GATE Institute of Technology and Management Sciences::TIRUPATI

MCA 102A – Programming in Python: Complete Lab Manual

S.NO	Experiment
1.	Write a Python program to print 'Hello, World!'.
2.	Write a Python program to count frequency of characters in a string.
3.	Write a Python program to generate Pascal's Triangle.
4.	Write a Python program to find most frequent word in a list.
5.	Write a Python program to implement a simple Bank Account using OOP.
6.	Write a Python program to Find the Factorial of a given Number.
7.	Write a Python program to Find the largest element among three numbers.
8.	Write a Program to display all prime numbers within an interval.
9.	Write a Program to Calculate LCM of Two numbers in Python.
10.	Write a Python Program to Print the Fibonacci sequence of a given Number.
11.	Write a Python Program to Print the Sum of Natural Numbers of a given Number.
12.	Write a Python Program for Basic Mathematical Operations.
13.	Write a Python Program to Check Palindrome String or not a Palindrome String for a given string.
14.	Write a Python Program to add two matrices using nested loops read matrix values form keyboard.
15.	Write a Python Program to find all Armstrong numbers for given Interval.

Experiment 1: Write a Python program to print 'Hello, World!'

Aim

Write a Python program to print 'Hello, World!'

Algorithm

1. Start
2. Print message
3. End

Pseudocode

```
BEGIN
PRINT 'Hello, World!'
END
```

Experiment 2: Write a Python program to count frequency of characters in a string

Aim

Write a Python program to count frequency of characters in a string

Algorithm

1. Start
2. Read string
3. For each char count occurrences
4. Print result
5. End

Pseudocode

```
BEGIN
INPUT string
FOR each character
COUNT frequency
PRINT result
END
```

Experiment 3: Write a Python program to generate Pascal's Triangle.

Aim

Write a Python program to generate Pascal's Triangle

Algorithm

1. Start
2. Input rows
3. Generate triangle iteratively
4. Print rows
5. End

Pseudocode

BEGIN
INPUT rows
SET triangle
FOR i in rows
COMPUTE values
PRINT
END

Experiment 4: Write a Python program to find most frequent word in a list

Aim

Write a Python program to find most frequent word in a list

Algorithm

1. Start
2. Input list
3. Count frequency of each word
4. Find max
5. Print result
6. End

Pseudocode

BEGIN
INPUT list
COUNT frequency
FIND max
PRINT result
END

Experiment 5: Write a Python program to implement a simple Bank Account using OOP

Aim

Write a Python program to implement a simple Bank Account using OOP

Algorithm

1. Start
2. Define class with deposit, withdraw, balance
3. Create object
4. Call methods
5. End

Pseudocode

BEGIN
DEFINE class
METHODS deposit, withdraw, balance
CREATE object
CALL methods
END

Experiment 6: Write a Python program to Find the Factorial of a given Number.

Aim

Write a Python program to Find the Factorial of a given Number.

Algorithm

Step 1: Start the program.
Step 2: Read an integer n from the user.
Step 3: Initialize a variable factorial with value 1.
Step 4: Check if n is negative:

- If n < 0, print "Factorial is not defined for negative numbers" and go to Step 7.
  Step 5: If n >= 0, do the following:
- For each integer i from 1 to n, multiply factorial by i.

factorial = factorial * i
Step 6: Print "Factorial of n is factorial".
Step 7: End the program.

Pseudocode

Step1:Takes input

Step2:Initializes factorial

Step3:Handles negative numbers

Step4:Uses a for loop to calculate factorial

Step:5Prints the result

Experiment 7: Write a Python program to Find the largest element among three numbers.

Aim: To Write a Python program to Find the largest element among three numbers.

Algorithm

Step 1: Start the program.
Step 2: Read three numbers num1, num2, and num3 from the user.
Step 3: Compare the numbers to find the largest:

- If num1 > num2 and num1 > num3, then largest = num1.
- Else if num2 > num3, then largest = num2.
- Else, largest = num3.

Step 4: Print "The largest number is largest".
Step 5: End the program.

Pseudocode

START

```
1.INPUT num1, num2, num3   // Read three numbers from the user
2.IF num1 > num2 AND num1 > num3 THEN
    largest = num1
3.ELSE IF num2 > num3 THEN
    largest = num2
ELSE
    largest = num3
END IF
4.PRINT "The largest number is", largest
    END
```

Experiment 8: Write a Program to display all prime numbers within an interval

Aim:To Write a Program to display all prime numbers within an interval

Algorithm

Step 1: Start the program.

Step 2: Read two integers from the user: lower (start of range) and upper (end of range).

Step 3: Print a message: "Prime numbers between lower and upper are:".

Step 4: Loop through each number num from lower to upper:

- Step 4.1: If num is greater than 1, proceed (since primes are greater than 1).
- Step 4.2: Check if num is divisible by any integer i from 2 to num-1:
    - If num % i == 0, then num is not prime → break the inner loop.
- Step 4.3: If the inner loop completes without breaking, then num is prime → print num.

Step 5: Repeat for all numbers in the range.

Step 6: End the program.

Pseudocode

```
FOR num FROM lower TO upper DO
    IF num > 1 THEN
      SET isPrime = True

      FOR i FROM 2 TO num - 1 DO
        IF num MOD i == 0 THEN
          SET isPrime = False
          BREAK
        END IF
      END FOR

      IF isPrime == True THEN
        PRINT num
      END IF
```

```
    END IF
  END FOR
END
```

Experiment 9: Write a Program to Calculate LCM of Two numbers in Python.

Aim:To Write a Program to Calculate LCM of Two numbers in Python.

Algorithm

Step 1: Start the program.

Step 2: Read two integers from the user, num1 and num2.

Step 3: Find the greater number between num1 and num2 and assign it to lcm.

Step 4: Repeat the following until LCM is found:

- If lcm is divisible by both num1 and num2 (lcm % num1 == 0 and lcm % num2 == 0), then lcm is found → go to Step 5.
- Else, increment lcm by 1 and check again.

Step 5: Print "The LCM of num1 and num2 is lcm".

Step 6: End the program.

Pseudocode

```
START

  INPUT num1   // First number

  INPUT num2   // Second number

  // Step 1: Find the greater number to start checking

  IF num1 > num2 THEN

    lcm = num1

  ELSE

    lcm = num2

  END IF
```

```
  // Step 2: Find LCM
  WHILE True DO
    IF lcm MOD num1 == 0 AND lcm MOD num2 == 0 THEN
      BREAK   // LCM found
    ELSE
      lcm = lcm + 1   // Increment lcm and check again
    END IF
  END WHILE
  PRINT "LCM of", num1, "and", num2, "is", lcm
END
```

Experiment 10: Write a Python Program to Print the Fibonacci sequence of a given Number

Aim:To Write a Python Program to Print the Fibonacci sequence of a given Number.

Algorithm

Step 1: Start the program.

Step 2: Read an integer n from the user, which represents the number of terms to print.

Step 3: Initialize two variables:

- **a = 0 (first term)**
- **b = 1 (second term)**

Step 4: Print a message: "Fibonacci sequence:".

Step 5: Loop from i = 0 to i < n:

- **Print the value of a.**

- **Calculate the next term: next_term = a + b.**
- **Update the variables:**
    - **a = b**
    - **b = next_term**

Step 6: Repeat Step 5 until all n terms are printed.

Step 7: End the program.

Pseudocode

```
START
  INPUT n   // Number of terms to print

  // Step 1: Initialize first two terms
  a = 0
  b = 1
  PRINT "Fibonacci sequence:"
  // Step 2: Loop to generate Fibonacci numbers
  FOR i FROM 0 TO n-1 DO
     PRINT a
     next_term = a + b
     a = b
     b = next_term
  END FOR
END
```

Experiment 11: Write a Python Program to Print the Sum of Natural Numbers of a given Number.

Aim:To Write a Python Program to Print the Sum of Natural Numbers of a given Number.

Algorithm

Step1: Start.

Step 2: Display the message: "Enter a non-negative integer n:".

Step 3: Read the user input as a string (call it input_str).

Step 4:Try to convert input_str to integer n.

- **If conversion fails (ValueError), display "Invalid input — enter an integer." and Stop.**

Step 5: If n < 0, display "Enter a non-negative integer." and Stop.

Step 6: Set sum = 0.

Step 7: Set i = 1.

Step8: While i <= n, do:

- **sum = sum + i**
- **i = i + 1**

Step9: End while. At this point sum holds the total.

Step 10: Display "Sum = " followed by the value of sum.

Step 11:Stop.

Pseudocode

```
START
  PRINT "Enter a non-negative integer n:"
  READ n
   IF n < 0 THEN
    PRINT "Enter a non-negative integer"
    STOP
  ENDIF
  sum ← 0
    FOR i ← 1 TO n DO
    sum ← sum + i
  ENDFOR
  PRINT "Sum =", sum
STOP
```

Experiment 12 Write a Python Program for Basic Mathematical Operations.

Aim: To a Write a Python Program for Basic Mathematical Operations

Algorithm

Start

1. **Display a menu of operations:**
    1. **Add**
    2. **Subtract**
    3. **Multiply**
    4. **Divide**
    5. **Exit**
2. **Repeat until the user chooses to exit (option 5):**

1. **Input the user's choice of operation.**
2. **If choice is 5, exit the program.**
3. **If choice is one of 1, 2, 3, 4:**
    1. **Input the first number (num1).**
    2. **Input the second number (num2).**
    3. **Check if input is numeric; if not, display error and repeat.**
    4. **Perform the selected operation:**
        - **1: Add → num1 + num2**
        - **2: Subtract → num1 - num2**
        - **3: Multiply → num1 * num2**
        - **4: Divide → Check if num2 != 0 then num1 / num2; else display "Division by zero error"**
    5. **Display the result.**
4. **Else (choice not 1–5), display error message "Invalid choice".**

5. Ask the user if they want to perform another calculation.

- **If yes, repeat from step 2.**
- **If no, exit the program.**

Stop

Pseudocode

```
START
FUNCTION add(x, y)
  RETURN x + y
END FUNCTION
FUNCTION subtract(x, y)
  RETURN x - y
END FUNCTION
FUNCTION multiply(x, y)
  RETURN x * y
END FUNCTION
FUNCTION divide(x, y)
  IF y = 0 THEN
    RETURN "Error! Division by zero"
  ELSE
    RETURN x / y
  ENDIF
END FUNCTION
LOOP
  DISPLAY "Select operation:"
  DISPLAY "1. Add"
```

```
DISPLAY "2. Subtract"
DISPLAY "3. Multiply"
DISPLAY "4. Divide"
DISPLAY "5. Exit"
READ choice
IF choice = 5 THEN
  DISPLAY "Exiting program. Thank you!"
  BREAK
ENDIF
IF choice IN (1, 2, 3, 4) THEN
  READ num1
  READ num2
  IF choice = 1 THEN
    result ← add(num1, num2)
    DISPLAY num1 + " + " + num2 + " = " + result
  ELSE IF choice = 2 THEN
    result ← subtract(num1, num2)
    DISPLAY num1 + " - " + num2 + " =
```

Experiment 13 Write a Python Program to Check Palindrome String or not a Palindrome String for a given string

Aim: To Write a Python Program to Check Palindrome String or not a Palindrome String for a given string

Algorithm

Start

1. **Input a string from the user.**
2. **Preprocess the string:**
    - **Remove spaces.**
    - **Convert all characters to lowercase.**
3. **Reverse the processed string.**
4. **Compare the original processed string with the reversed string:**
    - **If both are the same → The string is a palindrome.**
    - **Otherwise → The string is not a palindrome.**
5. **Display the result.**

Pseudocode

START

DISPLAY "Enter a string:"

READ string

 # Preprocess the string

clean_string ← REMOVE_SPACES(string)

clean_string ← CONVERT_TO_LOWERCASE(clean_string)

 # Reverse the string

reversed_string ← REVERSE(clean_string)

 # Check if the string is a palindrome

IF clean_string = reversed_string THEN

DISPLAY string + " is a palindrome."

ELSE

DISPLAY string + " is not a palindrome."

ENDIF

STOP

Experiment 14 Write a Python Program to add two matrices using nested loops read matrix values form keyboard.

Aim: To Write a Python Program to add two matrices using nested loops read matrix values form keyboard.

Algorithm

Start

1. **Input the number of rows (rows) and columns (cols) for the matrices.**
2. **Initialize two empty matrices: matrix1 and matrix2.**

Input Elements of First Matrix

3. **For each row i from 0 to rows-1:**
    1. **Initialize an empty list row.**
    2. **For each column j from 0 to cols-1:**
        - **Input element [i+1, j+1] from the user.**
        - **Append the element to row.**
    3. **Append row to matrix1.**

Input Elements of Second Matrix

4. **Repeat the same steps as above to input elements into matrix2.**

Add the Matrices

5. **Initialize an empty matrix result.**
6. **For each row i from 0 to rows-1:**
    1. **Initialize an empty list row_sum.**
    2. **For each column j from 0 to cols-1:**
        - **Compute sum = matrix1[i][j] + matrix2[i][j].**
        - **Append sum to row_sum.**
    3. **Append row_sum to result.**

Display Result

7. **For each row in result, display the row.**

Stop

Pseudocode

```
START
# Input matrix size
DISPLAY "Enter number of rows:"
READ rows
DISPLAY "Enter number of columns:"
READ cols
# Initialize matrices
matrix1 ← empty list
matrix2 ← empty list
# Input elements of first matrix
DISPLAY "Enter elements of first matrix:"
FOR i FROM 0 TO rows-1
  row ← empty list
```

```
    FOR j FROM 0 TO cols-1
        DISPLAY "Enter element [", i+1, ",", j+1, "]:"
        READ num
        APPEND num TO row
    END FOR
    APPEND row TO matrix1
END FOR
# Input elements of second matrix
DISPLAY "Enter elements of second matrix:"
FOR i FROM 0 TO rows-1
    row ← empty list
    FOR j FROM 0 TO cols-1
        DISPLAY "Enter element [", i+1, ",", j+1, "]:"
        READ num
        APPEND num TO row
    END FOR
    APPEND row TO matrix2
END FOR
# Add matrices
result ← empty list
FOR i FROM 0 TO rows-1
    row_sum ← empty list
    FOR j FROM 0 TO cols-1
        sum ← matrix1[i][j] + matrix2[i][j]
```

```
        APPEND sum TO row_sum
    END FOR
    APPEND row_sum TO result
END FOR
# Display the result
DISPLAY "Sum of the two matrices:"
FOR EACH row IN result
    DISPLAY row
END FOR
STOP
```

Experiment 15 Write a Python Program to find all Armstrong numbers for given Interval.

Aim: To Write a Python Program to find all Armstrong numbers for given Interval.

Algorithm

Start

1. **Input the lower bound (lower) and upper bound (upper) of the interval.**
2. **Display a message: "Armstrong numbers between lower and upper are:".**

Check Each Number in Interval

3. **For each number num from lower to upper (inclusive):**
    1. **Calculate the number of digits in num → num_digits.**
    2. **Initialize sum_of_powers = 0.**
    3. **Set a temporary variable temp = num.**
    4. **While temp > 0:**
        - **Extract the last digit → digit = temp % 10.**
        - **Add digit ** num_digits to sum_of_powers.**
        - **Remove the last digit → temp = temp // 10.**
    5. **If sum_of_powers == num:**
        - **Display num as an Armstrong number.**

Stop

4. **End of program after checking all numbers in the interval.**

Pseudocode

START

DISPLAY "Enter the lower bound:"

READ lower

```
DISPLAY "Enter the upper bound:"
READ upper
DISPLAY "Armstrong numbers between", lower, "and", upper, "are:"
FOR num FROM lower TO upper DO
   num_digits ← LENGTH_OF(num)        # Count the number of digits
   sum_of_powers ← 0
   temp ← num
   WHILE temp > 0 DO
      digit ← temp MOD 10
      sum_of_powers ← sum_of_powers + (digit ^ num_digits)
      temp ← temp DIV 10
   END WHILE
   IF sum_of_powers = num THEN
      DISPLAY num
   END IF
END FOR
STOP
```