PROGRAM-1

AIM: To find the factorial of a number.

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
```

PROGRAM-2

AIM: Generate Fibonacci series of N terms.

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
```

PROGRAM-3

AIM: To find the sum of all items in a list.

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 numbers separated by comma:1,2,3,4
Sum is:10
```

PROGRAM-4

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

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

PROGRAM-5

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

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
```

PROGRAM-6

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

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]
```

PROGRAM-7

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.

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
```

PROGRAM-8

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.

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.
```

PROGRAM-9

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

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
Is palindrome
```

```
Enter the number to be checked:122
Is not palindrome
```

PROGRAM-10

AIM: Write a program to generate all factors of a number.

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]
```

PROGRAM-11

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

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: 152
152 is not an Armstrong number.
```

PROGRAM-12

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

```
1
2 4
3 6 9
4 8 12 16
```

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
```

PROGRAM-13

AIM: . Construct following pattern using nested loop

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

SOURCE CODE:

```
for i in range(1, 6):
    print('* ' * i)
for i in range(4, 0, -1):
    print('* ' * i)
```

OUTPUT:

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```