**LAB CYCLE - 3**

**Experiment No :1**

**Date :14/11/2024**

**Aim :**   
  
To find the factorial of a number.

**Pseudocode :**

PROMPT "Enter a number:" AND STORE input IN num

SET factorial TO 1

FOR i FROM 1 TO num DO

SET factorial TO factorial \* I

PRINT "Factorial of", num, "is", factorial

**Source Code :**

num=int(input("Enter a number:"))

f=1

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

    f=f\*i;

print(f"Factorial of {num} is {f}")

**Output :**

Enter a Number: 5

Factorial of 5 is 120

**Result :**The program is successfully executed and the output is verified.

**Experiment No :2**

**Date :14/11/2024**

**Aim :**

Generate Fibonacci series of N terms.

**Pseudocode :**

SET n1 to 0, n2 to 1

GET num from user

IF num > 0:

PRINT "Fibonacci series:"

IF num >= 1:

PRINT n1

IF num >= 2:

PRINT n2

FOR i FROM 3 TO num:

SET n3 to n1 + n2

SET n1 to n2

SET n2 to n3

PRINT n3

**Source Code :**

n1,n2 =0,1

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

if num>0:

    print("Fibonacci series:")

    if num >= 1:

        print(n1)

    if num >= 2:

        print(n2)

    for i in range(3,num+1):

        n3 = n1 + n2

        n1 = n2

        n2 = n3

        print(n3)

**Output :**

Enter the limit: 5

Fibonacci series:  
0

1

1

2

3

**Result :** The program is successfully executed and the output is verified.

**Experiment No :3**

**Date :14/11/2024**

**Aim :**

To find the sum of all items in a list.

**Pseudocode :**

GET num from user, split by comma

SET sum to 0

FOR each i in num:

ADD int(i) to sum

PRINT "Sum is:", sum

**Source Code :**

num=input("Enter numbers separated by comma:").split(",")

sum=0

for i in num:

    sum=sum+int(i)

print(f"Sum is:{sum}")

**Output :**

Enter the numbers separated by comma:1,2,3,4

sum is:10

**Result :** The program is successfully executed and the output is verified.

**Experiment No :4**

**Date :14/11/2024**

**Aim :**

Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.  
  
**Pseudocode :**

DEFINE function EVEN\_PERFECT\_SQUARES(start, end):

CREATE empty list results

SET start\_root to square root of start (integer part)

SET end\_root to square root of end (integer part)

FOR root FROM start\_root TO end\_root:

SET num to root \* root

IF num is between 1000 and 9999 AND num is even:

SET digits to string of num

IF all digits of num are even:

ADD num to results

RETURN results

GET start\_range from user

GET end\_range from user

SET result to EVEN\_PERFECT\_SQUARES(start\_range, end\_range)

IF result is not empty:

PRINT "Even perfect squares:", result

ELSE:

PRINT "No perfect squares found in the specified range."

**Source Code :**

def even\_perfect\_squares(start, end):

    results = []

    start\_root = int(start\*\*0.5)

    end\_root = int(end\*\*0.5)

    for root in range(start\_root, end\_root + 1):

        num = root \* root

        if 1000 <= num <= 9999 and num % 2 == 0:

            digits = str(num)

            if all(int(digit) % 2 == 0 for digit in digits):

                results.append(num)

    return results

start\_range = int(input("Enter start range: "))

end\_range = int(input("Enter end range: "))

result = even\_perfect\_squares(start\_range, end\_range)

if result:

    print(f"Even perfect squares: {result}")

else:

    print("No perfect squares found in the specified range.")

**Output :**

Enter the start range: 1000  
Enter the end range: 9999  
Even perfect squares: [4624, 6084, 6400, 8464]

**Result :** The program is successfully executed and the output is verified.

**Experiment No :5**

**Date :14/11/2024**

**Aim :**

Write a program using a for loop to print the multiplication table of n, where n is entered by the user.

**Pseudocode :**

GET n from user

PRINT "Multiplication table of", n

FOR i FROM 1 TO 10:

PRINT n, "x", i, "=", n \* i

**Source Code :**

n=int(input("Enter the number for multiplication table:"))

print(f"multiplication table of {n} is :")

for i in range (1,11):

        print(f"{n} x {i}= {n\*i}")

**Output :**

Enter the number for multiplication table:2

multiplication table of 2 is:

2 x 1= 2

2 x 2= 4

2 x 3= 6

2 x 4= 8

2 x 5= 10

2 x 6 12

2 x 7= 14

2 x 8 16

2 x 9= 18

2 x 10 20

**Result :** The program is successfully executed and the output is verified.

**Experiment No :6**

**Date :14/11/2024**

**Aim :**   
  
Write a program to display alternate prime numbers till N (obtain N from the user).

**Pseudocode :**

DEFINE function PR(n):

SET count to 0

FOR i FROM 1 TO n:

IF n % i == 0:

INCREMENT count by 1

IF count > 2:

RETURN 0

ELSE:

RETURN 1

GET n from user

CREATE empty list ls

CREATE empty list pls

FOR lim FROM 2 TO n:

ADD lim to ls

FOR lim in ls:

IF PR(lim) == 1:

ADD lim to pls

PRINT every second element of pls

**Source Code :**

def pr(n):

        count=0

        for i in range (1,n+1):

                if n%i==0:

                        count=count+1

        if count>2:

                return 0

        else:

                return 1

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

ls=[]

pls=[]

for lim in range(2,n+1):

        ls.append(lim)

for lim in ls:

        if pr(lim)==1:

                pls.append(lim)

print(pls[::2])

**Output :**

Enter the limit:5

[2, 5]

**Result :** The Program is successfully executed and the output is verified.

**Experiment No :7**

**Date :14/11/2024**

**Aim :**   
  
Write a program to compute and display the sum of all integers that are divisible by 6 but not by 4, and that lie below a user-given upper limit.

**Pseudocode :**

GET n from user

SET sum to 0

FOR i FROM 1 TO n-1:

IF i is divisible by 6 AND not divisible by 4:

ADD i to sum

PRINT "sum is:", sum

**Source Code :**

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

sum=0

for i in range (1,n):

        if i%6==0 and i%4!=0:

                sum=sum+i

print(f"sum is: {sum}")

**Output :**

Enter the limit: 12

Sum is: 6

**Result :** The program is successfully executed and the output is verified.

**Experiment No :8**

**Date :14/11/2024**

**Aim :**   
  
Calculate the sum of the digits of each number within a specified range (from 1 to a user-defined upper limit). Print the sum only if it is prime.

**Pseudocode :**

IMPORT math library

DEFINE function SUM\_OF\_DIGITS(n):

RETURN sum of digits of n

DEFINE function IS\_PRIME(n):

IF n <= 1:

RETURN False

IF n == 2:

RETURN True

IF n is even:

RETURN False

FOR i FROM 3 TO square root of n:

IF n is divisible by i:

RETURN False

RETURN True

DEFINE function SUM\_DIGITS\_IN\_RANGE(upper\_limit):

FOR num FROM 1 TO upper\_limit:

SET digit\_sum to SUM\_OF\_DIGITS(num)

IF digit\_sum is prime:

PRINT "Sum of digits of", num, "is", digit\_sum, "which is prime."

GET limit from user

CALL SUM\_DIGITS\_IN\_RANGE(limit)

**Source Code :**

import math

def sum\_of\_digits(n):

    return sum(int(digit) for digit in str(n))

def is\_prime(n):

    if n <= 1:

        return False

    if n == 2:

        return True

    if n % 2 == 0:

        return False

    for i in range(3, int(math.sqrt(n)) + 1, 2):

        if n % i == 0:

            return False

    return True

def sum\_digits\_in\_range(upper\_limit):

    for num in range(1, upper\_limit + 1):

        digit\_sum = sum\_of\_digits(num)

        if is\_prime(digit\_sum):

            print(f"Sum of digits of {num} is {digit\_sum}, which is prime.")

limit = int(input("Enter an upper limit: "))

sum\_digits\_in\_range(limit)

**Output :**

Enter an upper limit: 5

Sum of digits of 2 is 2, which is prime.

Sum of digits of 3 is 3, which is prime.

Sum of digits of 5 is 5, which is prime.

**Result :** The program is successfully executed and the output is verified.

**Experiment No :9**

**Date :14/11/2024**

**Aim :**

A number is input through the keyboard. Write a program to determine if it’s palindromic.

**Pseudocode :**

GET n from user

IF n is equal to n reversed:

PRINT "Is palindrome"

ELSE:

PRINT "Is not palindrome"

**Source Code :**

n=input("Enter the number to be checked:")

if n==n[::-1]:

        print("Is palindrome")

else:

        print("Is not palindrome")

**Output :**

Enter the number to be checked: 121

121 Is palindrome

Enter the number to be checked: 122

122 Is not palindrome

**Result :** The program is successfully executed and the output is verified.

**Experiment No :10**

**Date :14/11/2024**

**Aim :**

Write a program to generate all factors of a number.

**Pseudocode :**

GET n from user

CREATE empty list fact

FOR i FROM 1 TO n:

IF n is divisible by i:

ADD i to fact

PRINT "Factors of", n, "is", fact

**Source Code :**

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

fact=[]

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

        if n%i==0:

                fact.append(i)

print(f"factors of {n} is {fact}")

**Output :**

Enter the number: 12

factors of 12 is [1, 2, 3, 4, 6, 12]

**Result :** The program is successfully executed and the output is verified.

**Experiment No :11**

**Date :14/11/2024**

**Aim :**

Write a program to find whether the given number is an Armstrong number or not.

**Pseudocode :**

GET number from user

SET original\_num to number

SET sum\_of\_cubes to 0

WHILE number > 0:

SET digit to number % 10

ADD digit^3 to sum\_of\_cubes

SET number to number // 10

IF sum\_of\_cubes is equal to original\_num:

PRINT original\_num, "is an Armstrong number."

ELSE:

PRINT original\_num, "is not an Armstrong number."

**Source Code :**

number=int(input("Enter a number: "))

original\_num=number

sum\_of\_cubes = 0

while number > 0:

    digit = number % 10

    sum\_of\_cubes += digit \*\* 3

    number //= 10

if sum\_of\_cubes == original\_num:

    print(f"{original\_num} is an Armstrong number.")

else:

    print(f"{original\_num} is not an Armstrong number.")

**Output :**

Enter a number: 153

153 is an Armstrong number.

Enter a number: 153

153 is an Armstrong number.

**Result :** The program is successfully executed and the output is verified.

**Experiment No :12**

**Date :14/11/2024**

**Aim :**

Display the given pyramid with the step number accepted from the user.

1   
2 4   
3 6 9  
4 8 12 16

**Pseudocode :**

GET n from user

FOR i FROM 1 TO n:

FOR j FROM 1 TO i:

PRINT i \* j, WITHOUT newline

PRINT a newline

**Source Code :**

n=int(input("Enter the number of steps for the pyramid: "))

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

    for j in range(1,i+1):

        print(i\*j, end=' ')

    print()

**Output :**

Enter the number of steps for the pyramid: 4

1   
2 4   
3 6 9  
4 8 12 16

**Result :** The program is successfully executed and the output is verified.

**Experiment No :13**

**Date :14/11/2024**

**Aim :**

Construct following pattern using nested loop   
  
\*  
\*  \*  
\*  \*  \*   
\*  \*  \*  \*

\*  \*  \*  \*  \*    
\*  \*  \*  \*     
\*  \*  \*       
\*  \*     
\*

**Pseudocode :**

FOR i FROM 1 TO 5:

FOR j FROM 1 TO i:

PRINT "\*", WITHOUT newline

PRINT a newline

FOR i FROM 4 DOWN TO 1:

FOR j FROM 1 TO i:

PRINT "\*", WITHOUT newline

PRINT a newline

**Source Code :**

for i in range(1, 6):

    for j in range(i):

    print(“\*”,end=” ”)

    print( )

for i in range(4,0,-1):

    for j in range(i):

    print(“\*”,end=” ”)

    print( )

**Output :**

\*  
\*  \*  
\*  \*  \*   
\*  \*  \*  \*

\*  \*  \*  \*  \*    
\*  \*  \*  \*     
\*  \*  \*       
\*  \*     
\*

**Result :** The program is successfully executed and the output is verified.