecimal to Binary
    binary_result = bin(decimal_result)[2:]
    print(f"Hexadecimal to Binary: {binary_result}")

    # Convert Hexadecimal to Octal
    octal_result = oct(decimal_result)[2:]
    print(f"Hexadecimal to Octal: {octal_result}")

else:
    print("Invalid base input!")
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program for various base conversion functions was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	OPERATORS AND CONDITIONAL STATEMENTS

AIM:

To develop a Python program to demonstrate the usage of operators and conditional statements using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter two input numbers.
Step 3: Compute the results for various operators and print the result.
Step 4: Compute the results using various conditional statements and print the result.
Step 5: Stop the program.

PROGRAM:

```
#Input
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Arithmetic operators
print("\nArithmetic Operations:")
print("Addition:", num1 + num2)
print("Subtraction:", num1 - num2)
print("Multiplication:", num1 * num2)
print("Division:", num1 / num2)
print("Floor Division:", num1 // num2)
print("Modulus:", num1 % num2)
print("Exponentiation:", num1 ** num2)

# Comparison operators
print("\nComparison Operations:")
print("Equal to:", num1 == num2)
print("Not equal to:", num1 != num2)
print("Greater than:", num1 > num2)
print("Less than:", num1 < num2)
print("Greater than or equal to:", num1 >= num2)
print("Less than or equal to:", num1 <= num2)

# Logical operators
print("\nLogical Operations (using True and False):")

bool1 = True
bool2 = False

print("AND:", bool1 and bool2)
print("OR:", bool1 or bool2)
print("NOT:", not bool1)
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
# Assignment operators
print("\nAssignment Operations:")

x = 5
print("Initial value of x:", x)
x += 2  # x = x + 2
print("x += 2:", x)
x -= 3  # x = x - 3
print("x -= 3:", x)
x *= 4  # x = x * 4
print("x *= 4:", x)
x /= 2  # x = x / 2
print("x /= 2:", x)
x //= 2 # x = x // 2
print("x //= 2:", x)
x %= 3 # x = x % 3
print("x %= 3:", x)
x **= 2 # x = x ** 2
print("x **= 2:", x)

# Bitwise operators (demonstrated with integers)
print("\nBitwise Operators (using integers):")

a = 0b10101100
b = 0b10011001
print("Binary Input 1", bin(a))
print("Binary Input 2", bin(b))
print("a & b:", bin(a & b))
print("a | b:", bin(a | b))
print("a ^ b:", bin(a ^ b))
print("~a:", bin(~a))
print("a << 2:", bin(a << 2)) # Left shift
print("a >> 2:", bin(a >> 2))  # Right shift

# Conditional Statements
print("\nConditional Statements:")

if num1 > num2:
    print(f"{num1} is greater than {num2}")
elif num1 < num2:
    print(f"{num1} is less than {num2}")
else:
    print(f"{num1} is equal to {num2}")

# Nested if-else
if num1 > 0:
  if num1 % 2 == 0:
    print(f"{num1} is positive and even")
  else:
    print(f"{num1} is positive and odd")
elif num1 < 0:
    print(f"{num1} is negative")
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
else:
  print(f"{num1} is zero")

# For loop
print("\nLoop Demonstration")
for i in range(1, 5):
    print(i)

# While loop
count = 0
while count < 3:
    print("Count:", count)
    count += 1
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage of operators and conditional statements was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	CONTROL STRUCTURES

AIM:

To develop a Python program to demonstrate the usage of control structures using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter input number.
Step 3: Compute the results for various control structures.
Step 4: Print the computed result.
Step 5: Stop the program.

PROGRAM:

```
#Input
x = int(input("Enter the Input number : "))

# Conditional statements (if, elif, else)

print("Conditional statements (if, elif, else)")
if x > 5:
  print("x is greater than 5")
elif x == 5:
  print("x is equal to 5")
else:
  print("x is less than 5")

# Loops (for loop)
print("For loop")
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
  print(fruit)

# Loops (while loop)
print("while loop")
i = 0
while i < 5:
  print(i)
  i += 1

# Loop control statements (break)
print("break statement")
for i in range(10):
  if i == 5:
    break
  print(i)
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
# Loop control statements (continue)
print("continue statement")
for i in range(10):
  if i % 2 == 0:
    continue  # Skip even numbers
  print(i)

# Exception handling (try, except, finally)
print("Exception handling")

try:
  result = 10 / 0
except ZeroDivisionError:
  print("Division by zero error!")
finally:
  print("This always executes.")

# Function definition and call
print("Function")

def greet(name):
  """This function greets the person passed in as a parameter."""
  print(f"Hello, {name} How Are You?")

greet("Hariharan")

# List comprehension
print("List comprehension")
numbers = [1, 2, 3, 4, 5]
squared_numbers = [number**2 for number in numbers]
print(squared_numbers)

# Dictionary comprehension
print("Dictionary comprehension")
numbers = [1, 2, 3, 4, 5]
number_squares = {number: number**2 for number in numbers}
print(number_squares)

# Pass statement (used as a placeholder)
print("Pass statement")
for i in range(5):
    pass #do nothing

# Match-case statement (Python 3.10+)
print("Match-case statement")
def http_error(status):
    match status:
        case 400:
            return "Bad request"
        case 404:
            return "Not found"
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
        case 418:
            return "I'm a teapot"
        case _:
            return "Something's wrong with the internet"

print(http_error(404))
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage of control structures was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	ARRAY OPERATION

AIM:

To develop a Python program to array operation using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter input array.
Step 3: Compute the results for various array operations.
Step 4: Print the computed result.
Step 5: Stop the program.

PROGRAM:

```
# Taking user input for array elements
arr = list(map(int, input("Enter array elements separated by space: ").split()))

# Display the array
print("Array:", arr)

# Sum of elements
sum_arr = 0
for num in arr:
    sum_arr += num
print("Sum of elements:", sum_arr)

# Maximum element
max_arr = arr[0]
for num in arr:
    if num > max_arr:
        max_arr = num
print("Maximum element:", max_arr)

# Minimum element
min_arr = arr[0]
for num in arr:
    if num < min_arr:
        min_arr = num
print("Minimum element:", min_arr)

# Reverse the array
rev_arr = []
for i in range(len(arr) - 1, -1, -1):
    rev_arr.append(arr[i])
print("Reversed array:", rev_arr)

# Sorting the array (Bubble Sort)
sorted_arr = arr[:]  # Copy original array
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
for i in range(len(sorted_arr)):
    for j in range(len(sorted_arr) - i - 1):
        if sorted_arr[j] > sorted_arr[j + 1]:
            sorted_arr[j], sorted_arr[j + 1] = sorted_arr[j + 1], sorted_arr[j]
print("Sorted array (Ascending):", sorted_arr)

# Sorting in descending order
sorted_arr_desc = arr[:]  # Copy original array
for i in range(len(sorted_arr_desc)):
    for j in range(len(sorted_arr_desc) - i - 1):
        if sorted_arr_desc[j] < sorted_arr_desc[j + 1]:
            sorted_arr_desc[j], sorted_arr_desc[j + 1] = sorted_arr_desc[j + 1], sorted_arr_desc[j]
print("Sorted array (Descending):", sorted_arr_desc)


# Checking if an element exists in the array
search_num = int(input("Enter number to search: "))
found = False
for num in arr:
    if num == search_num:
        found = True
        break
print(f"Element {search_num} exists in array:", found)

# Count occurrences of an element
count_num = int(input("Enter number to count occurrences: "))
count = 0
for num in arr:
    if num == count_num:
        count += 1
print(f"Occurrences of {count_num}:", count)

# Removing duplicates from array
unique_arr = []
for num in arr:
    if num not in unique_arr:
        unique_arr.append(num)
print("Array without duplicates:", unique_arr)
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to array operation was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	STRING FUNCTIONS

AIM:

To develop a Python program to demonstrate the usage of String functions using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter input array.
Step 3: Compute the results for various array operations.
Step 4: Print the computed result.
Step 5: Stop the program.

PROGRAM:

```
# Taking user input for a string
s = input("Enter a string: ")

# Display the original string
print("Original String:", s)

# Convert to uppercase
upper_s = ""
for char in s:
    if 'a' <= char <= 'z':  # Check if lowercase
        upper_s += chr(ord(char) - 32)  # Convert to uppercase
    else:
        upper_s += char
print("Uppercase String:", upper_s)

# Convert to lowercase
lower_s = ""
for char in s:
    if 'A' <= char <= 'Z':  # Check if uppercase
        lower_s += chr(ord(char) + 32)  # Convert to lowercase
    else:
        lower_s += char
print("Lowercase String:", lower_s)

# Find the length of the string
length = 0
for char in s:
    length += 1
print("Length of String:", length)

# Reverse the string
reversed_s = ""
for i in range(length - 1, -1, -1):
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
reversed_s += s[i]
print("Reversed String:", reversed_s)

# Count occurrences of a character
char_to_count = input("Enter a character to count its occurrences: ")
count = 0
for char in s:
    if char == char_to_count:
        count += 1
print(f"Occurrences of '{char_to_count}':", count)

# Check if the string is a palindrome
is_palindrome = True
for i in range(length // 2):
    if s[i] != s[length - i - 1]:
        is_palindrome = False
        break
print("Is Palindrome:", is_palindrome)

# Remove spaces from the string
no_space_s = ""
for char in s:
    if char != " ":
        no_space_s += char
print("String without spaces:", no_space_s)

# Find if a substring exists
substring = input("Enter a substring to search: ")
found = False
for i in range(length - len(substring) + 1):
    if s[i:i+len(substring)] == substring:
        found = True
        break
print(f"Substring '{substring}' found:", found)

# Replace a character
char_to_replace = input("Enter character to replace: ")
replace_with = input("Enter replacement character: ")
replaced_s = ""
for char in s:
    if char == char_to_replace:
        replaced_s += replace_with
    else:
        replaced_s += char
print("String after replacement:", replaced_s)

# Convert first letter of each word to uppercase (Title Case)
title_s = ""
capitalize_next = True
for char in s:
    if char == " ":
        capitalize_next = True
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
  title_s += char
    elif capitalize_next and 'a' <= char <= 'z':
        title_s += chr(ord(char) - 32)  # Convert to uppercase
        capitalize_next = False
    else:
        title_s += char
print("Title Case String:", title_s)

# Check if string contains only digits
is_digit = True
for char in s:
    if not ('0' <= char <= '9'):
        is_digit = False
        break
print("Contains only digits:", is_digit)
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage of String functions was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	CLASSES AND FUNCTIONS

AIM:

To develop a Python program to demonstrate the usage of classes and functions using the interactive shell.

ALGORITHM

Step: 1. Start the program.
Step: 2. Ask user to enter account holder name
Step: 3. Set balance to 0
Step: 4. Repeat the following steps:
- a. Show menu (Deposit, Withdraw, Show Balance, Exit)
- b. Ask user to choose an option
- c. If user chooses Deposit:
  - i. Ask for amount
  - ii. Add amount to balance
  - iii. Show new balance
- d. If user chooses Withdraw:
  - i. Ask for amount
  - ii. If amount is more than balance, show "Insufficient balance"
  - iii. Else, subtract amount and show new balance
- e. If user chooses Show Balance:
  - i. Display account holder name and balance
- f. If user chooses Exit:
  - i. End the loop

Step: 5. Stop the program.

PROGRAM:

```
# Define the BankAccount class
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"₹{amount} deposited. New balance: ₹{self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
        elif amount <= 0:
            print("Withdrawal amount must be positive.")
        else:
            self.balance -= amount
            print(f"₹{amount} withdrawn. New balance: ₹{self.balance}")

    def display_balance(self):
        print(f"Account Holder: {self.account_holder}")
        print(f"Current Balance: ₹{self.balance}")


# Function to interact with the user
def main():
    name = input("Enter account holder name: ")
    account = BankAccount(name)

    while True:
        print("\n1. Deposit")
        print("2. Withdraw")
        print("3. Display Balance")
        print("4. Exit")

        choice = input("Enter your choice: ")
        if choice == '1':
            amount = float(input("Enter amount to deposit: "))
            account.deposit(amount)
        elif choice == '2':
            amount = float(input("Enter amount to withdraw: "))
            account.withdraw(amount)
        elif choice == '3':
            account.display_balance()
        elif choice == '4':
            print("Thank you for using the bank system!")
            break
        else:
            print("Invalid choice. Try again!")
# Call the main function
if __name__ == "__main__":
    main()
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage of classes and functions was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

Exp. No. : Date:	IMPLEMENT RECURSIVE FUNCTION

AIM:

To develop a Python program to implement recursive function using the interactive shell.

ALGORITHM

Step: 1. Start the program.
Step: 2. Read a number n
Step: 3. If n is 0 or 1, return 1
Step: 4. Else, return n * factorial(n - 1)
Step: 5. Display the result
Step: 6. Stop the program.

PROGRAM:

```
# Recursive function to find factorial
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Main program
num = int(input("Enter a number to find its factorial: "))

if num < 0:
    print("Factorial is not defined for negative numbers.")
else:
    result = factorial(num)
    print(f"The factorial of {num} is {result}")
```

DEPARTMENT OF BME		
PREPARATION	15	
PERFORMANCE	15	
RESULT	10	
VIVA	10	
TOTAL	50	
SIGNATURE		

RESULT:

The Python program to implement recursive function was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

Exp. No. : Date:	IMPLEMENT LAMBDA FUNCTION

AIM:

To develop a Python program to implement lambda function using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Compute the results for various lambda function.
Step 3: Print the computed result.
Step 4: Stop the program.

PROGRAM:

```
from functools import reduce  # Needed for reduce()

# 1. Simple lambda function (square of a number)
square = lambda x: x * x
print("Square of 6 is:", square(6))
# 2. Lambda with two arguments (sum of two numbers)
add = lambda a, b: a + b
print("Sum of 4 and 5 is:", add(4, 5))
# 3. Lambda with map() – double all numbers in a list
numbers = [1, 2, 3, 4, 5]
doubled = list(map(lambda x: x * 2, numbers))
print("Doubled numbers:", doubled)
# 4. Lambda with filter() – filter even numbers
even = list(filter(lambda x: x % 2 == 0, numbers))
print("Even numbers:", even)
# 5. Lambda with reduce() – find the product of all numbers
product = reduce(lambda x, y: x * y, numbers)
print("Product of all numbers:", product)
```

DEPARTMENT OF BME		
PREPARATION	15	
PERFORMANCE	15	
RESULT	10	
VIVA	10	
TOTAL	50	
SIGNATURE		

RESULT:

The Python program to implement lambda function was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	FILE MANIPULATION

AIM:

To develop a Python program to demonstrate the usage file manipulation using the interactive shell.

ALGORITHM

Step: 1. Start the program.
Step: 2. Display options for file operations (create, read, append, rename, delete, exit).
Step: 3. Get user choice and request appropriate input (filename, content, etc.).
Step: 4. Perform the chosen operation (create, read, append, rename, delete) using the file operations.
Step: 5. Show result of the operation (success or failure message).
Step: 6. Repeat or Exit based on user choice.

PROGRAM:

```
import os
# 1. Create and write to a file
def create_and_write_file(filename, content):
    with open(filename, 'w') as file:
        file.write(content)
    print(f"File '{filename}' created and content written.")

# 2. Read from a file
def read_file(filename):
    try:
        with open(filename, 'r') as file:
            content = file.read()
        print(f"Content of '{filename}':\n{content}")
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
# 3. Append to a file
def append_to_file(filename, content):
    with open(filename, 'a') as file:
        file.write(content)
    print(f"Content appended to '{filename}'.")
# 4. Rename a file
def rename_file(old_filename, new_filename):
    try:
        os.rename(old_filename, new_filename)
        print(f"File renamed from '{old_filename}' to '{new_filename}'.")
    except FileNotFoundError:
        print(f"File '{old_filename}' not found.")
# 5. Delete a file
def delete_file(filename):
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
try:
        os.remove(filename)
        print(f"File '{filename}' deleted.")
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
# Main program with user input
def main():
    while True:
        print("\nChoose an operation:")
        print("1. Create and Write to File")
        print("2. Read from File")
        print("3. Append to File")
        print("4. Rename File")
        print("5. Delete File")
        print("6. Exit")
        choice = input("Enter your choice (1-6): ")
        if choice == '1':
            filename = input("Enter the filename to create: ")
            content = input("Enter the content to write: ")
            create_and_write_file(filename, content)
        elif choice == '2':
            filename = input("Enter the filename to read: ")
            read_file(filename)
        elif choice == '3':
            filename = input("Enter the filename to append to: ")
            content = input("Enter the content to append: ")
            append_to_file(filename, content)
        elif choice == '4':
            old_filename = input("Enter the current filename to rename: ")
            new_filename = input("Enter the new filename: ")
            rename_file(old_filename, new_filename)
        elif choice == '5':
            filename = input("Enter the filename to delete: ")
            delete_file(filename)
        elif choice == '6':
            print("Exiting program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 6.")
# Run the program
if __name__ == "__main__":
    main()
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage file manipulation was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Exp. No. : Date:	LISTS, SETS, DICTIONARIES AND TUPLES

AIM:

To develop a Python program to demonstrate the usage of lists, sets, dictionaries and tuples using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Prompt the user to enter input array.
Step 3: Compute the results for various operations of lists, sets, dictionaries.
Step 4: Print the computed result.
Step 5: Stop the program.

PROGRAM:

```
def list_operations():
    my_list = []
    while True:
        print("\n--- List Operations ---")
        print("1. Add an item to the list")
        print("2. Remove an item from the list")
        print("3. Display the list")
        print("4. Exit")
        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            item = input("Enter an item to add to the list: ")
            my_list.append(item)
            print(f"Item '{item}' added to the list.")
        elif choice == '2':
            item = input("Enter an item to remove from the list: ")
            if item in my_list:
                my_list.remove(item)
                print(f"Item '{item}' removed from the list.")
            else:
                print(f"Item '{item}' not found in the list.")
        elif choice == '3':
            print("Current list:", my_list)
        elif choice == '4':
            break
        else:
            print("Invalid choice, please try again.")

def set_operations():
    my_set = set()
    while True:
        print("\n--- Set Operations ---")
        print("1. Add an item to the set")
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
print("2. Remove an item from the set")
        print("3. Display the set")
        print("4. Exit")
        choice = input("Enter your choice (1-4): ")
        if choice == '1':
            item = input("Enter an item to add to the set: ")
            my_set.add(item)
            print(f"Item '{item}' added to the set.")
        elif choice == '2':
            item = input("Enter an item to remove from the set: ")
            if item in my_set:
                my_set.remove(item)
                print(f"Item '{item}' removed from the set.")
            else:
                print(f"Item '{item}' not found in the set.")
        elif choice == '3':
            print("Current set:", my_set)
        elif choice == '4':
            break
        else:
            print("Invalid choice, please try again.")
def dict_operations():
    my_dict = {}
    while True:
        print("\n--- Dictionary Operations ---")
        print("1. Add or update a key-value pair")
        print("2. Remove a key-value pair")
        print("3. Display the dictionary")
        print("4. Exit")
        choice = input("Enter your choice (1-4): ")
        if choice == '1':
            key = input("Enter the key: ")
            value = input("Enter the value: ")
            my_dict[key] = value
            print(f"Key '{key}' with value '{value}' added/updated in the dictionary.")
        elif choice == '2':
            key = input("Enter the key to remove: ")
            if key in my_dict:
                del my_dict[key]
                print(f"Key '{key}' removed from the dictionary.")
            else:
                print(f"Key '{key}' not found in the dictionary.")
        elif choice == '3':
            print("Current dictionary:", my_dict)
        elif choice == '4':
            break
        else:
            print("Invalid choice, please try again.")
def tuple_operations():
    my_tuple = ()
    while True:
```

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

```
print("\n--- Tuple Operations ---")
```

```
        print("1. Add an item to the tuple (create a new tuple)")
        print("2. Display the tuple")
        print("3. Exit")
        choice = input("Enter your choice (1-3): ")
        if choice == '1':
            item = input("Enter an item to add to the tuple: ")
            my_tuple += (item,)
            print(f"Item '{item}' added to the tuple.")
        elif choice == '2':
            print("Current tuple:", my_tuple)
        elif choice == '3':
            break
        else:
            print("Invalid choice, please try again.")
def main():
    while True:
        print("\n--- Choose Data Structure ---")
        print("1. List Operations")
        print("2. Set Operations")
        print("3. Dictionary Operations")
        print("4. Tuple Operations")
        print("5. Exit")
        choice = input("Enter your choice (1-5): ")
        if choice == '1':
            list_operations()
        elif choice == '2':
            set_operations()
        elif choice == '3':
            dict_operations()
        elif choice == '4':
            tuple_operations()
        elif choice == '5':
            print("Exiting program. Goodbye!")
            break
        else:
            print("Invalid choice, please try again.")
# Run the program
if __name__ == "__main__":
    main()
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to demonstrate the usage of lists, sets, dictionaries and tuples String functions was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

Exp. No. : Date:	IMPLEMENT FUNCTION

AIM:

To develop a Python program to implement function using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Compute the results for various lambda function.
Step 3: Print the computed result.
Step 4: Stop the program.

PROGRAM:

```
# Function template to check if a number is prime
def is_prime(n):

    if n <= 1:
        return False  # Numbers less than or equal to 1 are not prime
    for i in range(2, int(n ** 0.5) + 1):  # Check divisibility from 2 to the square root of n
        if n % i == 0:  # If n is divisible by any number in this range, it's not prime
            return False
    return True  # If no divisors are found, n is prime


# Main program to get user input and call the function
print("Prime Number Checker")

# Get user input
num = int(input("Enter a number to check if it's prime: "))

# Call the function with the user input
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

DEPARTMENT OF BME		
PREPARATION	15	
PERFORMANCE	15	
RESULT	10	
VIVA	10	
TOTAL	50	
SIGNATURE		

RESULT:

The Python program to implement function was successfully executed, and the output was verified.

SAMPLE INPUT AND OUTPUT:

Leet code Problem:
Leet code Program:
Leet code Sample input and Output:

Exp. No. : Date:	IMPLEMENT CLASSES

AIM:

To develop a Python program to implement classes using the interactive shell.

ALGORITHM

Step 1: Start the program.
Step 2: Compute the results for various classes.
Step 3: Print the computed result.
Step 4: Stop the program.

PROGRAM:

```
# Define a class
class Student:
    # Constructor
    def __init__(self, name, roll_number, grade):
        self.name = name
        self.roll_number = roll_number
        self.grade = grade
    # Method to display student info
    def display_info(self):
        print("Student Details:")
        print(f"Name       : {self.name}")
        print(f"Roll No.   : {self.roll_number}")
        print(f"Grade      : {self.grade}")
    # Method to update grade
    def update_grade(self, new_grade):
        self.grade = new_grade
        print(f"{self.name}'s grade updated to {self.grade}")
# Create objects of the class
student1 = Student("Anjali", 101, "A")
student2 = Student("Rahul", 102, "B")
# Use the methods
student1.display_info()
student2.display_info()
print("\n-- Updating Grade --")
student2.update_grade("A")
student2.display_info()
```

DEPARTMENT OF BME		
PREPARATION	**15**	
PERFORMANCE	**15**	
RESULT	**10**	
VIVA	**10**	
TOTAL	**50**	
SIGNATURE		

RESULT:

The Python program to implement classes was successfully executed, and the output was verified.