**Applications Development Laboratory (ADL)**

**Individual Work**

**Lab. No: 1 , Assignment No.: 0.3**

**Topic: Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll Number:** | **2305573** | **Branch/Section:** | **CSE-35** |
| **Name in Capital:** | **SHUBHAM KUMAR** | | |

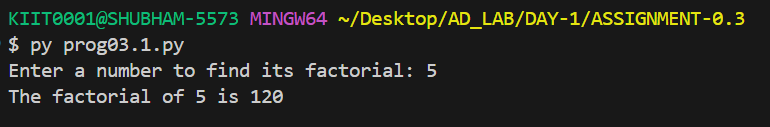
**Program No: 03.1**

**Program Title:**

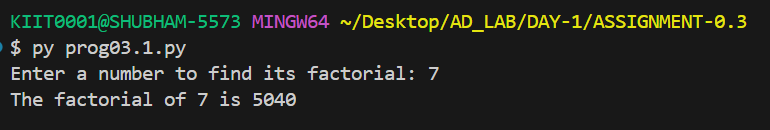
WAP to calculate the factorial of a given number.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

num=int(input("Enter a number to find its factorial: "))

result=factorial(num)

print(f"The factorial of {num} is {result}")

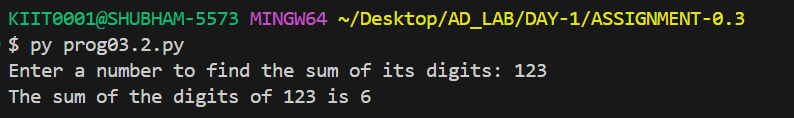
**Program No: 03.2**

**Program Title:**

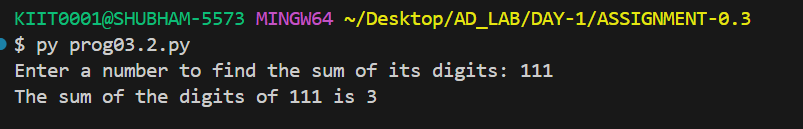
WAP to calculate the sum of digits of a given number.

**Input/Output Screenshots:**

**RUN-1:**

****

**RUN-2**

****

**Source code**

def sum\_of\_digits(n):

if n == 0:

return 0

else:

return n % 10 + sum\_of\_digits(n // 10)

num = int(input("Enter a number to find the sum of its digits: "))

result = sum\_of\_digits(num)

print(f"The sum of the digits of {num} is {result}")

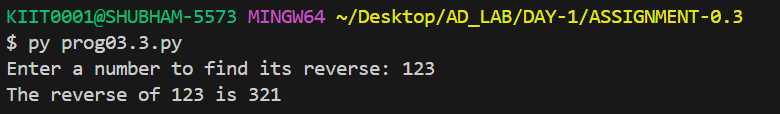
**Program No: 03.3**

**Program Title:**

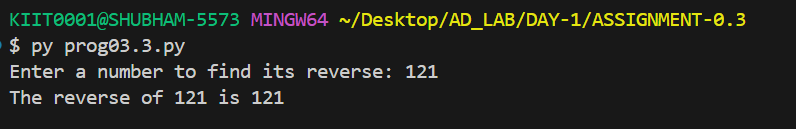
WAP to display the reverse of a number entered through keyboard.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def reverse\_number(n):

if n < 10:

return str(n)

else:

return str(n % 10) + reverse\_number(n // 10)

num = int(input("Enter a number to find its reverse: "))

result = reverse\_number(num)

print(f"The reverse of {num} is {result}")

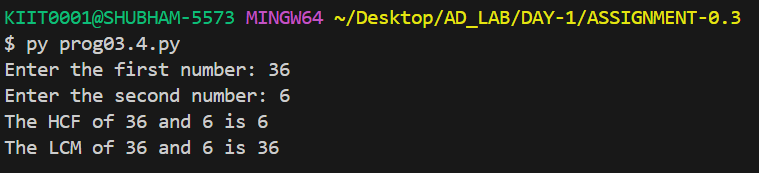
**Program No: 03.4**

**Program Title:**

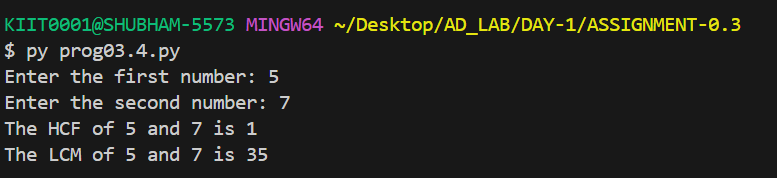
WAP to find the GCD/HCF of two numbers.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def hcf(a, b):

if b == 0:

return a

else:

return hcf(b, a % b)

def lcm(a, b):

return (a \* b) // hcf(a, b)

num1 = int(input("Enter the first number: "))

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

hcf\_result = hcf(num1, num2)

lcm\_result = lcm(num1, num2)

print(f"The HCF of {num1} and {num2} is {hcf\_result}")

print(f"The LCM of {num1} and {num2} is {lcm\_result}")

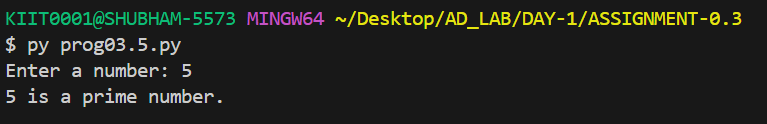
**Program No: 03.5**

**Program Title:**

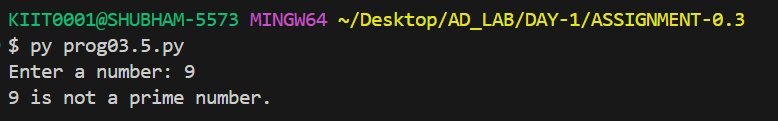
WAP to check whether a number n is prime number or not.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

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

if is\_prime(number):

print(f"{number} is a prime number.")

else:

print(f"{number} is not a prime number.")

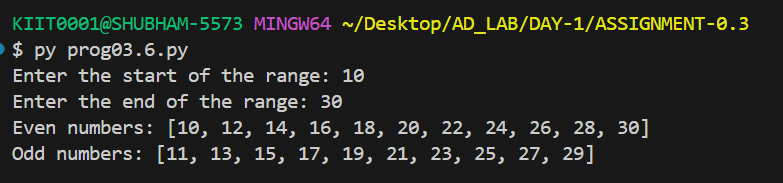
**Program No: 03.6**

**Program Title:**

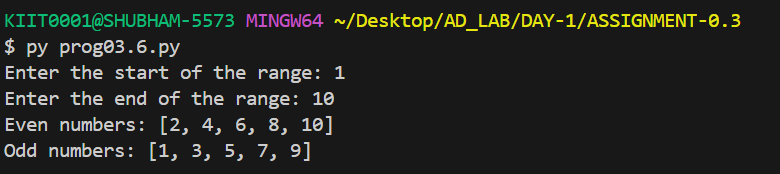
WAP to print all odd and even numbers separately within a given range. The range is input through user.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def print\_odd\_even\_numbers(start, end):

odd\_numbers = []

even\_numbers = []

for num in range(start, end + 1):

if num % 2 == 0:

even\_numbers.append(num)

else:

odd\_numbers.append(num)

print("Even numbers:", even\_numbers)

print("Odd numbers:", odd\_numbers)

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

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

print\_odd\_even\_numbers(start\_range, end\_range)

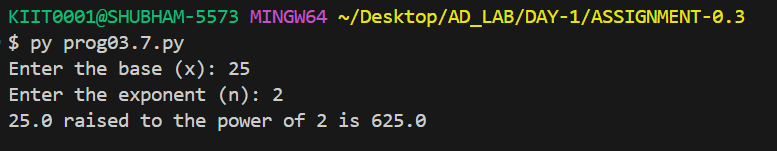
**Program No: 03.7**

**Program Title:**

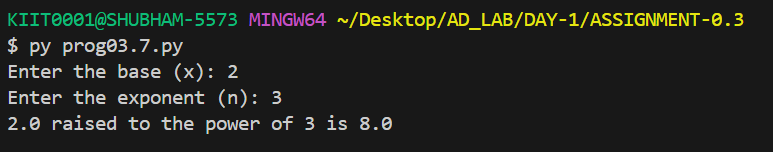
WAP to evaluate the equation y=x^n where n is a non-negative integer.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def power(x, n):

result = 1

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

result \*= x

return result

base = float(input("Enter the base (x): "))

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

print(f"{base} raised to the power of {exponent} is {power(base, exponent)}")

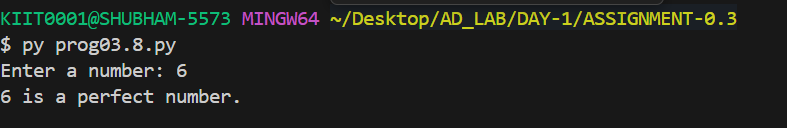
**Program No: 03.8**

**Program Title:**

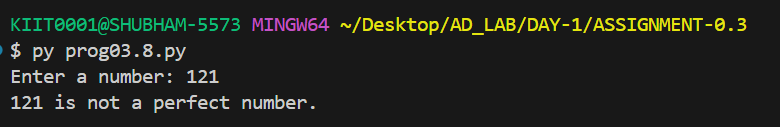
WAP to check whether an input integer is perfect number or not.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def is\_perfect\_number(n):

if n < 1:

return False

divisors\_sum = sum(i for i in range(1, n) if n % i == 0)

return divisors\_sum == n

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

if is\_perfect\_number(number):

print(f"{number} is a perfect number.")

else:

print(f"{number} is not a perfect number.")

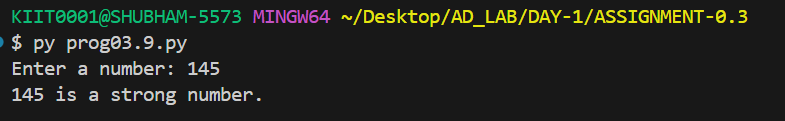
**Program No: 03.9**

**Program Title:**

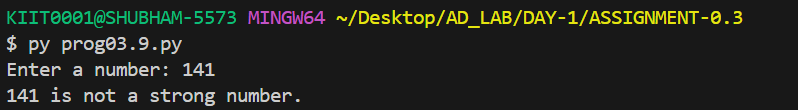
WAP to check whether an input integer is strong number or not.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def is\_strong\_number(num):

factorial = lambda x: 1 if x == 0 else x \* factorial(x - 1)

sum\_of\_factorials = sum(factorial(int(digit)) for digit in str(num))

return sum\_of\_factorials == num

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

if is\_strong\_number(number):

print(f"{number} is a strong number.")

else:

print(f"{number} is not a strong number.")

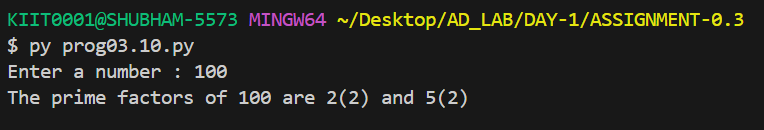
**Program No: 03.10**

**Program Title:**

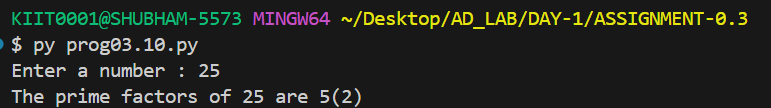
WAP to find out the prime factors of a number entered through keyboard (distinct).

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def prime\_factors(n):

factors = {}

divisor = 2

while divisor \* divisor <= n:

while n % divisor == 0:

factors[divisor] = factors.get(divisor, 0) + 1

n //= divisor

divisor += 1

if n > 1:

factors[n] = factors.get(n, 0) + 1

return factors

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

factors = prime\_factors(num)

output = " and ".join([f"{p}({count})" for p, count in factors.items()])

print(f"The prime factors of {num} are {output}")

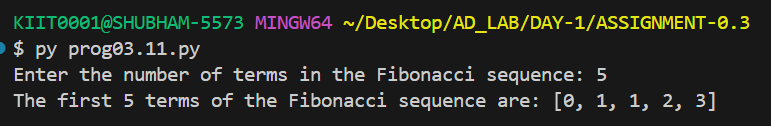
**Program No: 03.11**

**Program Title:**

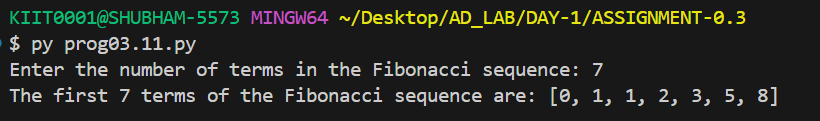
WAP to find the first n numbers of a Fibonacci sequence.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def fibonacci(n):

fib\_sequence = []

a, b = 0, 1

for \_ in range(n):

fib\_sequence.append(a)

a, b = b, a + b

return fib\_sequence

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

result = fibonacci(num)

print(f"The first {num} terms of the Fibonacci sequence are: {result}")

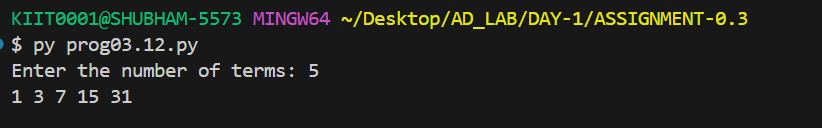
**Program No: 03.12**

**Program Title:**

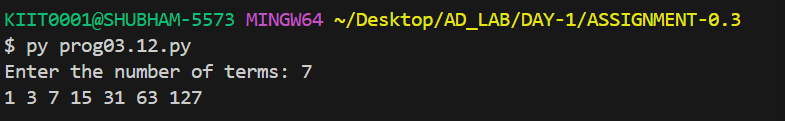
WAP to print the series as 1 3 7 15 31 ..........n, where n is given by user.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def print\_series(n):

term = 1

for i in range(n):

print(term, end=" ")

term = term \* 2 + 1

print()

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

print\_series(n)

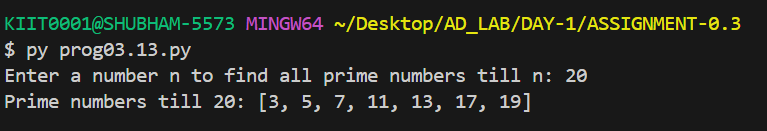
**Program No: 03.13**

**Program Title:**

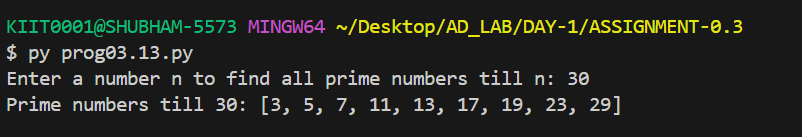
WAP to print the series as 3 5 7 11 13 17..........n, where n is given by user.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def primes\_till\_n(n):

primes = []

for num in range(3, n + 1):

if is\_prime(num):

primes.append(num)

return primes

n = int(input("Enter a number n to find all prime numbers till n: "))

prime\_numbers = primes\_till\_n(n)

print(f"Prime numbers till {n}: {prime\_numbers}")

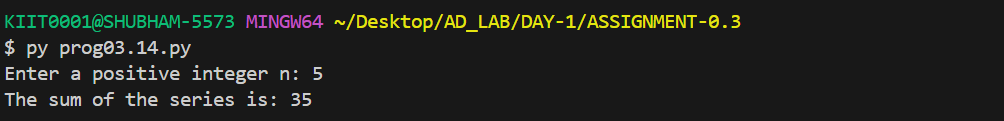
**Program No: 03.14**

**Program Title:**

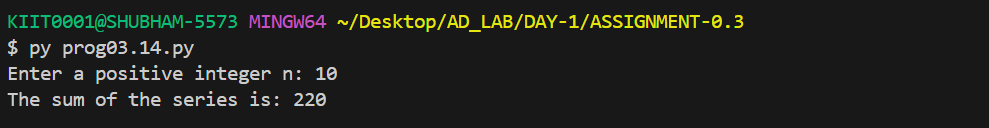
WAP to sum the following series S=1+(1+2)+(1+2+3)+...+(1+2+3+...+n).

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def sum\_of\_series(n):

total\_sum = 0

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

term\_sum = 0

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

term\_sum += j

total\_sum += term\_sum

return total\_sum

n = int(input("Enter a positive integer n: "))

result = sum\_of\_series(n)

print("The sum of the series is:", result)

**Program No: 03.15**

**Program Title:**

WAP to print the following pattern for n rows. Ex. for n=5 rows

\*

\* \*

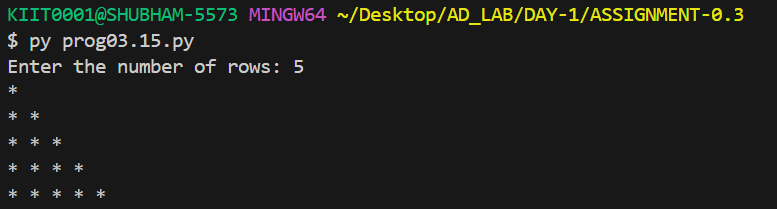
\* \* \*

\* \* \* \*

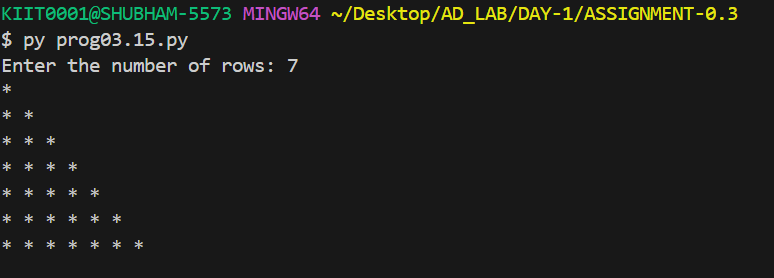
\* \* \* \* \*

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def print\_pattern(n):

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

for j in range(i):

print("\*", end=" ")

print()

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

print\_pattern(n)

**Program No: 03.16**

**Program Title:**

WAP to print the Following pattern for n rows. Ex. for n=5 rows

\*

\* \*

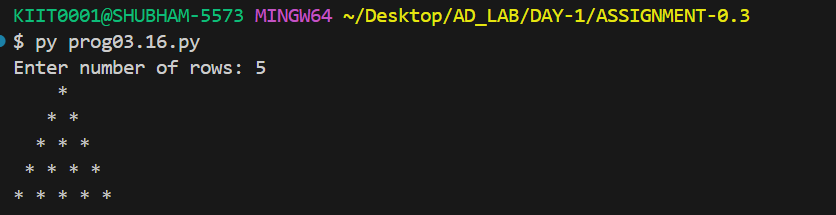
\* \* \*

\* \* \* \*

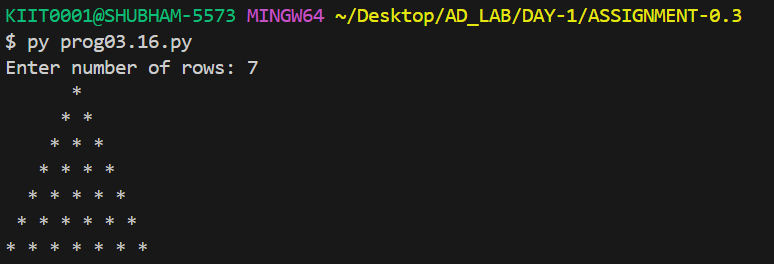
\* \* \* \* \*

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def print\_pattern(n):

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

print(" " \* (n - i), end="")

print("\* " \* i)

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

print\_pattern(rows)

**Program No: 03.17**

**Program Title:**

WAP to print the following pattern for n rows. Ex. for n=6 rows

1

0 1

1 0 1

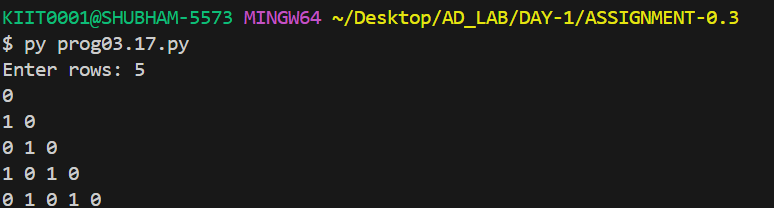
0 1 0 1

1 0 1 0 1

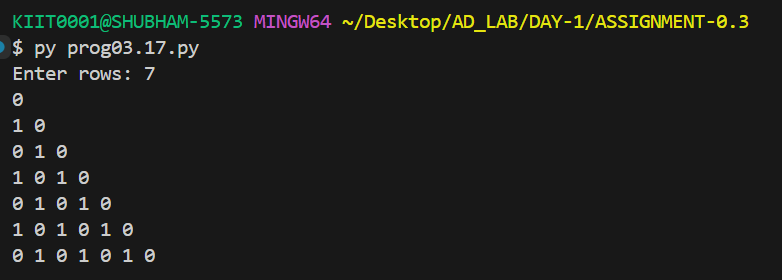
0 1 0 1 0 1

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def binary\_triangle(n):

for i in range(n):

for j in range(i + 1):

print((i + j) % 2, end=" ")

print()

rows = int(input("Enter rows: "))

binary\_triangle(rows)

**Program No: 03.18**

**Program Title:**

WAP to print the following pattern for n rows. Ex. for n=5 rows

A

B A

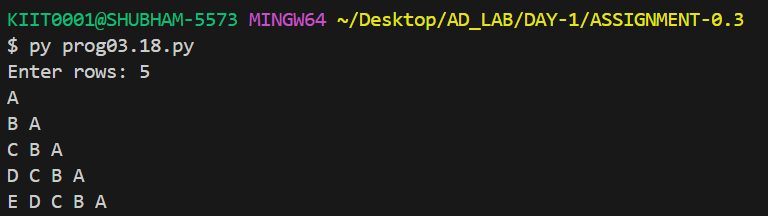
C B A

D C B A

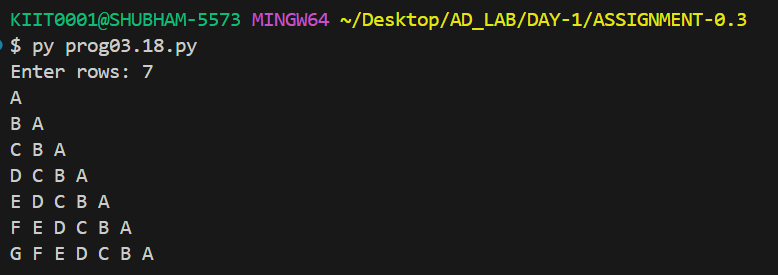
E D C B A

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def alpha\_pattern(n):

for i in range(n):

for j in range(i + 1):

print(chr(ord('A') + i - j), end=" ")

print()

rows = int(input("Enter rows: "))

alpha\_pattern(rows)

**Program No:03.19**

**Program Title:**

WAP to print the following pattern for n rows. Ex. for n=5 rows

1

2 1

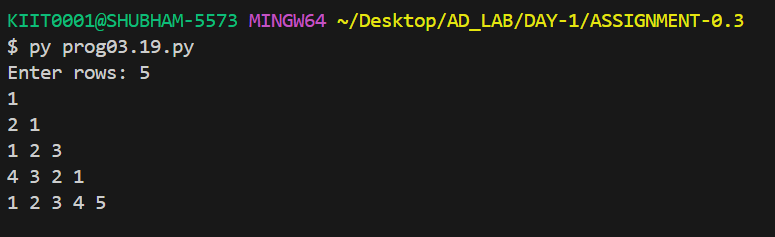
1 2 3

4 3 2 1

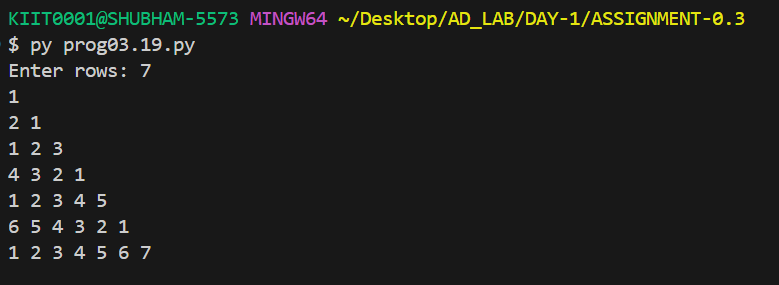
1 2 3 4 5

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def reverse\_pyramid(n):

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

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

print(j, end=" ")

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

print(j, end=" ")

print()

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

reverse\_pyramid(num)

**Program No: 03.20**

**Program Title:**

WAP to form reverse pyramid of numbers for a given number. Ex. for number 4

1 2 3 4 3 2 1

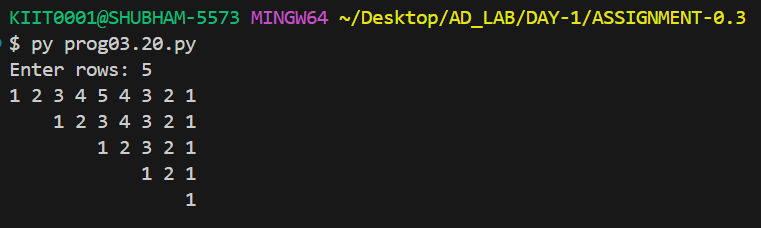
1 2 3 2 1

1 2 1

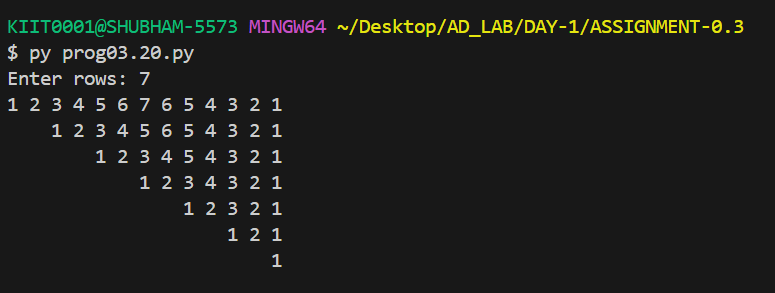
1

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def reverse\_pyramid(n):

max\_width = 2\*n - 1

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

count = 2\*i - 1

spaces = (max\_width - count) \* 2

print(" " \* spaces, end="")

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

print(j, end=" ")

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

print(j, end=" ")

print()

n = int(input("Enter rows: "))

reverse\_pyramid(n)

**Program No: 03.21**

**Program Title:**

WAP to generate the pascal triangle pyramid of numbers for a given number. Ex. for number 5

1

1 1

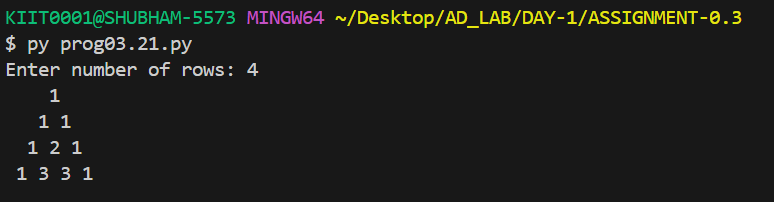
1 2 1

1 3 3 1

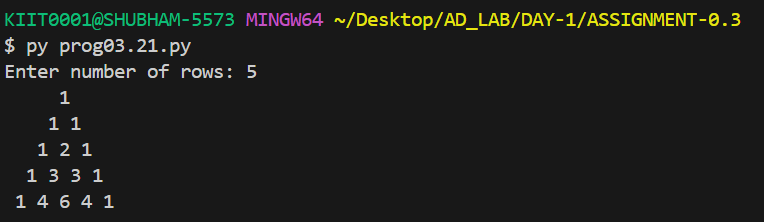
1 4 6 4 1

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def pascal\_triangle(n):

for i in range(n):

print(" " \* (n - i), end="")

val = 1

for j in range(i + 1):

print(val, end=" ")

val = val \* (i - j) // (j + 1)

print()

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

pascal\_triangle(rows)

**Program No: 03.22**

**Program Title:**

WAP to display the following style o/p for a given string input through keyboard.(Ex.for a string

“KIITCSIT”)

KIITCSITTISCTIIK

KIITCSI ISCTIIK

KIITCS SCTIIK

KIITC CTIIK

KIIT TIIK

KII IIK

KI IK

K K

KI IK

KII IIK

KIIT TIIK

KIITC CTIIK

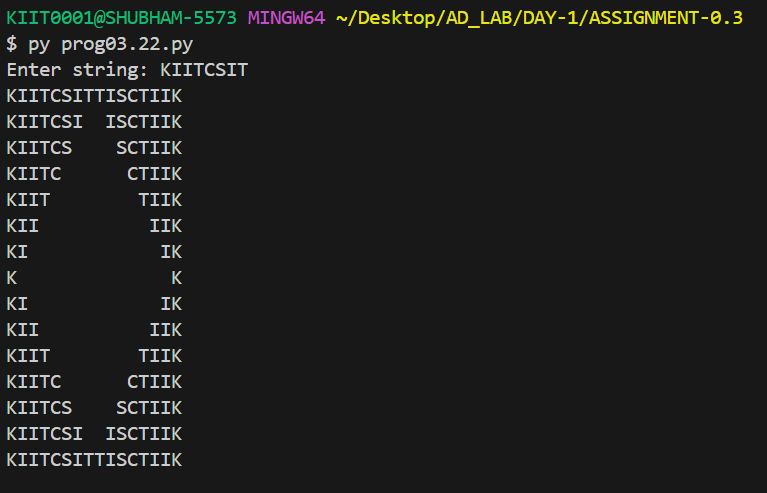
KIITCS SCTIIK

KIITCSI ISCTIIK

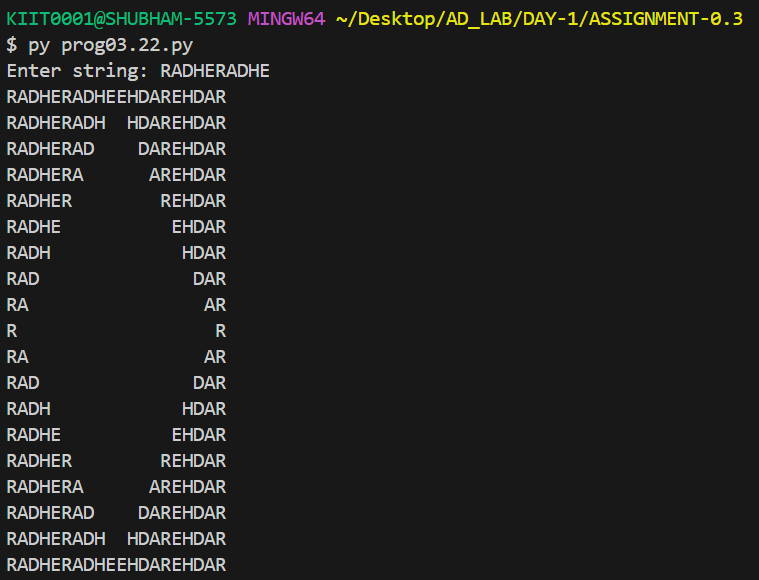
KIITCSITTISCTIIK

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def print\_string\_pattern(s):

n = len(s)

for i in range(n):

left = s[:n-i]

right = s[i:]

print(left, end=" " \* (2\*i))

print(right)

for i in range(n-2, -1, -1):

left = s[:n-i]

right = s[i:]

print(left, end=" " \* (2\*i))

print(right)

string = input("Enter a string: ")

print\_string\_pattern(string)

**Program No: 03.23**

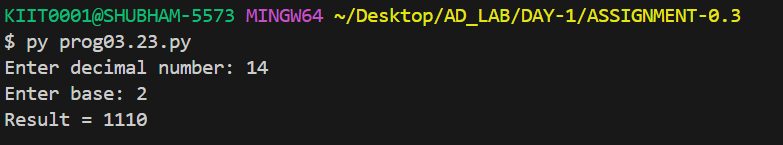
**Program Title:**

WAP to convert a decimal number into its equivalent number with base b. Decimal number and b are

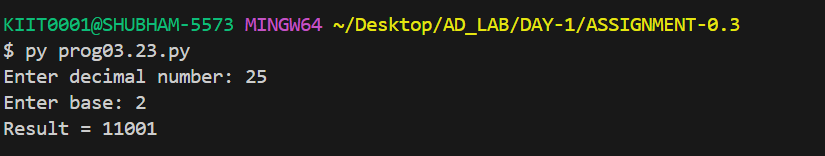
the user input.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def dec\_to\_base(num, b):

digits = "0123456789ABCDEF"

res = ""

while num > 0:

res = digits[num % b] + res

num //= b

return res

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

b = int(input("Enter base: "))

print("Result =", dec\_to\_base(num, b))

**Program No: 03.24**

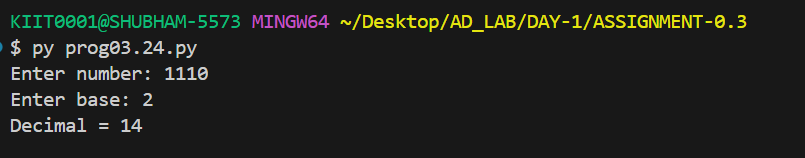
**Program Title:**

WAP to convert a number with base b into its equivalent decimal number. Number with base b & b are

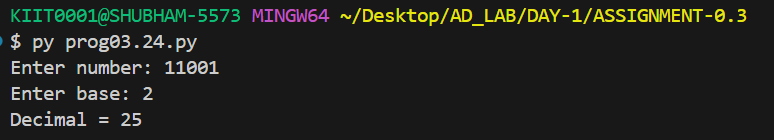
the user input.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def base\_to\_dec(num, b):

digits = "0123456789ABCDEF"

value = 0

for ch in num.upper():

value = value \* b + digits.index(ch)

return value

num = input("Enter number: ")

b = int(input("Enter base: "))

print("Decimal =", base\_to\_dec(num, b))

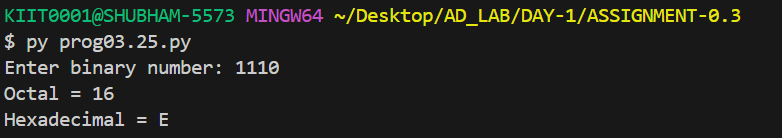
**Program No: 03.25**

**Program Title:**

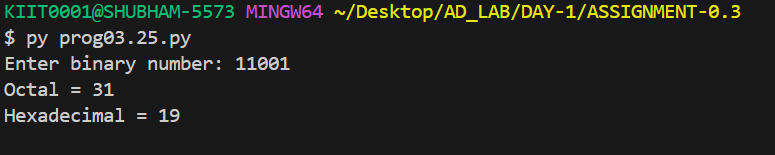
WAP to convert a binary number to its equivalent octal & hexa-decimal number.

**Input/Output Screenshots:**

**RUN-1:**



**RUN-2**



**Source code**

def bin\_to\_dec(binary):

dec = 0

for digit in binary:

dec = dec \* 2 + int(digit)

return dec

def dec\_to\_oct(n):

res = ""

while n > 0:

res = str(n % 8) + res

n //= 8

return res

def dec\_to\_hex(n):

digits = "0123456789ABCDEF"

res = ""

while n > 0:

res = digits[n % 16] + res

n //= 16

return res

binary = input("Enter binary number: ")

dec = bin\_to\_dec(binary)

print("Octal =", dec\_to\_oct(dec))

print("Hexadecimal =", dec\_to\_hex(dec))