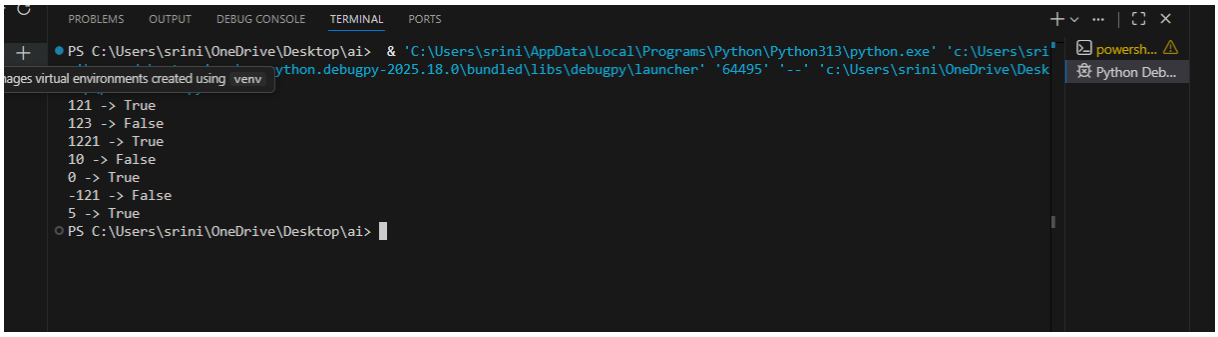
1.# Write a Python function that checks whether a given number is a palindrome.

Code:

```
def is_palindrome(number):
    if number < 0:
        return False
    temp=number
    rev=0
    while temp != 0:
        rev = rev * 10 + temp % 10
        temp //= 10
    return rev == number
numbers = [121, 123, 1221, 10, 0, -121, 5]
for n in numbers:
    print(f"{n} -> {is_palindrome(n)}")
```

Step-by-Step Explanation

1. If the number is **negative**, return False.
2. Store the original number in temp.
3. Reverse the number using a while loop.
4. Compare the reversed number with the original.
5. If both are equal, it is a **palindrome**; otherwise, it is **not**.



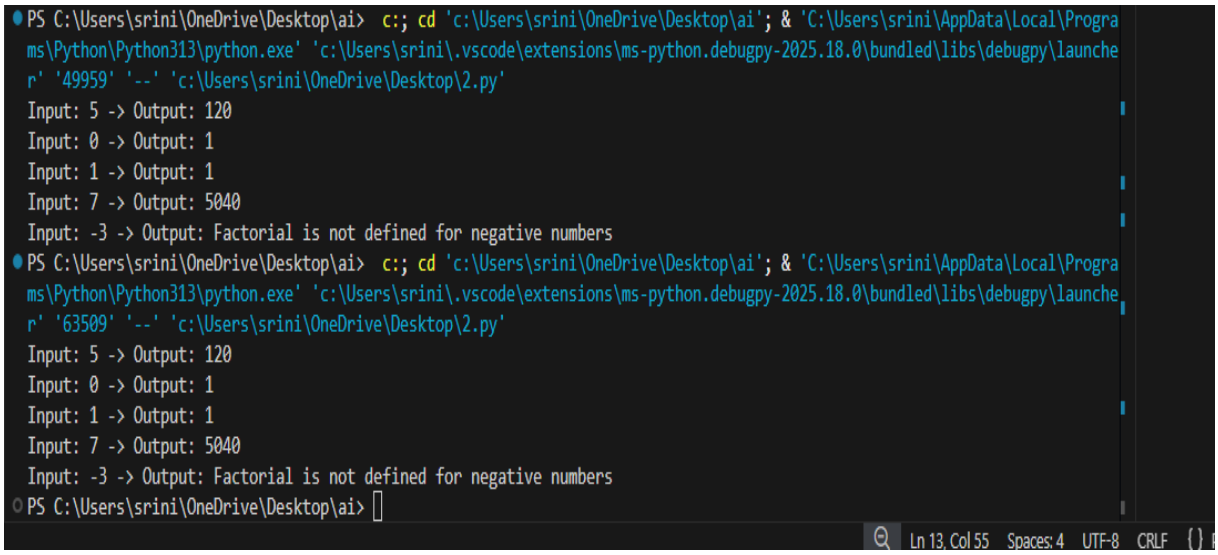
2. #Write a Python function to compute the factorial of a given number.

Code:

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    if n==0 or n==1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
test_values = [5, 0, 1, 7, -3]
for val in test_values:
    print(f"Input: {val} -> Output: {factorial(val)}")
```

Step-by-Step Explanation (Factorial Function)

1. If the number is **negative**, return an error message.
2. If the number is **0 or 1**, return 1 (base case).
3. Initialize result = 1.
4. Use a for loop from 2 to n and multiply each value with result.
5. Return the final factorial value.



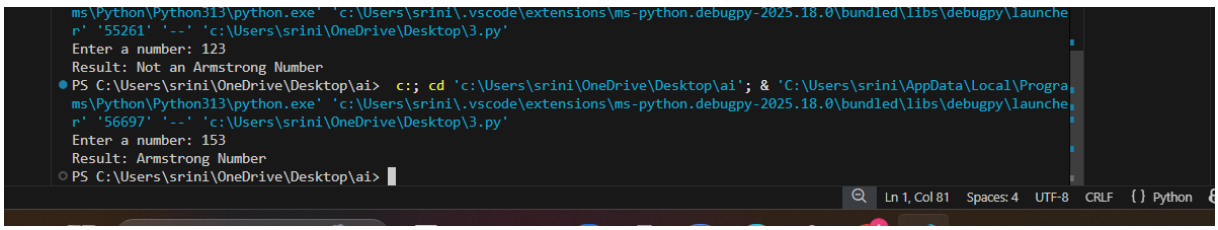
3.#Write a Python function to check whether a given number is an Armstrong number.

Code:

```
def is_armstrong(number):
    if number < 0:
        return "Invalid input"
    digits = str(number)
    power = len(digits)
    total = 0
    for d in digits:
        total += int(d) ** power
    if total == number:
        return "Armstrong Number"
    else:
        return "Not an Armstrong Number"
n = int(input("Enter a number: "))
print("Result:", is_armstrong(n))
```

Step-by-Step Explanation (Armstrong Number)

1. If the number is **negative**, return "Invalid input".
2. Convert the number to a **string** to get each digit easily.
3. Count the **number of digits**.
4. Raise each digit to the power of the total digits and **add them**.
5. Compare the sum with the original number.
6. If both are equal, it is an **Armstrong Number**; otherwise, it is **not**.

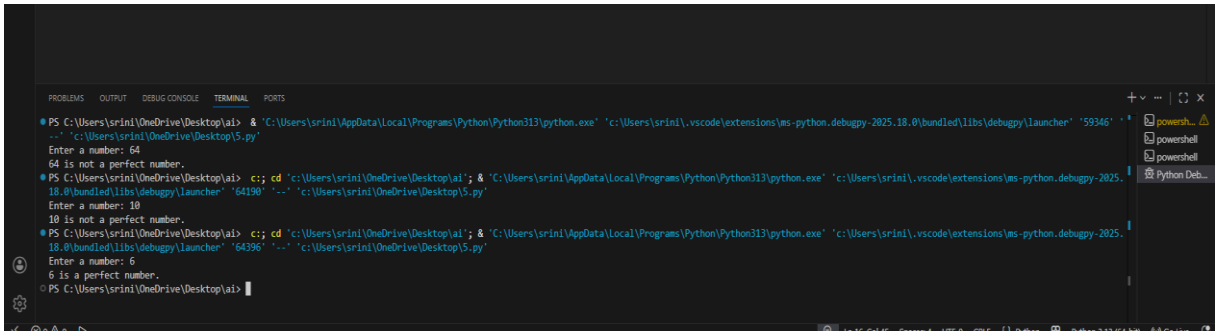


5. #Generate a Python function that checks whether a given number is a perfect number. The function should return True if the number is perfect, otherwise return False.

```
def is_perfect_number(n):
    if n < 1:
        return False
    total=1
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            total += i
            if i != n // i:
                total += n // i
    return total == n
num=int(input("Enter a number: "))
if is_perfect_number(num):
    print(f"{num} is a perfect number.")
else:
    print(f"{num} is not a perfect number.")
```

Step-by-Step Explanation (Perfect Number)

1. If the number is **less than 1**, return False.
2. Initialize total = 1 (since 1 is a proper divisor).
3. Loop from 2 to √n to find divisors.
4. If i divides n, add both i and n//i to total.
5. Compare the sum of proper divisors with the original number.
6. If both are equal, it is a **perfect number**; otherwise, it is **not**.



6. #Write a Python program that determines whether a given number is Even or Odd and includes proper input validation.

```
def even_or_odd():
    n=input("Enter a number: ")
    try:
        n=int(n)
        if n % 2 == 0:
            print("Even")
        else:
            print("Odd")
    except ValueError:
        print("Invalid input. Please enter an integer.")
even_or_odd()
```

Step-by-Step Explanation (Even or Odd with Validation)

1. Take input from the user.
2. Try to convert the input into an integer.
3. If conversion fails, display an **invalid input** message.
4. If the number is divisible by 2, print **Even**.
5. Otherwise, print **Odd**.