GATE INSTITUTE OF TECHNOLOGY & MANAGEMENT SCIENCES

(Approved by AICTE , Affiliated to Sri Venkateswara University , TIRUPATI)

ICET CODE : GTMT

Master of Computer Applications

SEMESTER - I

Hall Ticket No: _____

Name of the Student: K. RajaSree

Subject: programming in python

Year: I Semester: I Batch 2025 - 2026

GATE EDUCATIONAL INSTITUTIONS

DEPARTMENT OF COMPUTERS

YEAR I SEMESTER I

Reg No._____

CERTIFICATE

Certified that this is the bonafide record of practical work done in the laboratory by the candidate K. RajaSree *during the academic year* 2025 - 2027 *subject* programming in python.

No. of Practical's Conducted 15 ***No. of Practical's Attended*** 15

LECTURER **HEAD OF THE DEPARTMENT**

Submitted for the Practical Examination held on_____

Examiners

1.

2.

INDEX

INDEX

Hello World

1. Write a python program to print "Hello, World!"

Aim :-

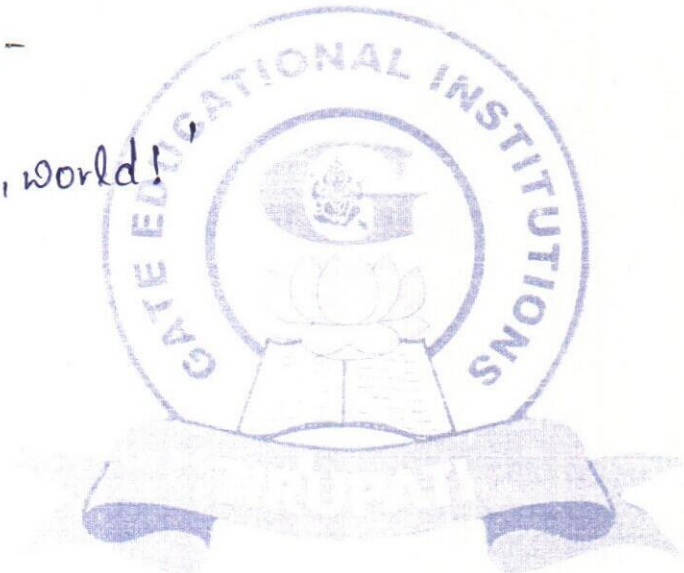
Write a python program to print 'Hello, World!'

Algorithm :-

1. Start
2. print message
3. End

pseudocode :-

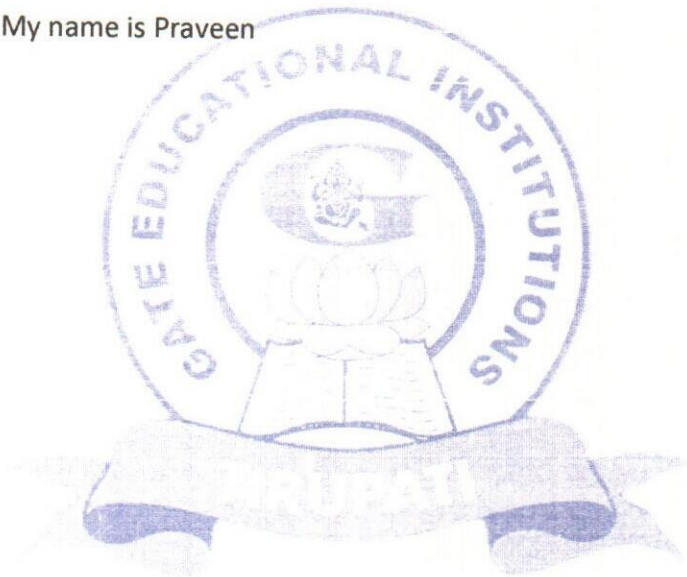
```
BEGIN
PRINT 'Hello, World!'
END
```



```
def main():
    print('Hello, World! My name is Praveen')

if __name__ == '__main__':
    main()
```

Sample Input & Expected Output

Input: (none)
Output: Hello, World! My name is Praveen



Count frequency of characters in a String

Practical No. : 2

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

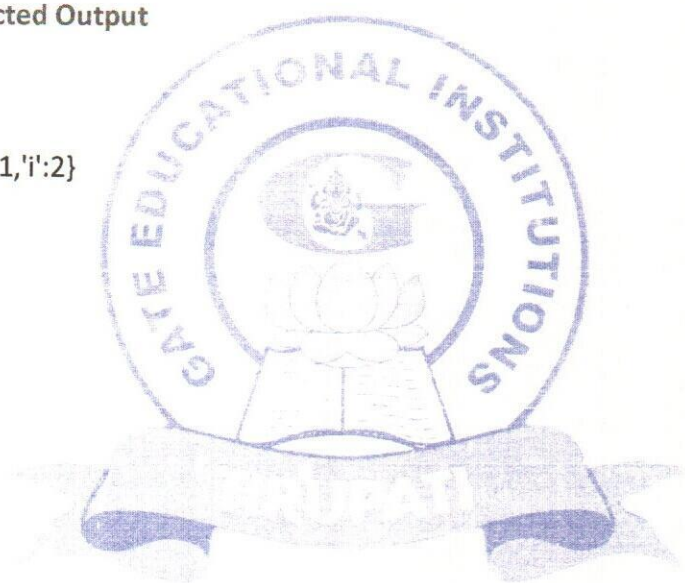
pseudocode :-

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

```
def char_frequency(s):
    freq = {}
    for ch in s:
        freq[ch] = freq.get(ch, 0) + 1
    return freq

print(char_frequency('nandini'))
```

Sample Input & Expected Output

Input: 'nandini'
Output: {'n':3,'a':1,'d':1,'i':2}



Generate a pascal's triangle

3. Write a python program to generate pascal's triangle.

Aim :-

Write a python program to generate pascal's triangle.

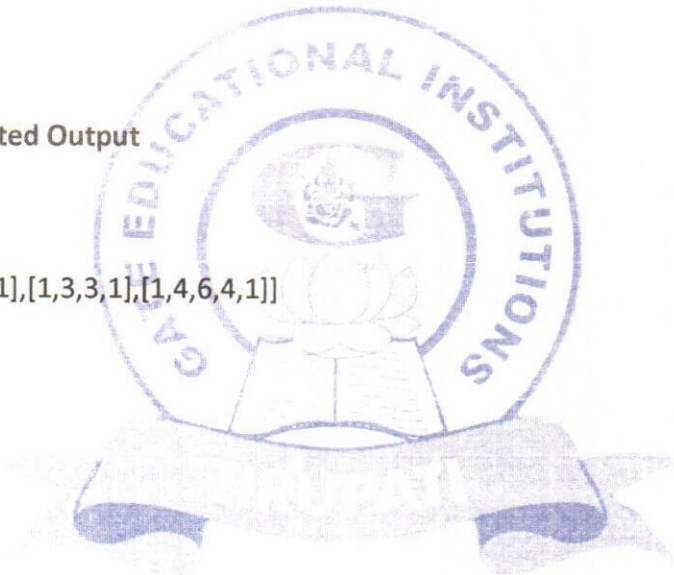
Algorithm :-

1. Start
2. Input rows
3. Generate triangle iterativesy
4. print rows
5. End.

pseudo code :-

```
BEGIN
INPUT rows
SET Triangle
for i in rows
COMPUTE Values
PRINT
END
```

```
def pascal_triangle(n):
    triangle = []
    for i in range(n):
        row = [1]*(i+1)
        for j in range(1,i):
            row[j] = triangle[i-1][j-1] + triangle[i-1][j]
        triangle.append(row)
    return triangle

print(pascal_triangle(5))
```



Sample Input & Expected Output

Input: 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]

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

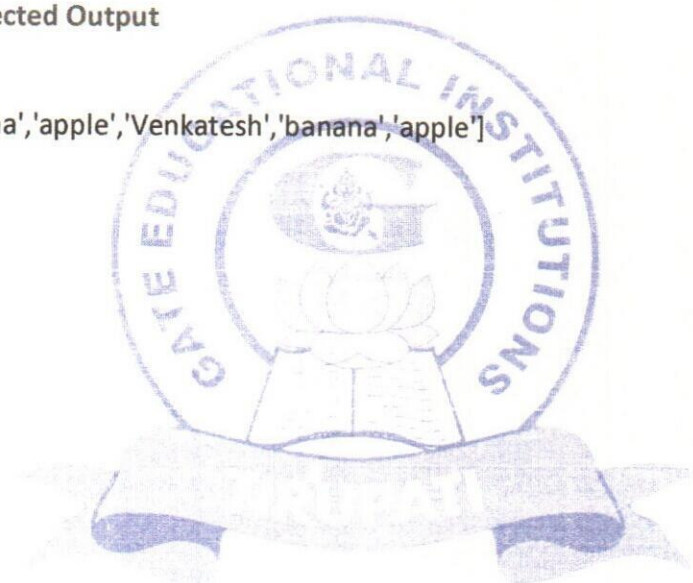
pseudocode :-

```
BEGIN
INPUT list
COUNT frequency
FIND max
PRINT result
END
```

```
def most_frequent_word(words):
    freq = {}
    for w in words:
        freq[w] = freq.get(w, 0) + 1
    return max(freq, key=freq.get)

print(most_frequent_word(['apple','banana','apple','Venkatesh','banana','apple']))
```

Sample Input & Expected Output

Input: ['apple','banana','apple','Venkatesh','banana','apple']
Output: 'apple'



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.

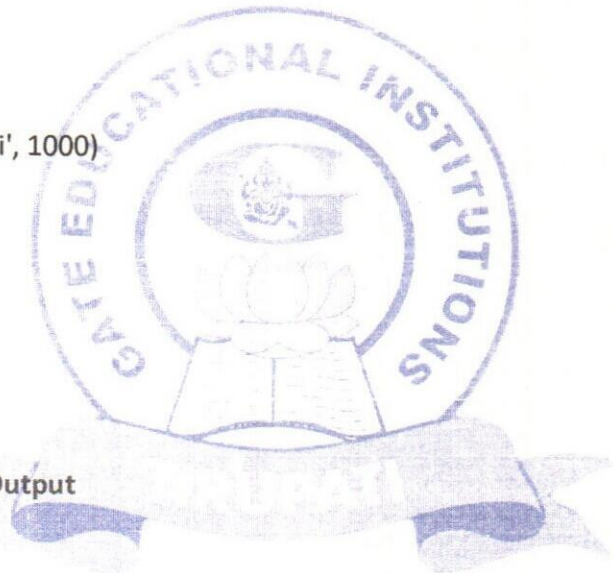
Algorithm :-

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
```

```
class BankAccount:
    def __init__(self, name, balance=0):
        self.name = name
        self.balance = balance
    def deposit(self, amount):
        self.balance += amount
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            raise ValueError('Insufficient funds')
    def get_balance(self):
        return self.balance

acc = BankAccount('Srawani', 1000)
acc.deposit(500)
acc.withdraw(300)
print(acc.get_balance())
```



Sample Input & Expected Output

Input: Deposit 200, Withdraw 100
Output: 1100

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 :-

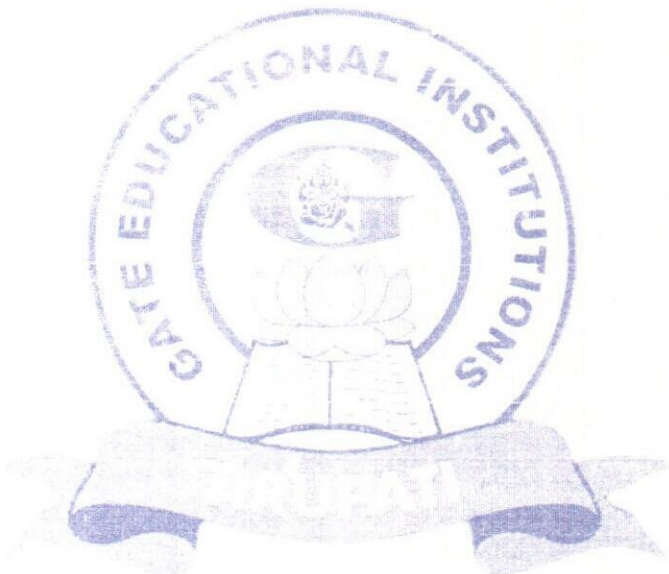
step - 1 :- Takes input

step - 2 :- initializes factorial

step - 3 :- Handles negative numbers.

step - 4 :- Uses for a loop to calculate factorial

step - 5 :- prints the result.



```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
   print(" Factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
      factorial = factorial*i
   print ("The factorial of", num,"is", factorial)
```

Sample Input & Expected Output

Enter a number: -1

Factorial does not exist for negative numbers

Enter a number: 9

The factorial of 9 is 362880

7. Write a python program to find the largest element among three numbers.

Aim :-

Write a python program to find the largest element among three numbers.

Algorithm :-

step1:- Step the program.

step-2 :- Read three numbers num 1, num 2 and num 3 from the user.

step-3 :- compare the numbers to find the largest.

* if num 1 > num 2 and num 1 > num 3, then largest = num 1.
* Else if num 2 > num 3, then largest = num 2
* Else, largest = num 3.

step-4 :- print "The largest number is largest".

step-5 :- End the program.

pseudo code :—

START

1. INPUT num 1 , num 2 , num 3 // Read three numbers from the user.

2. IF num 1 > num 2 AND num 1 > num 3 THEN

   largest = num 1.

3. Else IF num 2 > num 3 THEN

ELSE

   largest = num 3

END IF

4. PRINT "The largest number is", largest

END

```
num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num3 = float(input("Enter the third number: "))

# Step 2: Compare the numbers to find the largest

if num1 > num2 and num1 > num3:

    largest = num1

elif num2 > num3:

    largest = num2

else:

    largest = num3

    print(f"The largest number is {largest}")
```

Sample Input & Expected Output

Output:

Enter the first number: 85

Enter the second number: 74

Enter the third number: 112

The largest number is 112.0

To Display an prime numbers within an interval | Practical No. : 8

8. Write a python program to display an prime numbers within an interval.

Aim :-

To write a program to display an 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 it 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
  → prime print.

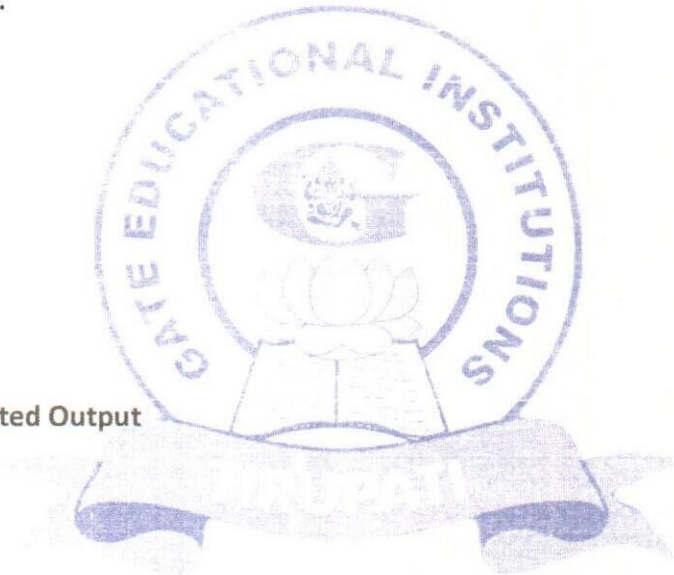
step-5 :- Repeat for all numbers in the range.

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 is prime = = True THEN
      PRINT num
   END IF
 END IF
END FOR
END
```

```
lower = int(input("enter lower range:"))
upper = int(input("enter upper range:"))
print("Prime numbers between", lower, "and", upper, "are:")
for num in range(lower, upper + 1):
  # all prime numbers are greater than 1
  if num > 1:
    for i in range(2, num):
      if (num % i) == 0:
        break
    else:
      print(num)
```



Sample Input & Expected Output

```
enter lower range:26
enter upper range:42
Prime numbers between 26 and 42 are:
29
31
37
41
```

To calculate LCM of two numbers in python

Practical No. : 9

9. Write a python 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, num 1 and num 2.

step-3 :- find the greater number between num 1 and num 2 and assign it to lcm.

step-4 :- Repeat the following until LCM is found :

* If lcm is divisible by both num 1 and num 2 (lcm % num 1 == 0 and lcm % num 2 == 0), then lcm is found → go to step-5.
* Else, increment lcm by 1 and check again.

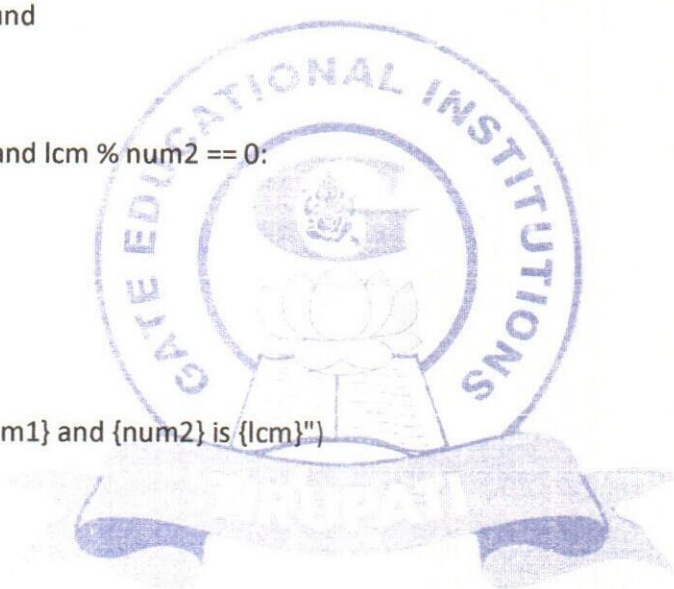
step-5 :- print " the LCM of num 1 and num 2 is lcm ".

step-6 :- End the program.

pseudocode :-

```
START
   INPUT num 1 // first number
   INPUT num 2 // second number
// step-1 :- find the greater number to start checking
IF num 1 > num 2 THEN
    lcm = num 1
ELSE
    lcm = num 2
END IF
// step-2 :- find LCM
WHILE True DO
   IF lcm MOD num 1 == 0 AND lcm MOD
num 2==0 THEN
    BREAK // LCM found
ELSE
   lcm = lcm + 1 // increment lcm and check again.
END IF
END WHILE
PRINT "LCM of ", num 1, "and ", num 2, "is ", lcm
END
```

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
# Find the greater number
if num1 > num2:
   lcm = num1
else:
   lcm = num2
# Loop until LCM is found
while True:
   if lcm % num1 == 0 and lcm % num2 == 0:
      break
   lcm += 1
# Print the result
print(f"The LCM of {num1} and {num2} is {lcm}")
```



Sample Input & Expected Output

Enter first number: 35

Enter second number: 55

The LCM of 35 and 55 is 385

To print the fibonacci sequence of a given number

Practical No. : 10

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 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.

pseudo code :-

```
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 fibanoccai numbers
FOR i from 0 To n-1 DO
    PRINT a
    next_term = a+b
    a = b
    b = next_term
  END FOR
END
```

```
n = int(input("Enter the number of terms in the Fibonacci sequence: "))

# Initialize the first two Fibonacci numbers
a, b = 0, 1

print("Fibonacci sequence:")

# Loop to print the first n terms
for i in range(n):
    print(a, end=" ")
    # Calculate the next term
    next_term = a + b
    a = b
    b = next_term
```

Sample Input & Expected Output

Enter the number of terms in the Fibonacci sequence: 18

Fibonacci sequence:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597

To print the Sum of natural numbers of a given number

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 :—

step-1 :- 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 (Value Error), display "Invalid input - enter an integer." and stops.

step-5 :- If $n < 0$, display "Enter a non-negative integer." and stop

step-6 :- set sum = 0

step-7 :- Set i = 1

step-8 :- while $i <= n$, do :

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

step-9 :- 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
```

```
def sum_natural_numbers(num):
    # Using the formula: sum = n(n+1)/2
    return num * (num + 1) // 2
def main():
    try:
        # Ask the user for input
        num = int(input("Enter a positive integer: "))
            if num <= 0:
            print("Please enter a number greater than 0.")
        else:
            total = sum_natural_numbers(num)
            print(f"Sum of natural numbers up to {num} is {total}")
        except ValueError:
        print("Invalid input. Please enter a valid integer.")
# Run the program
if __name__ == "__main__":
    main()
```

Sample Input & Expected Output

Enter a positive integer: 15

Sum of natural numbers up to 15 is 120

Basic Mathematical operations

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
    4. Multiply
    6. Divide
    8. Exit.
2. Repeat until the user chooses to exit (option 5):

0. Input the user's choice of operation.
1. If choice is 5, exit the program.
2. If choice is one of 1, 2, 3, 4 :
    1. Input the first number (num 1)
    2. Input the second number (num 2)
    3. check if input is numeric ; if not, display error and repeat.
    4. perform the selected operation :

- 1. Add → num 1 + num 2
- 2. Subtract → num 1 - num 2
- 3. Multiply → num 1 * num 2
- 4. Divide → check if num ! = 0

then num 1 / num 2 ; else display "Division by zero error"

5. Display the result.

3. Else (choice not 1-5), display error message "Invalid choice".

4. Ask the User if they want to perform another calculation.

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

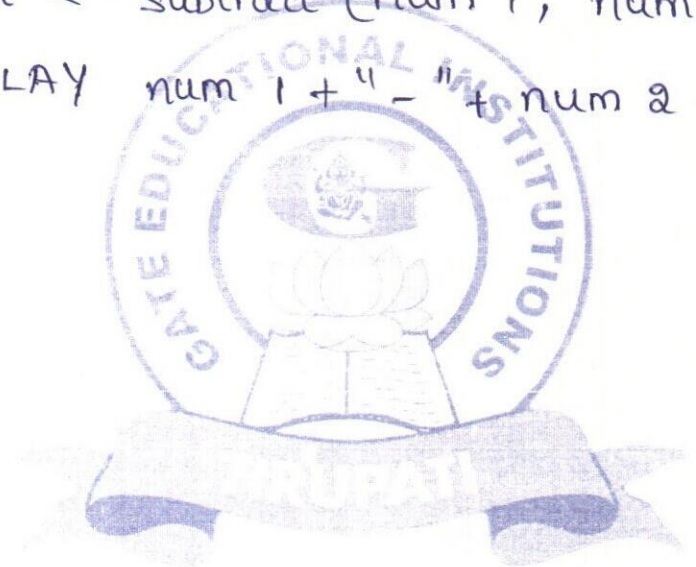
Stop.

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
     It choice = 5 THEN
        DISPLAY "Exiting program. Thank you !"
        BREAK
     ENDIF
     IF choice IN (1, 2, 3, 4) THEN
        READ num 1
```

```
    READ num 2
IF choice = 1 THEN
    result ← add (num 1, num 2)
    DISPLAY num 1 + " + " + num 2 + " = " +
                                        result
ELSEIF choice = 2 THEN
    result ← subtract (num 1, num 2)
    DISPLAY num 1 + " - " + num 2 + " =
```



```
# Python Calculator Program

# Function to add two numbers
def add(x, y):
    return x + y

# Function to subtract two numbers
def subtract(x, y):
    return x - y

# Function to multiply two numbers
def multiply(x, y):
    return x * y

# Function to divide two numbers
def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    return x / y

# Display menu
def display_menu():
    print("\nSelect operation:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")
    print("5. Exit")

# Main program loop
```

```
while True:
    display_menu()
    choice = input("Enter choice (1/2/3/4/5): ")
    if choice == '5':
        print("Exiting the calculator. Thank you!")
        break
    if choice in ('1', '2', '3', '4'):
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
        except ValueError:
            print("Invalid input! Please enter numeric values.")
            continue
        if choice == '1':
            print(f"{num1} + {num2} = {add(num1, num2)}")
        elif choice == '2':
            print(f"{num1} - {num2} = {subtract(num1, num2)}")
        elif choice == '3':
            print(f"{num1} * {num2} = {multiply(num1, num2)}")
        elif choice == '4':
            print(f"{num1} / {num2} = {divide(num1, num2)}")
    else:
        print("Invalid choice! Please select a valid option.")
```

Sample Input & Expected Output

Select operation:

1. Add

2. Subtract

3. Multiply

4. Divide

5. Exit

Enter choice (1/2/3/4/5): 1

Enter first number: 5

Enter second number: 6

5.0 + 6.0 = 11.0

Select operation:

1. Add

2. Subtract

3. Multiply

4. Divide

5. Exit

Enter choice (1/2/3/4/5): 2

Enter first number: 6

Enter second number: 7

6.0 - 7.0 = -1.0

Select operation:

1. Add

2. Subtract

3. Multiply

4. Divide

5. Exit

Enter choice (1/2/3/4/5): 3

Enter first number: 56

Enter second number: 87

56.0 * 87.0 = 4872.0

Select operation:

1. Add

2. Subtract

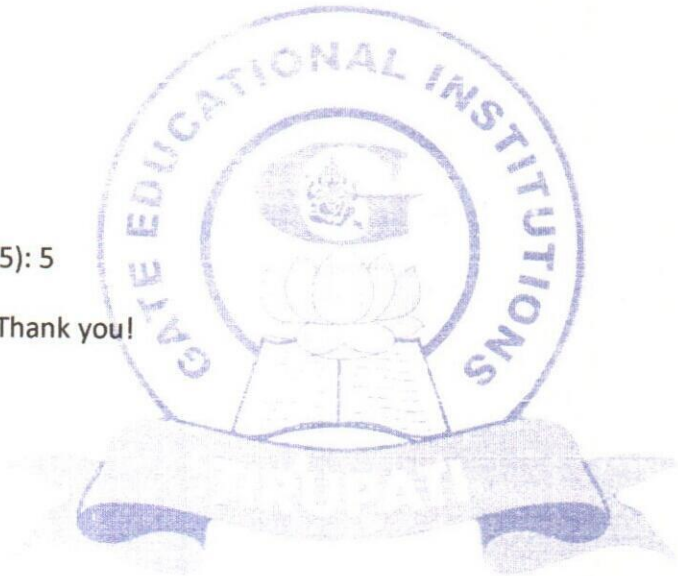
3. Multiply

4. Divide

5. Exit

Enter choice (1/2/3/4/5): 5

Exiting the calculator. Thank you!



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 :-

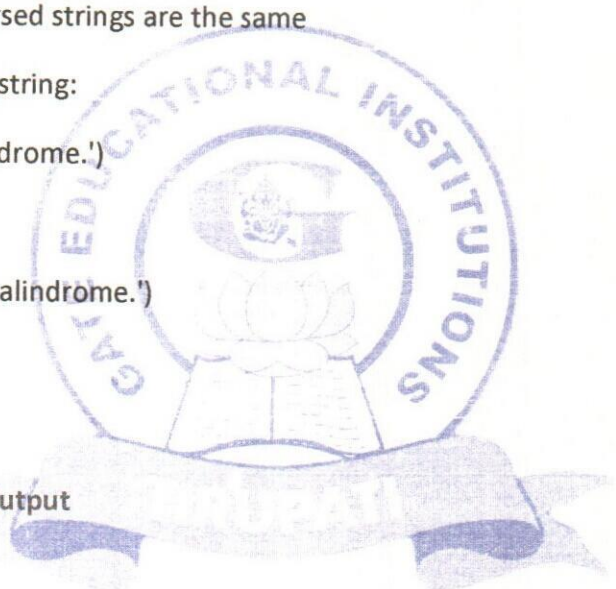
Start

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 :
   - It 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)
# Reversed 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
```

```
# Program to check if a string is a palindrome

# Read string input from user

string = input("Enter a string: ")

# Remove spaces and convert to lowercase for uniformity

clean_string = string.replace(" ", "").lower()

# Reverse the string

reversed_string = clean_string[::-1]

# Check if original and reversed strings are the same

if clean_string == reversed_string:

    print(f'"{string}" is a palindrome.')

else:

    print(f'"{string}" is not a palindrome.')
```



Sample Input & Expected Output

Enter a string: madam

"madam" is a palindrome.

Enter a string: lalitha

"lalitha" is not a palindrome.

14. Write a python program to add two matrices using nested loops read matrix values from keyboard.

Aim :—

To write a python program to add two matrices using nested loops read matrix values from keyboard.

Algorithm :—

start

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

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 matrix 1.

Input elements of Second matrix

4. Repeat the Same steps as above to input elements into matrix 2

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.

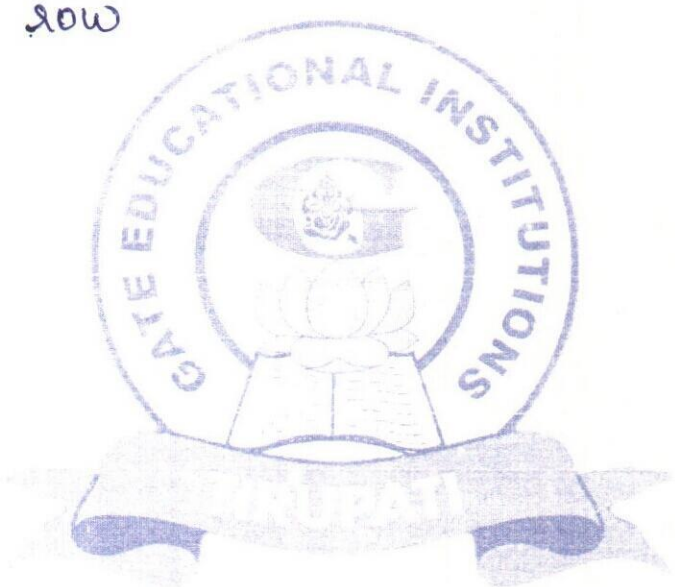
stop.

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.
```



```
# Program to add two matrices using nested loops

# Read number of rows and columns

rows = int(input("Enter the number of rows: "))

cols = int(input("Enter the number of columns: "))

# Initialize the matrices

matrix1 = []

matrix2 = []

# Input elements of the first matrix

print("Enter elements of the first matrix:")

for i in range(rows):

    row = []

    for j in range(cols):

        num = int(input(f"Element [{i+1},{j+1}]: "))

        row.append(num)

    matrix1.append(row)

# Input elements of the second matrix

print("Enter elements of the second matrix:")

for i in range(rows):

    row = []

    for j in range(cols):

        num = int(input(f"Element [{i+1},{j+1}]: "))

        row.append(num)

    matrix2.append(row)

# Add the two matrices

result = []
```

```
for i in range(rows):
    row = []
    for j in range(cols):
        row.append(matrix1[i][j] + matrix2[i][j])
    result.append(row)
# Display the result
print("\nSum of the two matrices:")
for row in result:
    print(row)
```

Enter the number of rows: 2

Enter the number of columns: 2

Enter elements of the first matrix:

Element [1,1]: 1

Element [1,2]: 3

Element [2,1]: 5

Element [2,2]: 6

Enter elements of the second matrix:

Element [1,1]: 5

Element [1,2]: 6

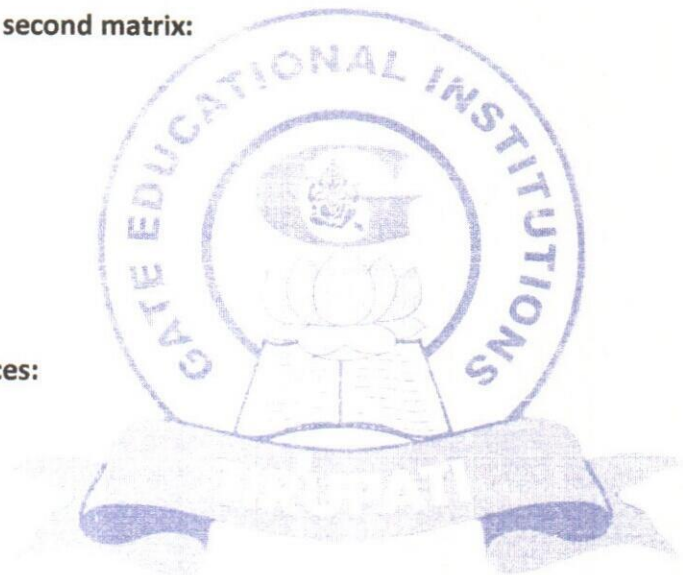
Element [2,1]: 7

Element [2,2]: 0

Sum of the two matrices:

[6, 9]

[12, 6]



To find all Armstrong numbers for given interval | Practical No. : 15

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 :".

check Each number in interval.

1. calculate the number of digits in num → num_digits
2. Initialize Sum_of_powers = 0
3. Set a temporary Vaiable 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 :-

```
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 no. 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
     DISPLAY num
    ENDIF
  END FOR
STOP
```

```
# Program to display all Armstrong numbers in a given interval

# Input the interval

lower = int(input("Enter the lower bound: "))

upper = int(input("Enter the upper bound: "))

print(f"Armstrong numbers between {lower} and {upper} are:")

# Loop through all numbers in the interval

for num in range(lower, upper + 1):

    num_digits = len(str(num))

    sum_of_powers = 0

    temp = num

    # Calculate sum of digits raised to the power of num_digits

    while temp > 0:

        digit = temp % 10

        sum_of_powers += digit ** num_digits

        temp = temp // 10

    # Check if sum equals the original number

    if sum_of_powers == num:

        print(num)
```

Sample Input & Expected Output

Practical No. :

Enter the lower bound: 10

Enter the upper bound: 500

Armstrong numbers between 10 and 500 are:

153

370

371

407