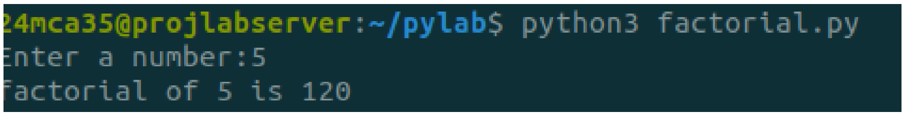
PGMM1
AIM:
Write a program to find the factorial of a number.

SOURCECODE:

```
number = int(input("Enter a number: "))
factorial = 1


if number < 0:
    print("Factorial is not defined for negative numbers.")
elif number == 0:
    print("The factorial of 0 is 1.")
else:
    for i in range(1, number + 1):
        factorial *= i
    print("The factorial of", number, "is", factorial)
```

OUTPUT:



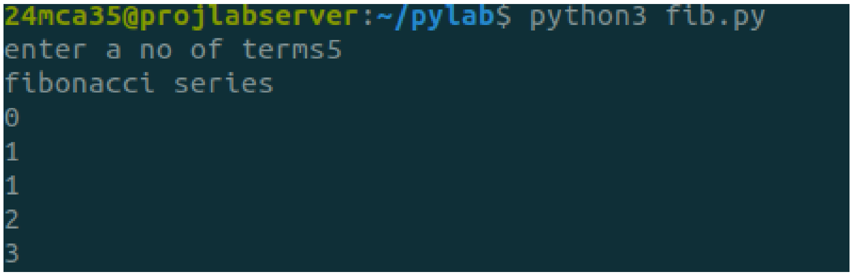
```
24mca35@projlabserver:~/pylab$ python3 factorial.py
Enter a number:5
factorial of 5 is 120
```

PGMM2
AIM:
Generate Fibonacci series of N terms.

SOURCECODE:

```
n=int(input("enter no of steps"))
for i in range (1,n+1):
        for j in range (1,i+1):
                print(i*j,end=" ")
        print()
```

OUTPUT:



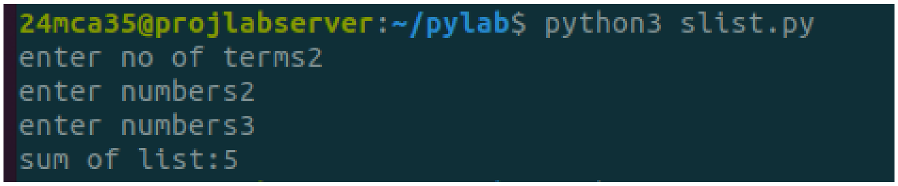
```
24mca35@projlabserver:~/pylab$ python3 fib.py
enter a no of terms5
fibonacci series
0
1
1
2
3
```

PGMM3
AIM:
Write a program to find the sum of all items in a list. [Using for loop]

SOURCECODE:

```
n=int(input("enter no of terms"))
sum=0
num=[]
for i in range (n):
        n1=int(input("enter numbers"))
        num.append(n1)
for i in  num:
        sum=sum+i
print(f"sum of list:{sum}")
```

OUTPUT:



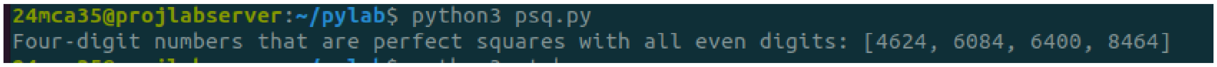
```
24mca35@projlabserver:~/pylab$ python3 slist.py
enter no of terms2
enter numbers2
enter numbers3
sum of list:5
```

PGMM4
AIM:
Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

SOURCECODE:

```
even_digit_squares = []

for num in range(32, 100):
    square = num * num
    if 1000 <= square <= 9999:
        square_str = str(square)
        if (square_str[0] in "02468" and
            square_str[1] in "02468" and
            square_str[2] in "02468" and
            square_str[3] in "02468"):
            even_digit_squares.append(square)

print("Four-digit numbers that are perfect squares with all even digits:", even_digit_squares)
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ python3 psq.py
Four-digit numbers that are perfect squares with all even digits: [4624, 6084, 6400, 8464]
```

PGMM5

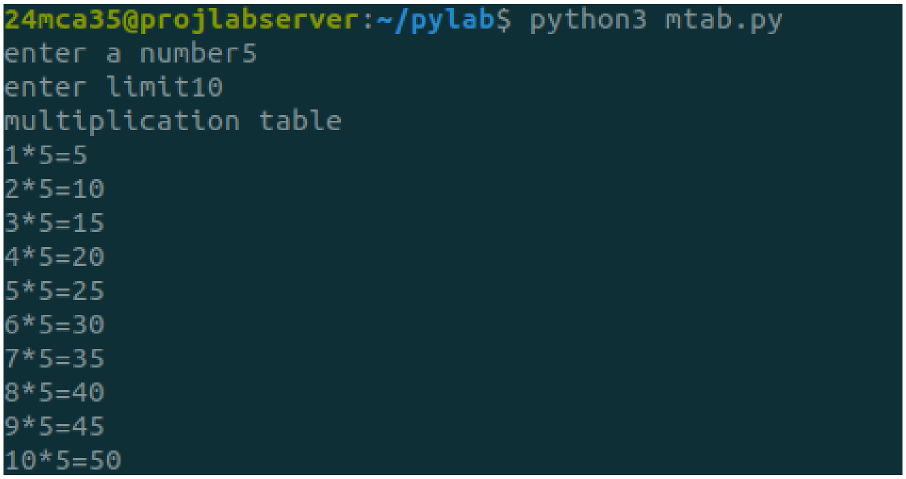
AIM:

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

SOURCECODE:

```
n=int(input("enter a number"))
m=int(input("enter limit"))
print("multiplication table")
for i in range(1,m+1):
        m1=i*n
        print(f"{i}*{n}={m1}")
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ python3 mtab.py
enter a number5
enter limit10
multiplication table
1*5=5
2*5=10
3*5=15
4*5=20
5*5=25
6*5=30
7*5=35
8*5=40
9*5=45
10*5=50
```

PGMM6

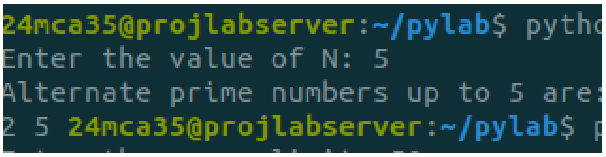
AIM:

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

SOURCECODE:

```
N = int(input("Enter the value of N: "))
count = 0  # Counter to keep track of alternate primes
print("Alternate prime numbers up to", N, "are:")

for num in range(2, N + 1):
    is_prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            is_prime = False
            break

    if is_prime:
        if count % 2 == 0:
            print(num, end=" ")
        count += 1
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ pytho
Enter the value of N: 5
Alternate prime numbers up to 5 are:
2 5 24mca35@projlabserver:~/pylab$ p
```

PGMM7

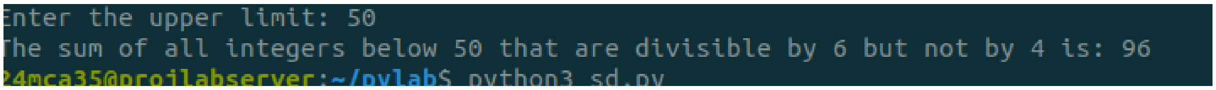
AIM:

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.

SOURCECODE:

```
  GNU nano 6.2                                                    ul.py
upper_limit = int(input("Enter the upper limit: "))
total_sum = 0


for num in range(1, upper_limit):
    if num % 6 == 0 and num % 4 != 0:
        total_sum += num

print("The sum of all integers below", upper_limit, "that are divisible by 6 but not by 4 is:",
total_sum)
```

OUTPUT:



```
Enter the upper limit: 50
The sum of all integers below 50 that are divisible by 6 but not by 4 is: 96
24mca35@projlabserver:~/pylab$ python3 sd.py
```

PGMM8

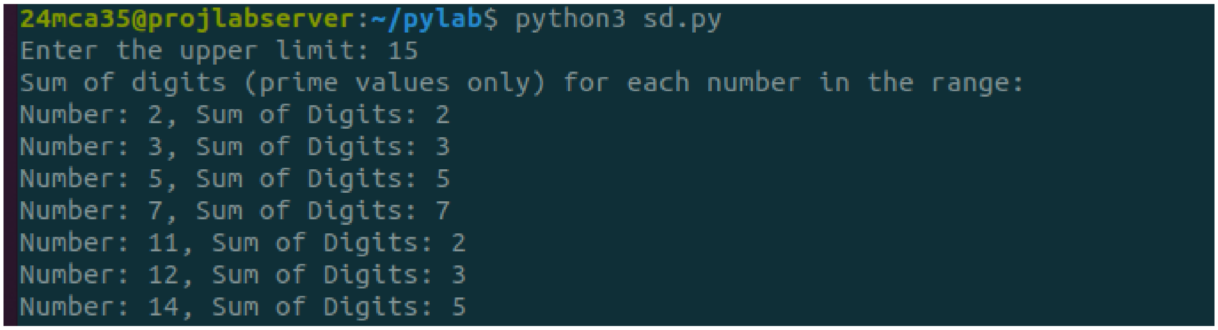
AIM:

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.

SOURCECODE:

```
upper_limit = int(input("Enter the upper limit: "))

print("Sum of digits (prime values only) for each number in the range:")

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

    digit_sum = 0
    temp = num
    while temp > 0:
        digit_sum += temp % 10
        temp //= 10

    if digit_sum <= 1:
        continue

    is_prime = True
    for i in range(2, int(digit_sum**0.5) + 1):
        if digit_sum % i == 0:
            is_prime = False
            break


    if is_prime:
        print(f"Number: {num}, Sum of Digits: {digit_sum}")
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ python3 sd.py
Enter the upper limit: 15
Sum of digits (prime values only) for each number in the range:
Number: 2, Sum of Digits: 2
Number: 3, Sum of Digits: 3
Number: 5, Sum of Digits: 5
Number: 7, Sum of Digits: 7
Number: 11, Sum of Digits: 2
Number: 12, Sum of Digits: 3
Number: 14, Sum of Digits: 5
```

PGMM9

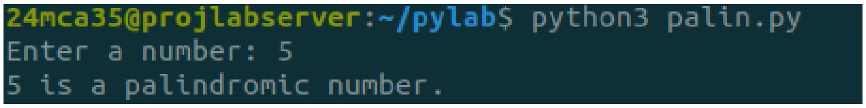
AIM:

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

SOURCECODE:

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

is_palindrome = True
length = len(number)


for i in range(length // 2):
    if number[i] != number[length - 1 - i]:
        is_palindrome = False
        break


if is_palindrome:
    print(f"{number} is a palindromic number.")
else:
    print(f"{number} is not a palindromic number.")
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ python3 palin.py
Enter a number: 5
5 is a palindromic number.
```

PGMM10

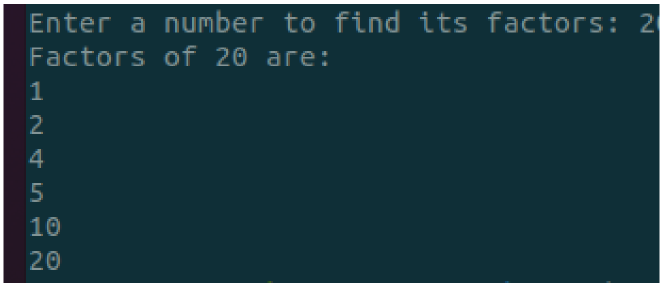
AIM:

Write a program to generate all factors of a number. [use while loop]

SOURCECODE:

```
number = int(input("Enter a number to find its factors: "))


factor = 1

print(f"Factors of {number} are:")


while factor <= number:
    if number % factor == 0:
        print(factor)
    factor += 1
```

OUTPUT:



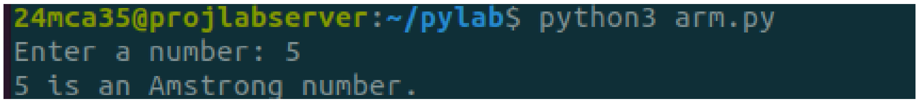
```
Enter a number to find its factors: 2
Factors of 20 are:
1
2
4
5
10
20
```

PGMM11
AIM:
Write a program to find whether the given number is an Armstrong number or not. [use while loop]

SOURCECODE:

```
number = int(input("Enter a number: "))
original_number = number
sum_of_powers = 0
num_digits = len(str(number))


while number > 0:
    digit = number % 10
    sum_of_powers += digit ** num_digits
    number //= 10

if original_number == sum_of_powers:
    print(f"{original_number} is an Amstrong number.")
else:
    print(f"{original_number} is not an Amstrong number.")
```

OUTPUT:



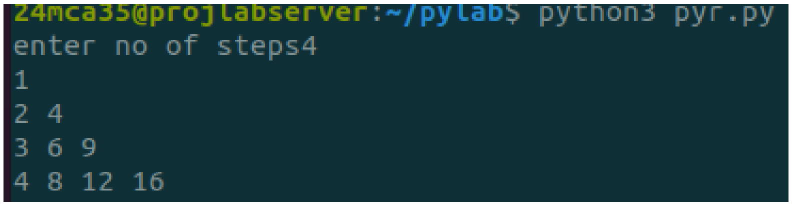
```
24mca35@projlabserver:~/pylab$ python3 arm.py
Enter a number: 5
5 is an Amstrong number.
```

PGMM12
AIM:
Display the given pyramid with the step number accepted from the user.

SOURCECODE:

```
n=int(input("enter no of steps"))
for i in range (1,n+1):
        for j in range (1,i+1):
                print(i*j,end=" ")
        print()
```

OUTPUT:



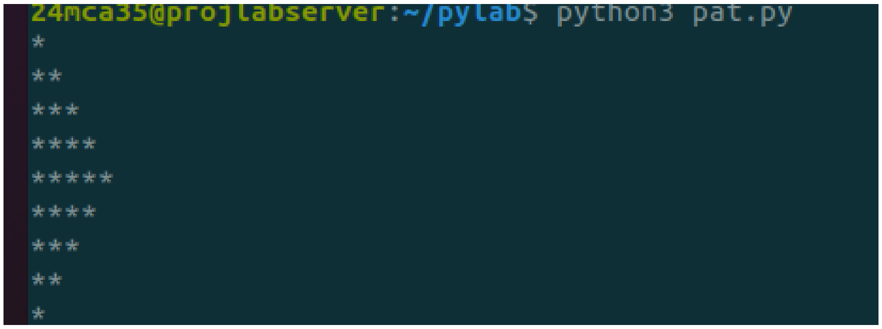
```
24mca35@projlabserver:~/pylab$ python3 pyr.py
enter no of steps4
1
2 4
3 6 9
4 8 12 16
```

PGMM13
AIM:
Construct a pattern using nested loop

SOURCECODE:

```
rows = 5
for i in range(1, rows + 1):
    for j in range(i):
        print("*", end=" ")
    print()
for i in range(rows - 1, 0, -1):
    for j in range(i):
        print("*", end=" ")
    print()
```

OUTPUT:



```
24mca35@projlabserver:~/pylab$ python3 pat.py
*
**
***
****
*****
****
***
**
*
```