P.DILLI BABU

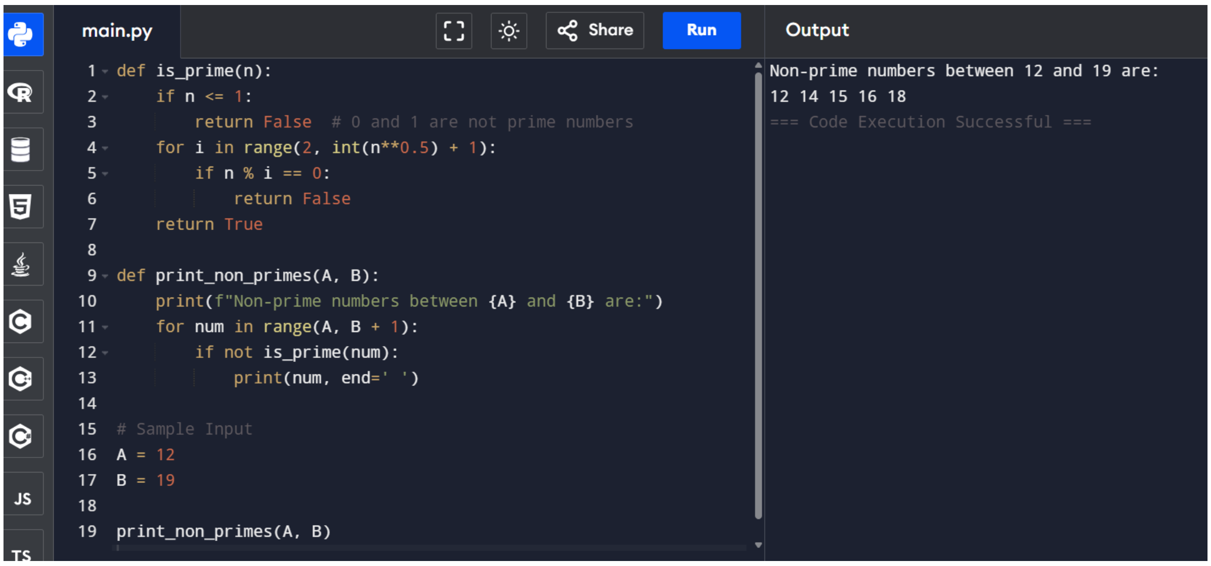
192472284

SLOT-B

PYTHON PROGRAMMING FOR BLOCK CHAIN PROJECTS

CSA0815

1. Write a program to print all the Non-Prime numbers between A and B? Sample Input: A = 12 B = 19



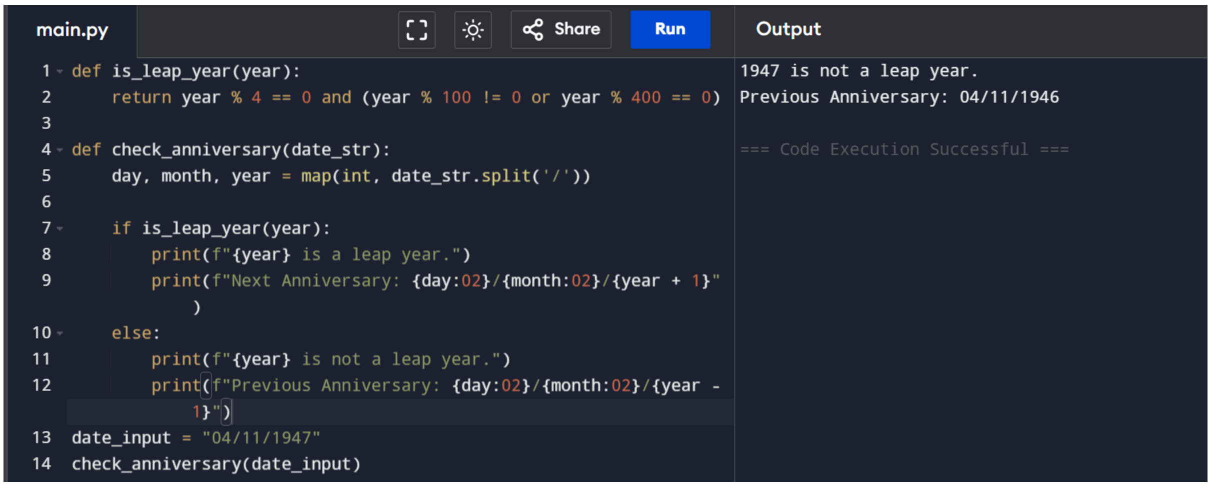
```python
def is_prime(n):
    if n <= 1:
        return False  # 0 and 1 are not prime numbers
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def print_non_primes(A, B):
    print(f"Non-prime numbers between {A} and {B} are:")
    for num in range(A, B + 1):
        if not is_prime(num):
            print(num, end=' ')

# Sample Input
A = 12
B = 19

print_non_primes(A, B)
```

```
Non-prime numbers between 12 and 19 are:
12 14 15 16 18
=== Code Execution Successful ===
```

2. Find the year of the given Anniversary is leap year or not. If leap year then print the next Anniversary, if not leap year then print the previous Anniversary.

Sample Input:

Enter Date: 04/11/1947

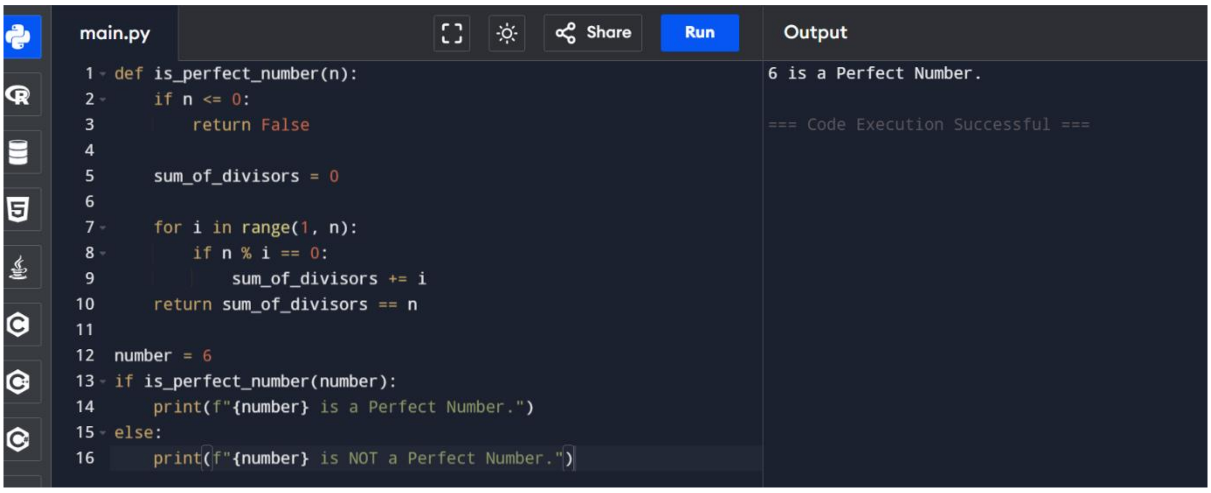


```python
def is_leap_year(year):
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

def check_anniversary(date_str):
    day, month, year = map(int, date_str.split('/'))

    if is_leap_year(year):
        print(f"{year} is a leap year.")
        print(f"Next Anniversary: {day:02}/{month:02}/{year + 1}"
            )
    else:
        print(f"{year} is not a leap year.")
        print(f"Previous Anniversary: {day:02}/{month:02}/{year -
            1}")
date_input = "04/11/1947"
check_anniversary(date_input)
```

```
1947 is not a leap year.
Previous Anniversary: 04/11/1946

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

3. Write a program to print the given number is Perfect number or not?

Sample Input: Given Number: 6



```python
def is_perfect_number(n):
    if n <= 0:
        return False

    sum_of_divisors = 0

    for i in range(1, n):
        if n % i == 0:
            sum_of_divisors += i
    return sum_of_divisors == n

number = 6
if is_perfect_number(number):
    print(f"{number} is a Perfect Number.")
else:
    print(f"{number} is NOT a Perfect Number.")
```

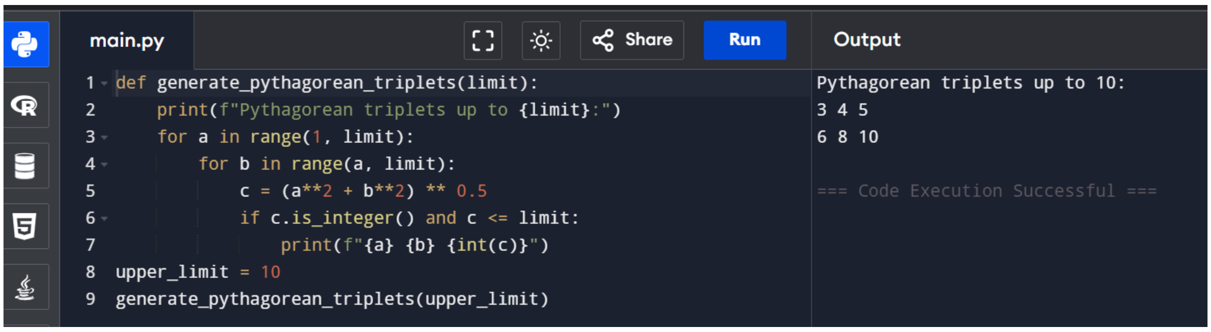
```
6 is a Perfect Number.

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

4. Write a program to generate Pythagorean Triplets for the given limit.

Enter upper limit: 10

3 4 5

8 6 10

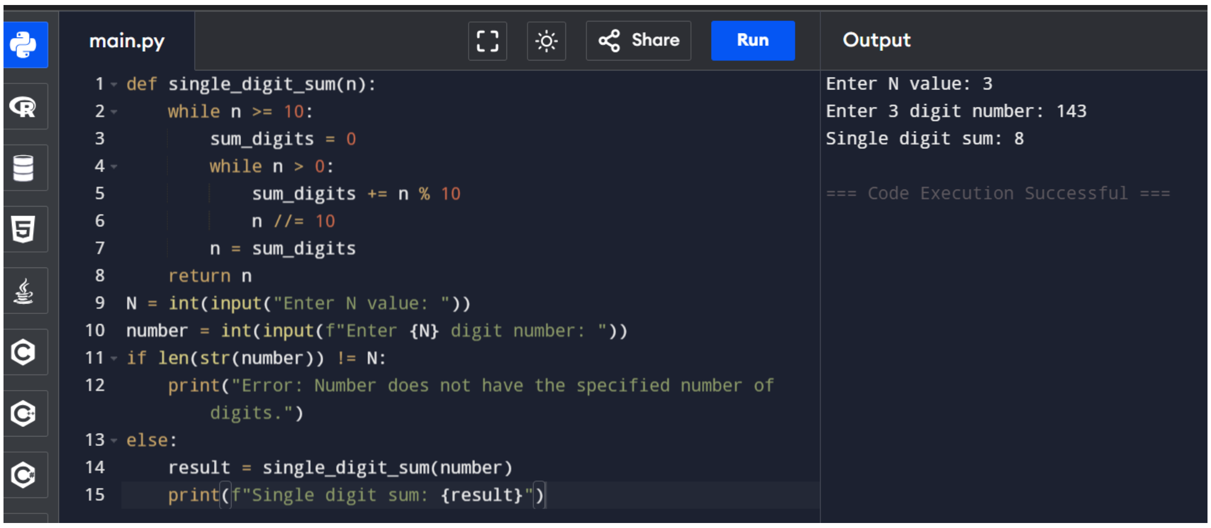


```python
def generate_pythagorean_triplets(limit):
    print(f"Pythagorean triplets up to {limit}:")
    for a in range(1, limit):
        for b in range(a, limit):
            c = (a**2 + b**2) ** 0.5
            if c.is_integer() and c <= limit:
                print(f"{a} {b} {int(c)}")
upper_limit = 10
generate_pythagorean_triplets(upper_limit)
```

```
Pythagorean triplets up to 10:
3 4 5
6 8 10

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

5. Write a program to find the sum of digits of N digit number (sum should be single digit)

Sample Input: Enter N value : 3 Enter 3 digit number: 143

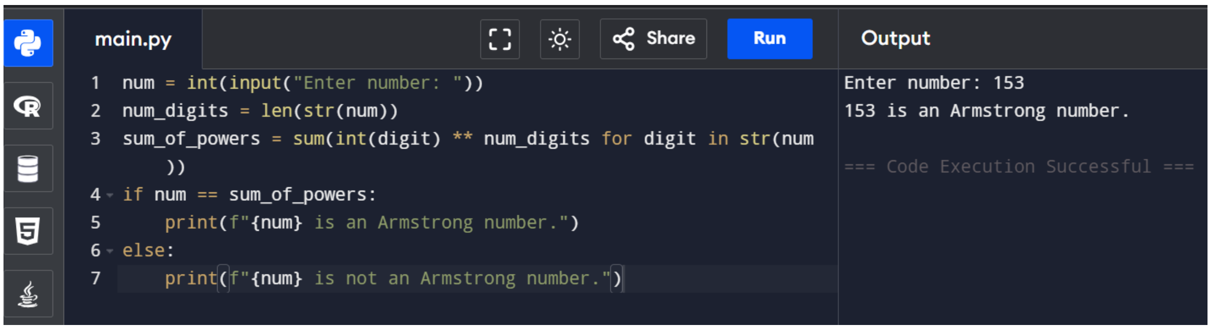


```python
def single_digit_sum(n):
    while n >= 10:
        sum_digits = 0
        while n > 0:
            sum_digits += n % 10
            n //= 10
        n = sum_digits
    return n
N = int(input("Enter N value: "))
number = int(input(f"Enter {N} digit number: "))
if len(str(number)) != N:
    print("Error: Number does not have the specified number of
        digits.")
else:
    result = single_digit_sum(number)
    print(f"Single digit sum: {result}")
```

```
Enter N value: 3
Enter 3 digit number: 143
Single digit sum: 8

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

6. Program to find whether the given number is Armstrong number or not

Sample Input: Enter number: 153

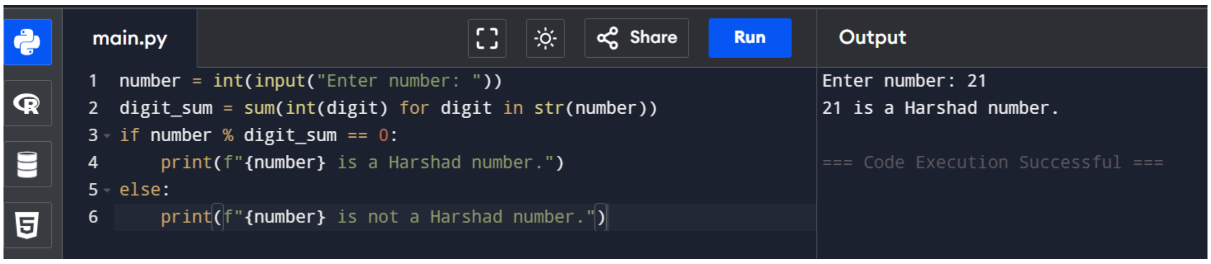


```python
num = int(input("Enter number: "))
num_digits = len(str(num))
sum_of_powers = sum(int(digit) ** num_digits for digit in str(num
    ))
if num == sum_of_powers:
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

```
Enter number: 153
153 is an Armstrong number.

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

7. Program to find whether the given number is Harshad number or not

Sample Input: Enter number: 21

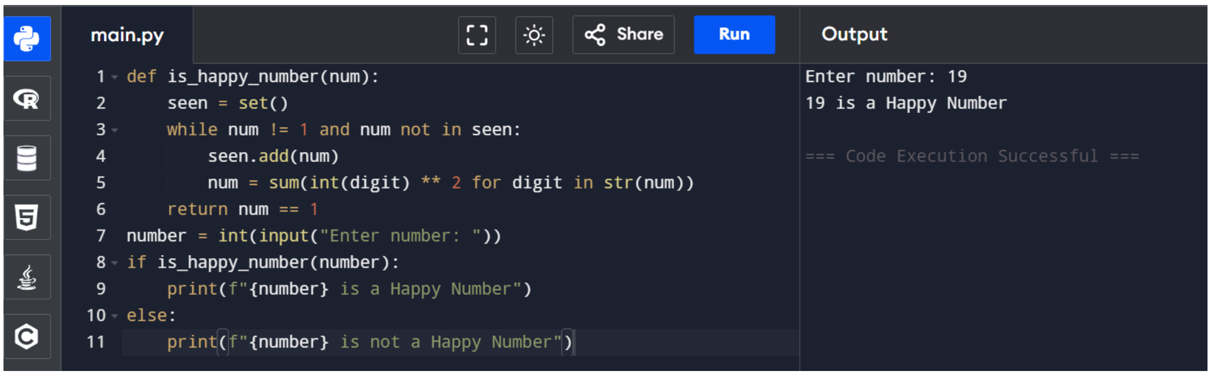


```python
number = int(input("Enter number: "))
digit_sum = sum(int(digit) for digit in str(number))
if number % digit_sum == 0:
    print(f"{number} is a Harshad number.")
else:
    print(f"{number} is not a Harshad number.")
```

```
Enter number: 21
21 is a Harshad number.

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

8. Program to find whether the given number is Happy number or not

Sample Input: Enter number: 19

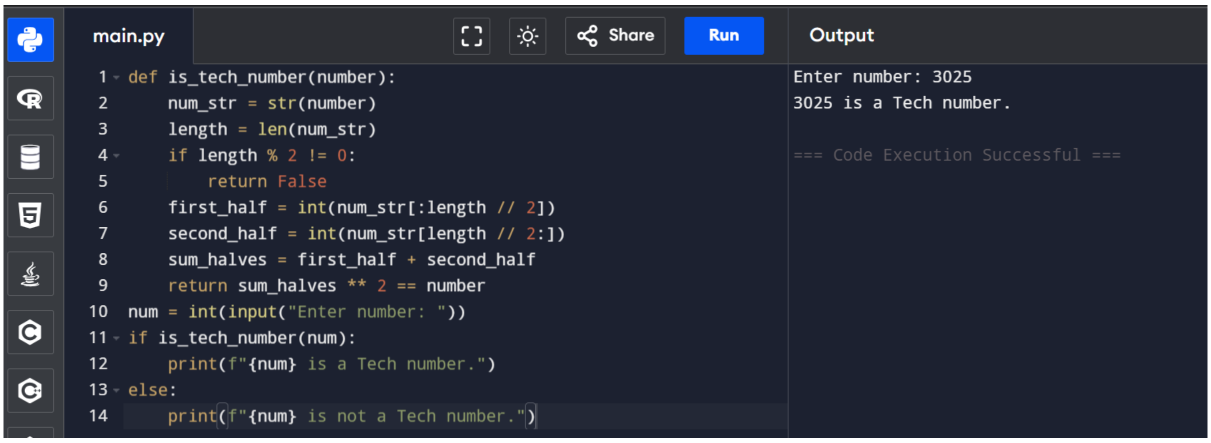


```python
def is_happy_number(num):
    seen = set()
    while num != 1 and num not in seen:
        seen.add(num)
        num = sum(int(digit) ** 2 for digit in str(num))
    return num == 1
number = int(input("Enter number: "))
if is_happy_number(number):
    print(f"{number} is a Happy Number")
else:
    print(f"{number} is not a Happy Number")
```

```
Enter number: 19
19 is a Happy Number

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

9. Program to find whether the given number is Tech number or not

Sample Input: Enter number: 3025



```python
def is_tech_number(number):
    num_str = str(number)
    length = len(num_str)
    if length % 2 != 0:
        return False
    first_half = int(num_str[:length // 2])
    second_half = int(num_str[length // 2:])
    sum_halves = first_half + second_half
    return sum_halves ** 2 == number
num = int(input("Enter number: "))
if is_tech_number(num):
    print(f"{num} is a Tech number.")
else:
    print(f"{num} is not a Tech number.")
```

```
Enter number: 3025
3025 is a Tech number.

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

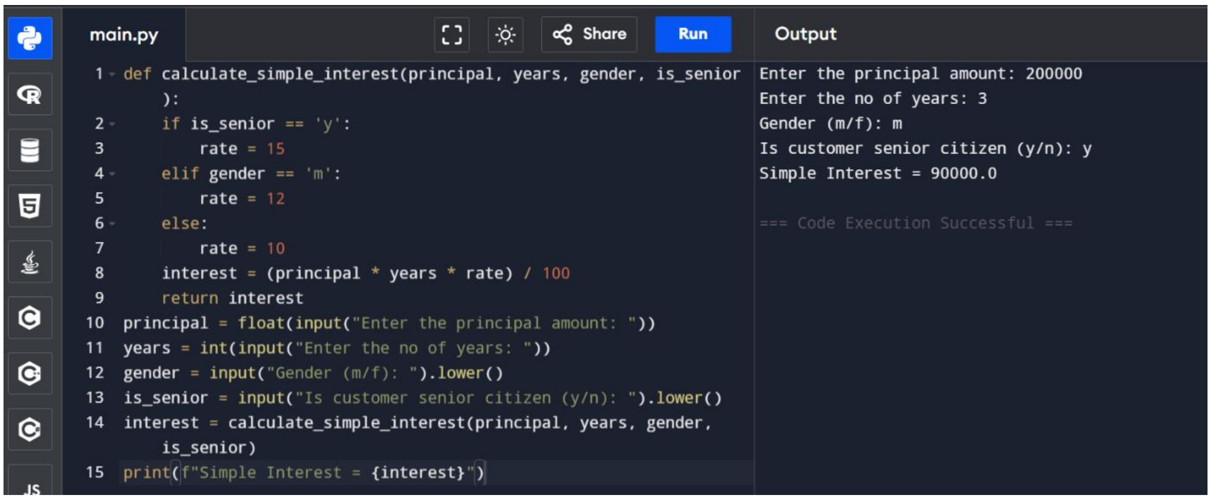
10. Write a program using function to calculate the simple interest. Suppose the customer is a senior citizen. She is being offered 15 percent rate of interest; he is being offered 12 percent rate of interest for all other customers, the ROI is 10 percent.

Sample Input:

Enter the principal amount: 200000 Enter the no of years: 3

Gender (m/f): m

Is customer senior citizen (y/n): n



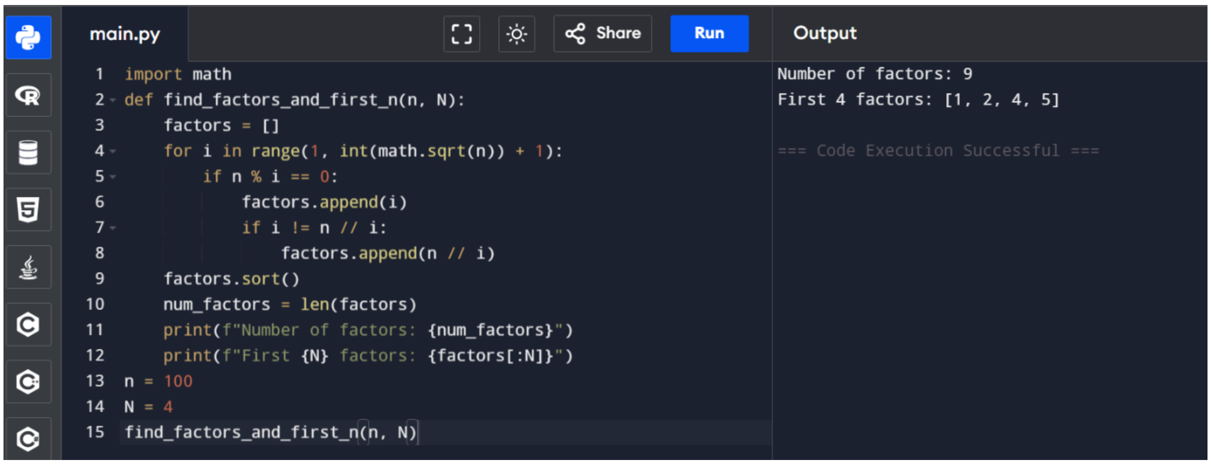
```python
def calculate_simple_interest(principal, years, gender, is_senior
    ):
    if is_senior == 'y':
        rate = 15
    elif gender == 'm':
        rate = 12
    else:
        rate = 10
    interest = (principal * years * rate) / 100
    return interest
principal = float(input("Enter the principal amount: "))
years = int(input("Enter the no of years: "))
gender = input("Gender (m/f): ").lower()
is_senior = input("Is customer senior citizen (y/n): ").lower()
interest = calculate_simple_interest(principal, years, gender,
    is_senior)
print(f"Simple Interest = {interest}")
```

```
Enter the principal amount: 200000
Enter the no of years: 3
Gender (m/f): m
Is customer senior citizen (y/n): y
Simple Interest = 90000.0

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

11. Find the number of factors for the given number and print the 1st N factors of the given number.

Sample Input: Given number: 100

N: 4



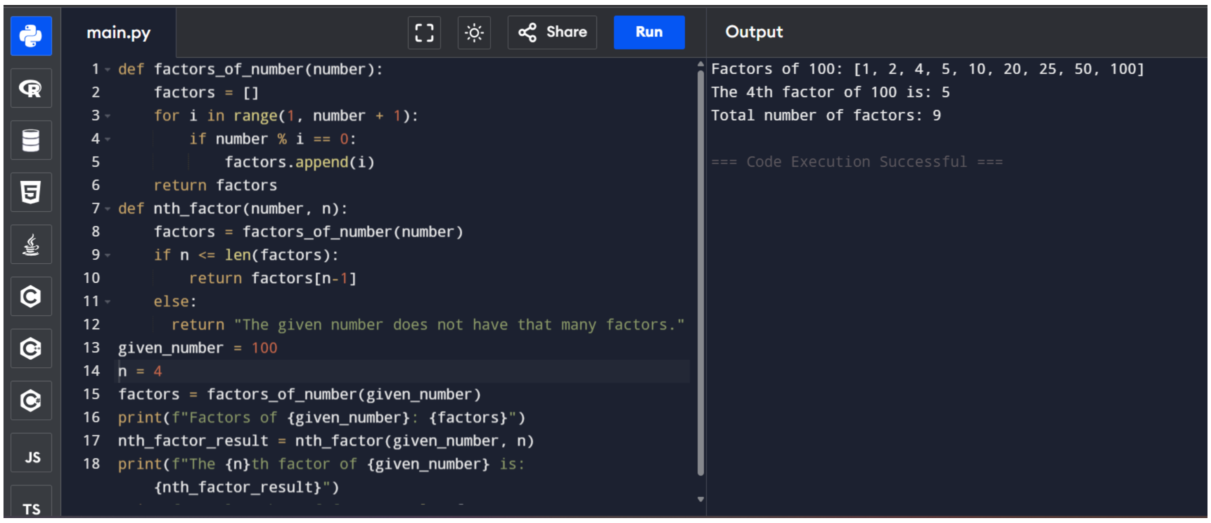
```python
import math
def find_factors_and_first_n(n, N):
    factors = []
    for i in range(1, int(math.sqrt(n)) + 1):
        if n % i == 0:
            factors.append(i)
            if i != n // i:
                factors.append(n // i)
    factors.sort()
    num_factors = len(factors)
    print(f"Number of factors: {num_factors}")
    print(f"First {N} factors: {factors[:N]}")
n = 100
N = 4
find_factors_and_first_n(n, N)
```

```
Number of factors: 9
First 4 factors: [1, 2, 4, 5]

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

12. Write a program to print number of factors and to print nth factor of the given number.

Sample Input: Given Number: 100

N = 4



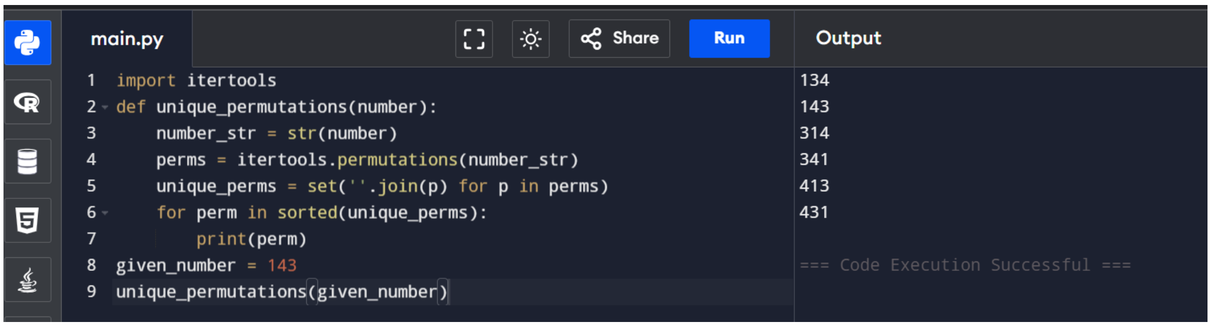
```python
def factors_of_number(number):
    factors = []
    for i in range(1, number + 1):
        if number % i == 0:
            factors.append(i)
    return factors
def nth_factor(number, n):
    factors = factors_of_number(number)
    if n <= len(factors):
        return factors[n-1]
    else:
        return "The given number does not have that many factors."
given_number = 100
n = 4
factors = factors_of_number(given_number)
print(f"Factors of {given_number}: {factors}")
nth_factor_result = nth_factor(given_number, n)
print(f"The {n}th factor of {given_number} is:
    {nth_factor_result}")
```

```
Factors of 100: [1, 2, 4, 5, 10, 20, 25, 50, 100]
The 4th factor of 100 is: 5
Total number of factors: 9

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

13. Write a program to print unique permutations of a given number Sample

Input:

Given Number: 143



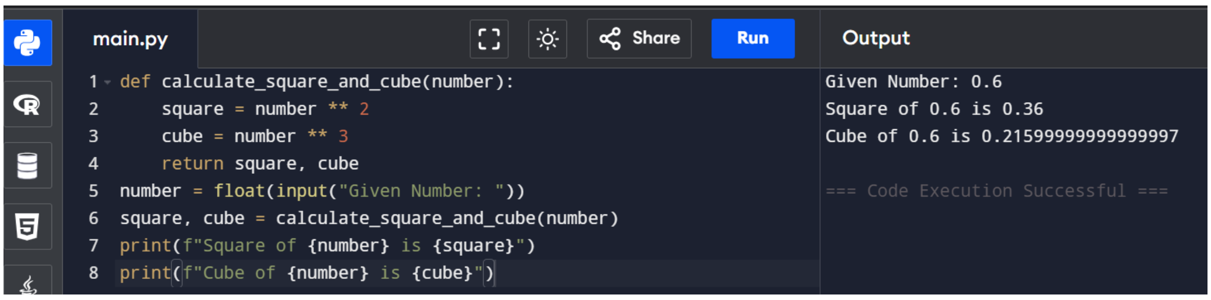
```python
import itertools
def unique_permutations(number):
    number_str = str(number)
    perms = itertools.permutations(number_str)
    unique_perms = set(''.join(p) for p in perms)
    for perm in sorted(unique_perms):
        print(perm)
given_number = 143
unique_permutations(given_number)
```

```
134
143
314
341
413
431

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

14. Write a program to find the square, cube of the given decimal number

Sample Input:

Given Number: 0.6

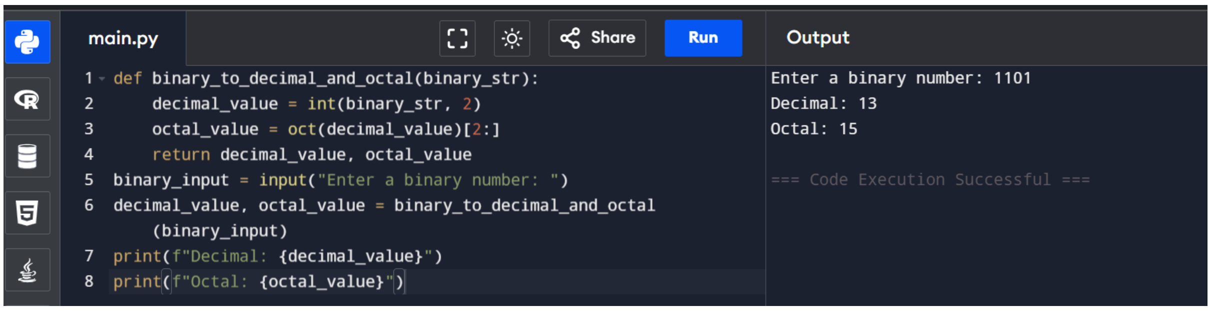


```python
def calculate_square_and_cube(number):
    square = number ** 2
    cube = number ** 3
    return square, cube
number = float(input("Given Number: "))
square, cube = calculate_square_and_cube(number)
print(f"Square of {number} is {square}")
print(f"Cube of {number} is {cube}")
```

```
Given Number: 0.6
Square of 0.6 is 0.36
Cube of 0.6 is 0.21599999999999997

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

15. Write a program to convert the Binary to Decimal, Octal Sample Input:

Given Number: 1101



```python
def binary_to_decimal_and_octal(binary_str):
    decimal_value = int(binary_str, 2)
    octal_value = oct(decimal_value)[2:]
    return decimal_value, octal_value
binary_input = input("Enter a binary number: ")
decimal_value, octal_value = binary_to_decimal_and_octal
    (binary_input)
print(f"Decimal: {decimal_value}")
print(f"Octal: {octal_value}")
```

```
Enter a binary number: 1101
Decimal: 13
Octal: 15

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