

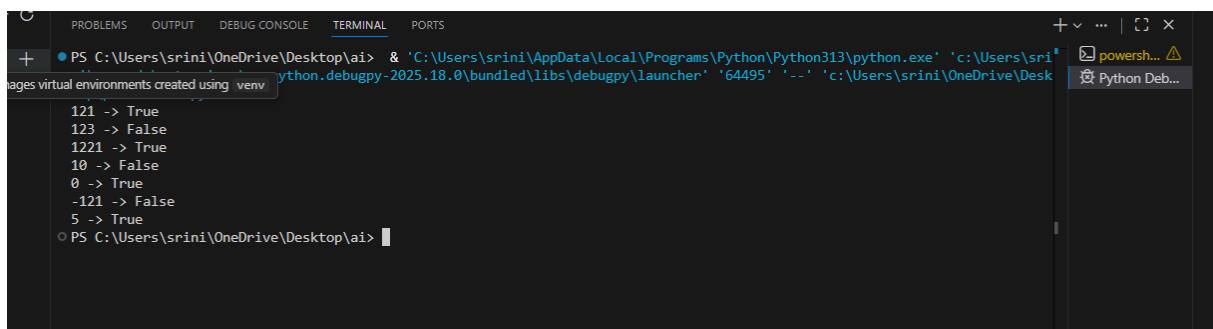
1.# Write a Python function that checks whether a given number is a palindrome.

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**.

A screenshot of a Python IDE's terminal window. The terminal shows the execution of the provided Python code. The output is as follows:
121 -> True
123 -> False
1221 -> True
10 -> False
0 -> True
-121 -> False
5 -> True
The terminal prompt is PS C:\Users\sri\OneDrive\Desktop\ai>.
The IDE interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal tab is active, showing the command prompt and the output of the Python script. The command prompt is PS C:\Users\sri\OneDrive\Desktop\ai> and the output is the result of the is_palindrome function for each number in the list [121, 123, 1221, 10, 0, -121, 5].

```
PS C:\Users\sri\OneDrive\Desktop\ai> & 'C:\Users\sri\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sri\OneDrive\Desktop\ai\python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '64495' '--' 'c:\Users\sri\OneDrive\Desktop\ai\ai.py'  
121 -> True  
123 -> False  
1221 -> True  
10 -> False  
0 -> True  
-121 -> False  
5 -> True  
PS C:\Users\sri\OneDrive\Desktop\ai>
```

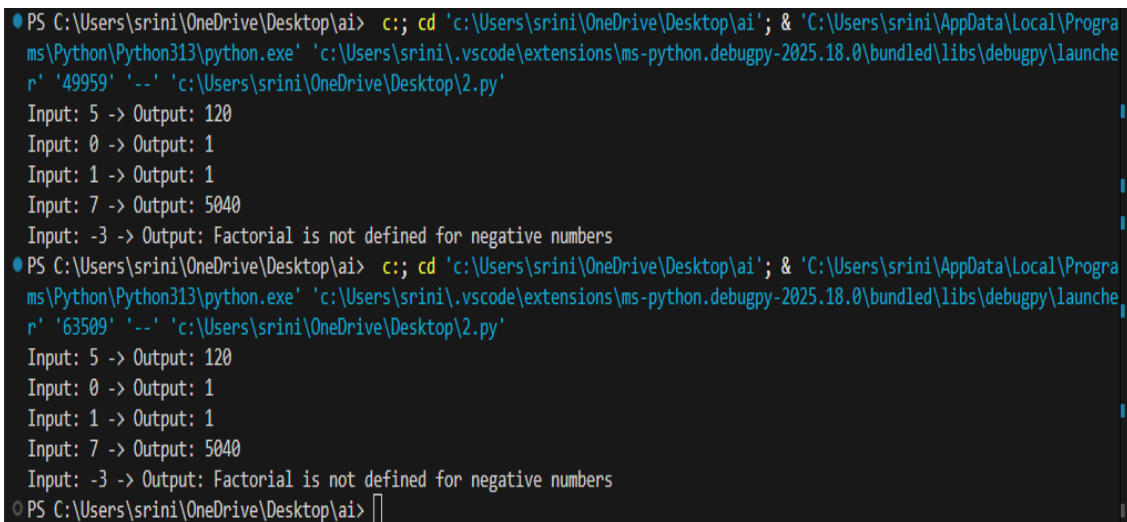
2. #Write a Python function to compute the factorial of a given number.

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.



```
PS C:\Users\sринi\OneDrive\Desktop\ai> c;; cd 'c:\Users\sринi\OneDrive\Desktop\ai'; & 'C:\Users\sринi\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sринi\.vscode\extensions\ms-python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '49959' '--' 'c:\Users\sринi\OneDrive\Desktop\2.py'  
Input: 5 -> Output: 120  
Input: 0 -> Output: 1  
Input: 1 -> Output: 1  
Input: 7 -> Output: 5040  
Input: -3 -> Output: Factorial is not defined for negative numbers  
PS C:\Users\sринi\OneDrive\Desktop\ai> c;; cd 'c:\Users\sринi\OneDrive\Desktop\ai'; & 'C:\Users\sринi\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sринi\.vscode\extensions\ms-python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '63509' '--' 'c:\Users\sринi\OneDrive\Desktop\2.py'  
Input: 5 -> Output: 120  
Input: 0 -> Output: 1  
Input: 1 -> Output: 1  
Input: 7 -> Output: 5040  
Input: -3 -> Output: Factorial is not defined for negative numbers  
PS C:\Users\sринi\OneDrive\Desktop\ai>
```

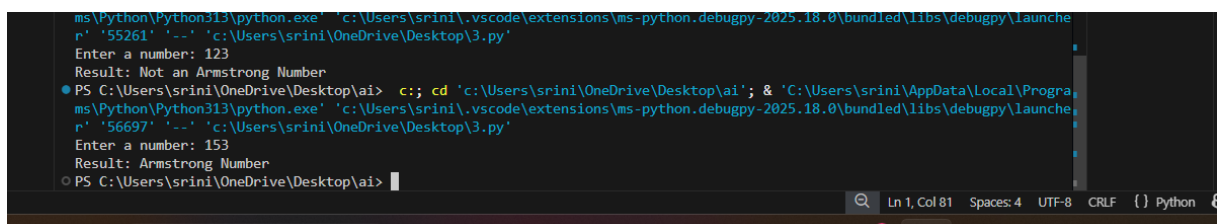
3.#Write a Python function to check whether a given number is an Armstrong number.

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**.



```
ms\Python\Python313\python.exe "c:\Users\sринi\.vscode\extensions\ms-python.debugpy-2025.18.0\bundled\libs\debugpy\launche  
r" '55261' '--' 'c:\Users\sринi\OneDrive\Desktop\3.py'  
Enter a number: 123  
Result: Not an Armstrong Number  
PS C:\Users\sринi\OneDrive\Desktop\ai> c;; cd 'c:\Users\sринi\OneDrive\Desktop\ai'; & 'C:\Users\sринi\AppData\Local\Progra  
ms\Python\Python313\python.exe' 'c:\Users\sринi\.vscode\extensions\ms-python.debugpy-2025.18.0\bundled\libs\debugpy\launche  
r' '56697' '--' 'c:\Users\sринi\OneDrive\Desktop\3.py'  
Enter a number: 153  
Result: Armstrong Number  
PS C:\Users\sринi\OneDrive\Desktop\ai>
```

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 \sqrt{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**.



The screenshot shows a VS Code terminal window with the following content:

```
PS C:\Users\srinil\OneDrive\Desktop\ai> cd 'c:\Users\srinil\AppData\Local\Programs\Python\Python313\python.exe' & 'c:\Users\srinil\.vscode\extensions\ms-python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '59346' -- "c:\Users\srinil\OneDrive\Desktop\5.py"  
Enter a number: 64  
64 is not a perfect number.  
PS C:\Users\srinil\OneDrive\Desktop\ai> cd 'c:\Users\srinil\AppData\Local\Programs\Python\Python313\python.exe' & 'c:\Users\srinil\.vscode\extensions\ms-python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '64398' -- "c:\Users\srinil\OneDrive\Desktop\5.py"  
Enter a number: 10  
10 is not a perfect number.  
PS C:\Users\srinil\OneDrive\Desktop\ai> cd 'c:\Users\srinil\AppData\Local\Programs\Python\Python313\python.exe' & 'c:\Users\srinil\.vscode\extensions\ms-python.debugpy-2025.18.0\bundle\libs\debugpy\launcher' '64398' -- "c:\Users\srinil\OneDrive\Desktop\5.py"  
Enter a number: 6  
6 is a perfect number.  
PS C:\Users\srinil\OneDrive\Desktop\ai>
```

```
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()
```

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**.

