YUVAN CHARAN,A

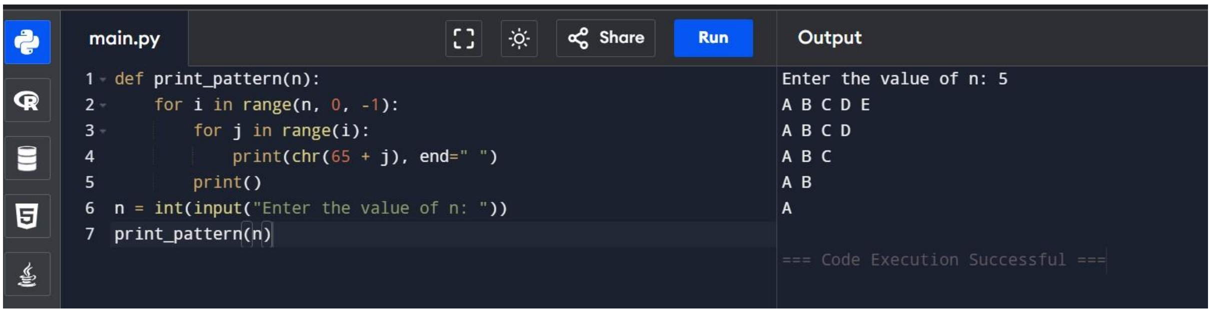
192424383

SLOT-B

PYTHON PROGRAMMING FOR BLOCK CHAIN PROJECTS

CSA0815

1.Write a PYTHON program to produce following design (If user enters n value as 5)

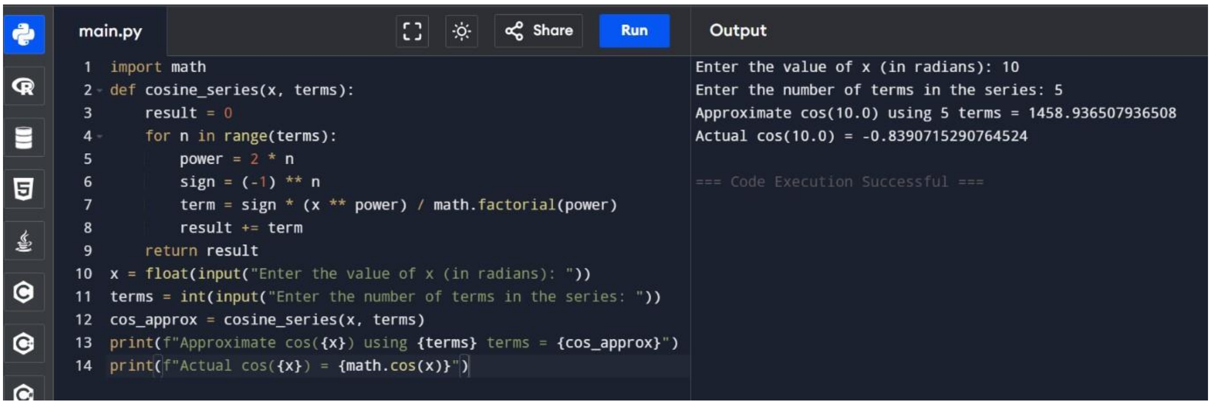


```python
def print_pattern(n):
    for i in range(n, 0, -1):
        for j in range(i):
            print(chr(65 + j), end=" ")
        print()
n = int(input("Enter the value of n: "))
print_pattern(n)
```

```
Enter the value of n: 5
A B C D E
A B C D
A B C
A B
A

=== Code Execution Successful ===
```

2.Write a PYTHON program to compute the cosine series: cos(x) = 1 – x2 / 2! + x4 / 4! – x6 / 6! + … xn / n!

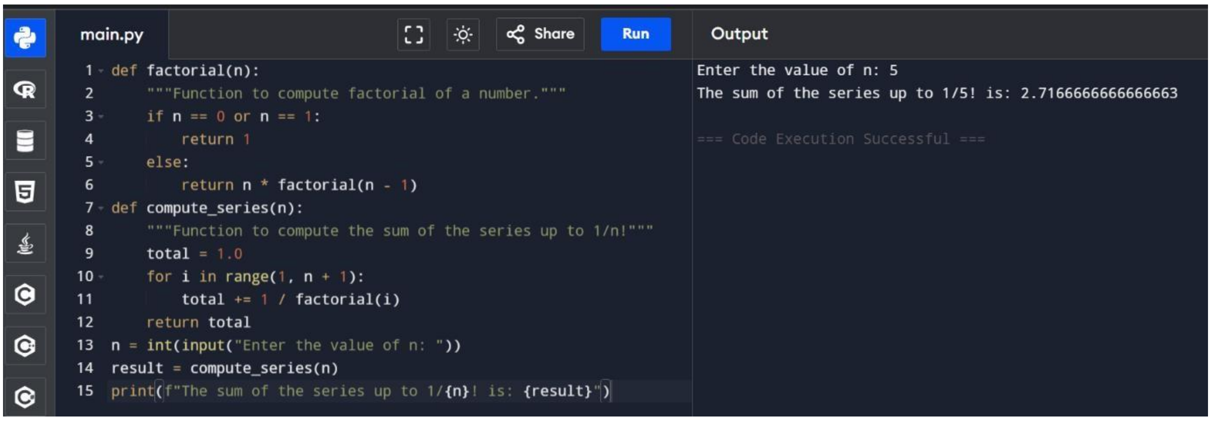


```python
import math
def cosine_series(x, terms):
    result = 0
    for n in range(terms):
        power = 2 * n
        sign = (-1) ** n
        term = sign * (x ** power) / math.factorial(power)
        result += term
    return result
x = float(input("Enter the value of x (in radians): "))
terms = int(input("Enter the number of terms in the series: "))
cos_approx = cosine_series(x, terms)
print(f"Approximate cos({x}) using {terms} terms = {cos_approx}")
print(f"Actual cos({x}) = {math.cos(x)}")
```

```
Enter the value of x (in radians): 10
Enter the number of terms in the series: 5
Approximate cos(10.0) using 5 terms = 1458.936507936508
Actual cos(10.0) = -0.8390715290764524

=== Code Execution Successful ===
```

3.Write a PYTHON program to sum the given sequence 1 + 1/ 1! + 1/ 2! + 1/3! + …. + 1/n!

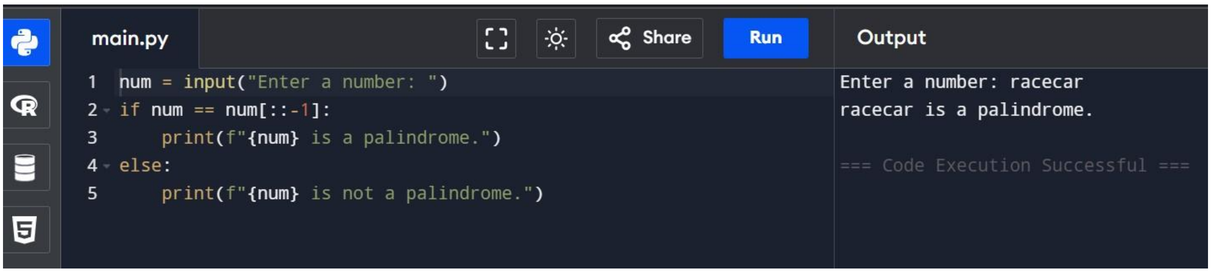


```python
def factorial(n):
    """Function to compute factorial of a number."""
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
def compute_series(n):
    """Function to compute the sum of the series up to 1/n!"""
    total = 1.0
    for i in range(1, n + 1):
        total += 1 / factorial(i)
    return total
n = int(input("Enter the value of n: "))
result = compute_series(n)
print(f"The sum of the series up to 1/{n}! is: {result}")
```

```
Enter the value of n: 5
The sum of the series up to 1/5! is: 2.7166666666666663

=== Code Execution Successful ===
```

4.Write a PYTHON program to check the entered number is palindrome or not



```python
num = input("Enter a number: ")
if num == num[::-1]:
    print(f"{num} is a palindrome.")
else:
    print(f"{num} is not a palindrome.")
```

```
Enter a number: racecar
racecar is a palindrome.

=== Code Execution Successful ===
```

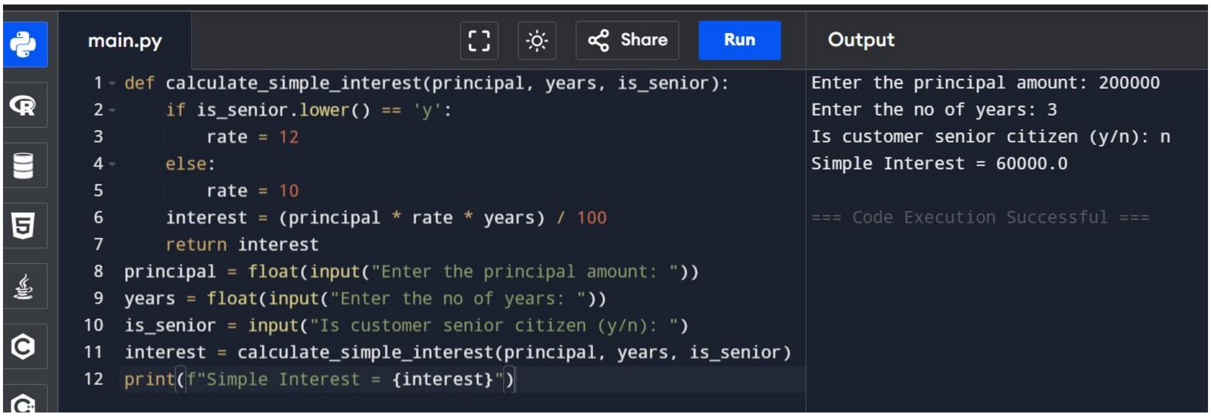
5. **Write a python program using function to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered 12 percentage rate of interest; for all other customers, the ROI is 10 percentage.**

Sample Input:

Enter the principal amount: 200000

Enter the no of years: 3

Is customer senior citizen (y/n): n

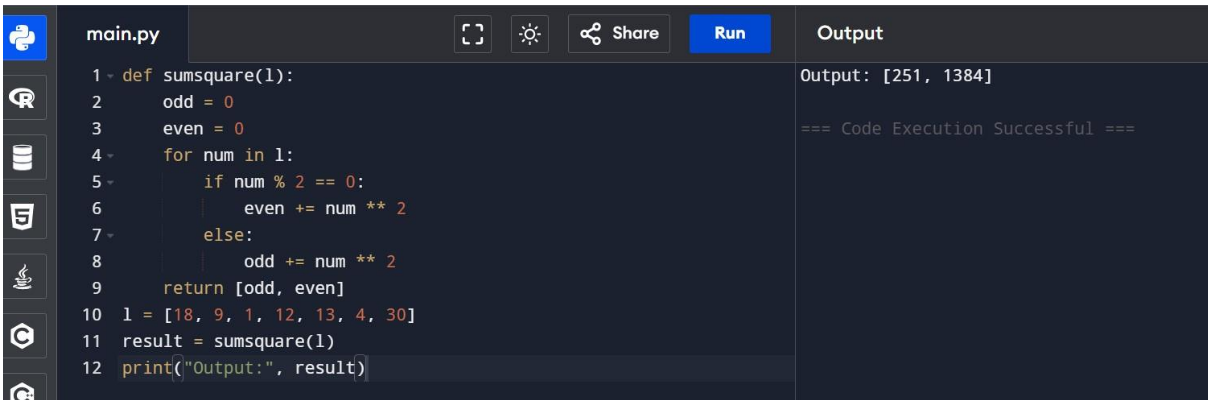


```python
def calculate_simple_interest(principal, years, is_senior):
    if is_senior.lower() == 'y':
        rate = 12
    else:
        rate = 10
    interest = (principal * rate * years) / 100
    return interest
principal = float(input("Enter the principal amount: "))
years = float(input("Enter the no of years: "))
is_senior = input("Is customer senior citizen (y/n): ")
interest = calculate_simple_interest(principal, years, is_senior)
print(f"Simple Interest = {interest}")
```

```
Enter the principal amount: 200000
Enter the no of years: 3
Is customer senior citizen (y/n): n
Simple Interest = 60000.0

=== Code Execution Successful ===
```

6. **Write a Python function sumsquare(l) that takes a nonempty list of integers and returns a list [odd,even], where odd is the sum of squares of all the odd numbers in l and even is the sum of squares of all the even numbers in l.**

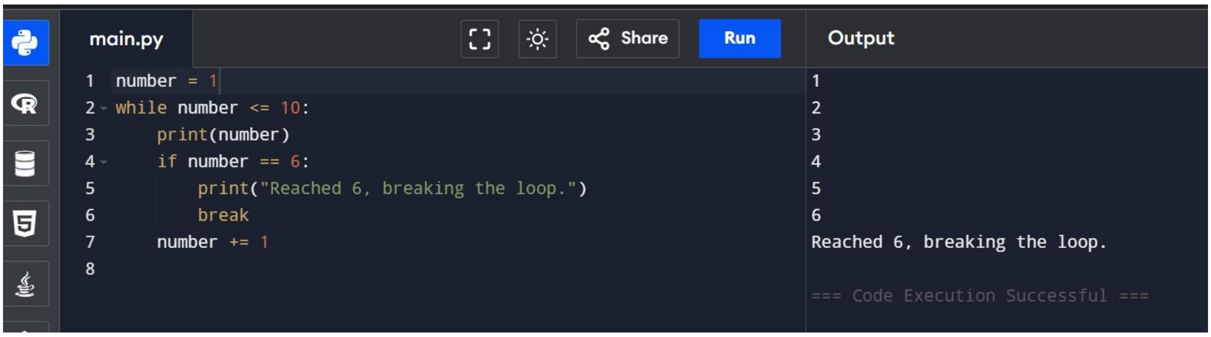


```python
def sumsquare(l):
    odd = 0
    even = 0
    for num in l:
        if num % 2 == 0:
            even += num ** 2
        else:
            odd += num ** 2
    return [odd, even]
l = [18, 9, 1, 12, 13, 4, 30]
result = sumsquare(l)
print("Output:", result)
```

```
Output: [251, 1384]

=== Code Execution Successful ===
```

7. **Write a PYTHON program to Print numbers using a loop with a break condition**

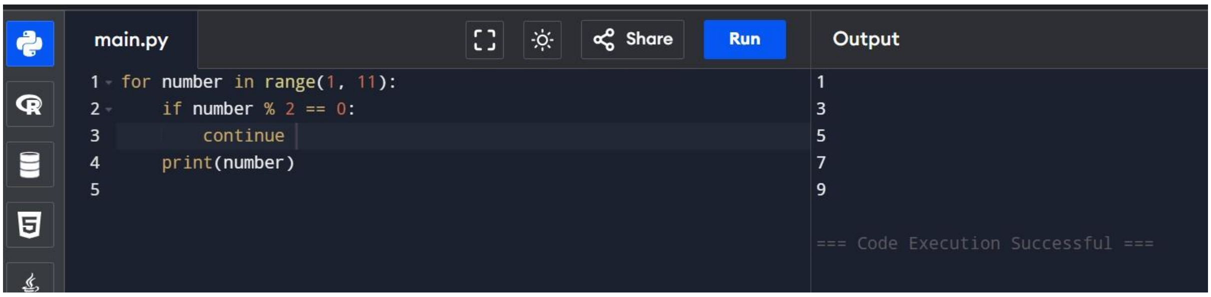


```python
number = 1
while number <= 10:
    print(number)
    if number == 6:
        print("Reached 6, breaking the loop.")
        break
    number += 1
```

```
1
2
3
4
5
6
Reached 6, breaking the loop.

=== Code Execution Successful ===
```

8. **Write a PYTHON program to Skip even numbers using continue statement**

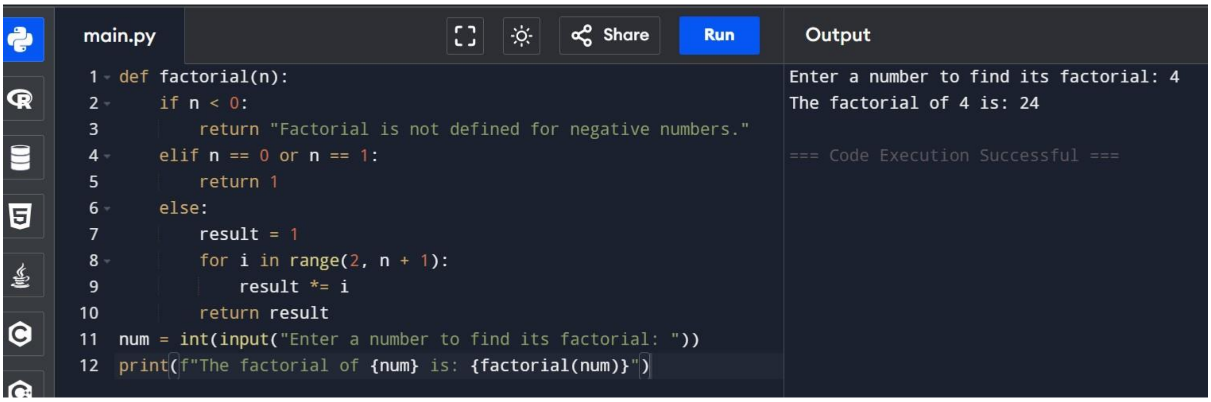


```python
for number in range(1, 11):
    if number % 2 == 0:
        continue
    print(number)
```

```
1
3
5
7
9

=== Code Execution Successful ===
```

9. **Write a PYTHON program to Find factorial of a number**

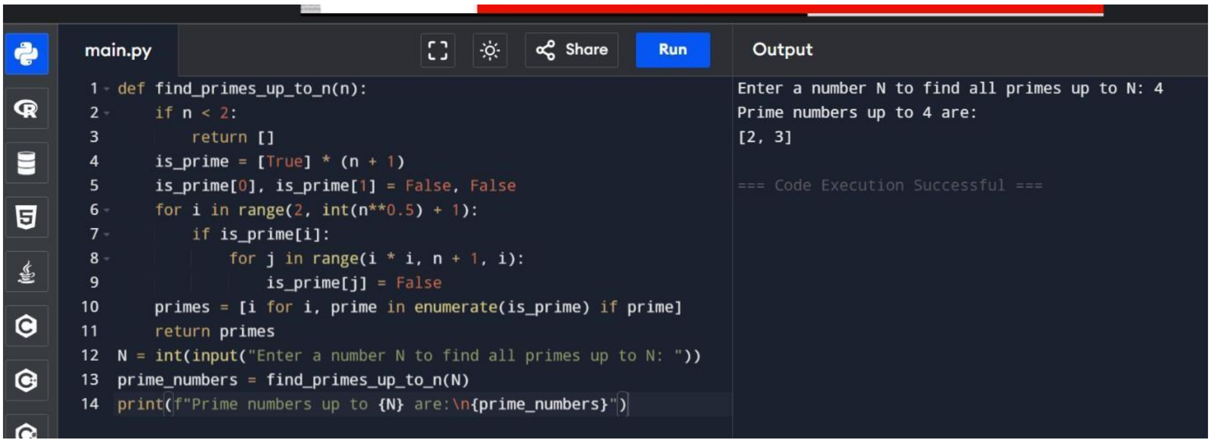


```python
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
num = int(input("Enter a number to find its factorial: "))
print(f"The factorial of {num} is: {factorial(num)}")
```

```
Enter a number to find its factorial: 4
The factorial of 4 is: 24

=== Code Execution Successful ===
```

10. **Write a PYTHON program to Find prime numbers up to N**

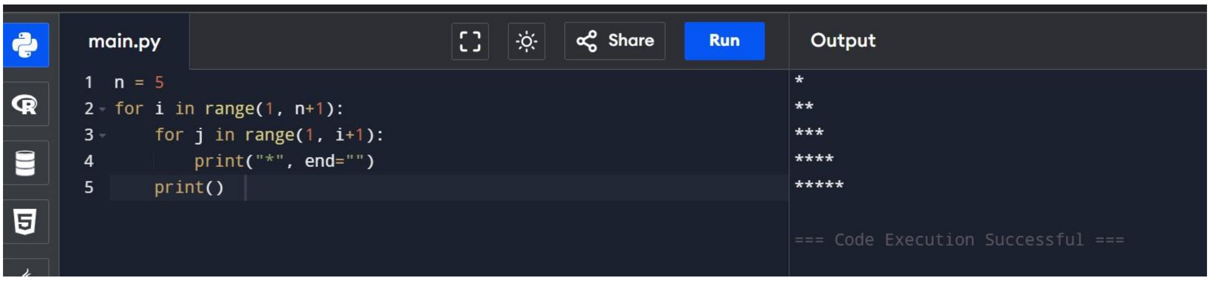


```python
def find_primes_up_to_n(n):
    if n < 2:
        return []
    is_prime = [True] * (n + 1)
    is_prime[0], is_prime[1] = False, False
    for i in range(2, int(n**0.5) + 1):
        if is_prime[i]:
            for j in range(i * i, n + 1, i):
                is_prime[j] = False
    primes = [i for i, prime in enumerate(is_prime) if prime]
    return primes
N = int(input("Enter a number N to find all primes up to N: "))
prime_numbers = find_primes_up_to_n(N)
print(f"Prime numbers up to {N} are:\n{prime_numbers}")
```

```
Enter a number N to find all primes up to N: 4
Prime numbers up to 4 are:
[2, 3]

=== Code Execution Successful ===
```

11. **Write a PYTHON program to Print a pattern using nested loops**



```python
n = 5
for i in range(1, n+1):
    for j in range(1, i+1):
        print("*", end="")
    print()
```

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


=== Code Execution Successful ===
```