

# AIAC LAB ASSIGNMENT 1

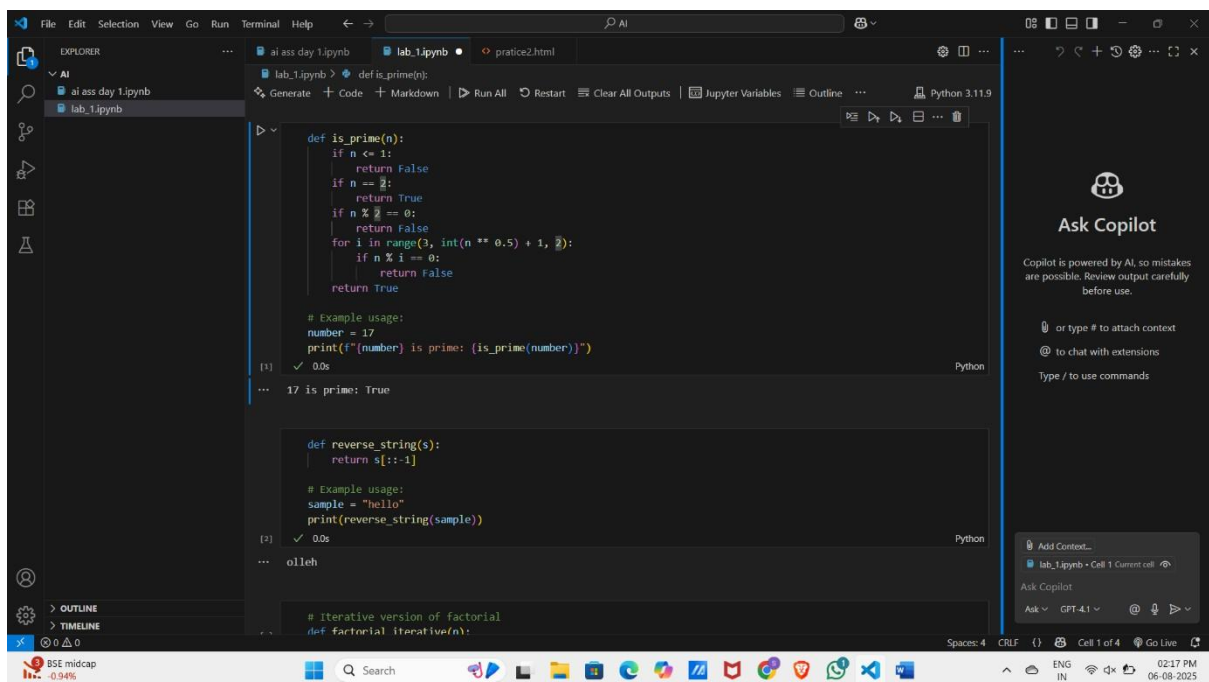
Name : A VARSHITH

H.No : 2403A52024

Batch : 02

**1)Problem statement-**check whether a number is prime or not. Take input dynamically from the user.

**Prompt-** write program to check whether the given number is prime or not



```
def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n ** 0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

# Example usage:
number = 17
print(f"{number} is prime: {is_prime(number)}")
```

[1] ✓ 0.0s Python

... 17 is prime: True

```
def reverse_string(s):
    return s[::-1]

# Example usage:
sample = "hello"
print(reverse_string(sample))
```

[2] ✓ 0.0s Python

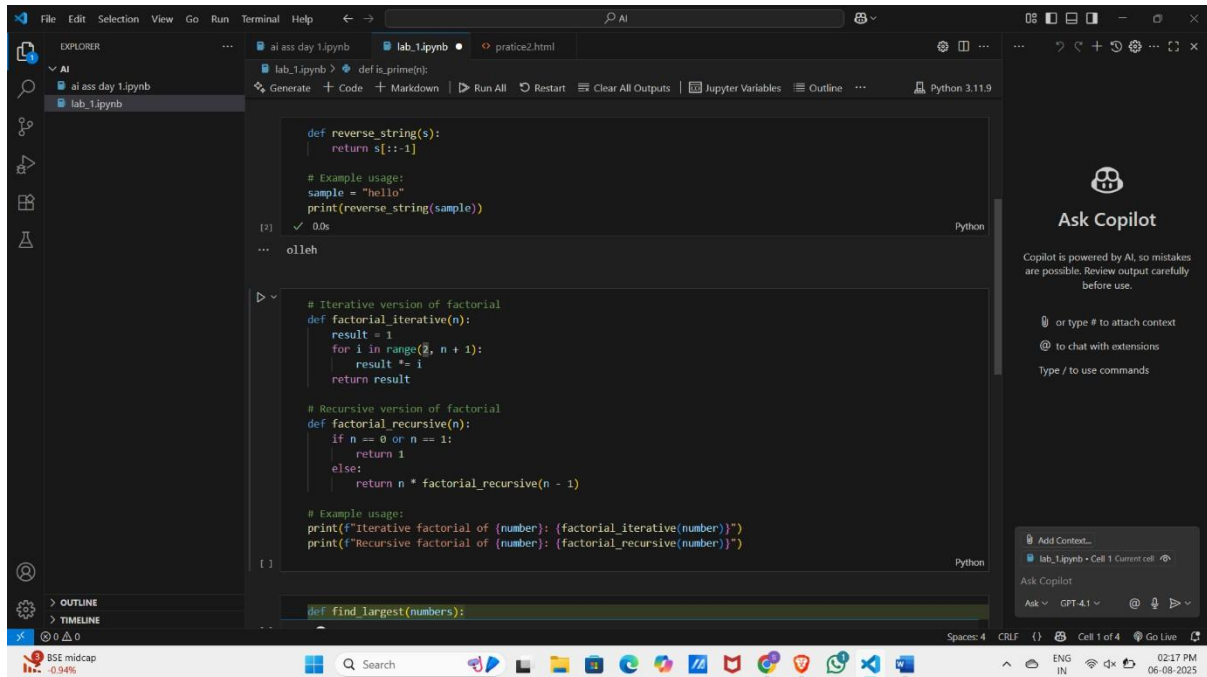
... olleh

```
# Iterative version of factorial
def factorial_iterative(n):
```

**Result-**17 is prime: True

**2)problem statement-** Generate a python code, use co-pilot to print the reverse of a string. Take the string from the user dynamically to reverse it.

**Prompt** – generate a code to reverse the string in dynamically



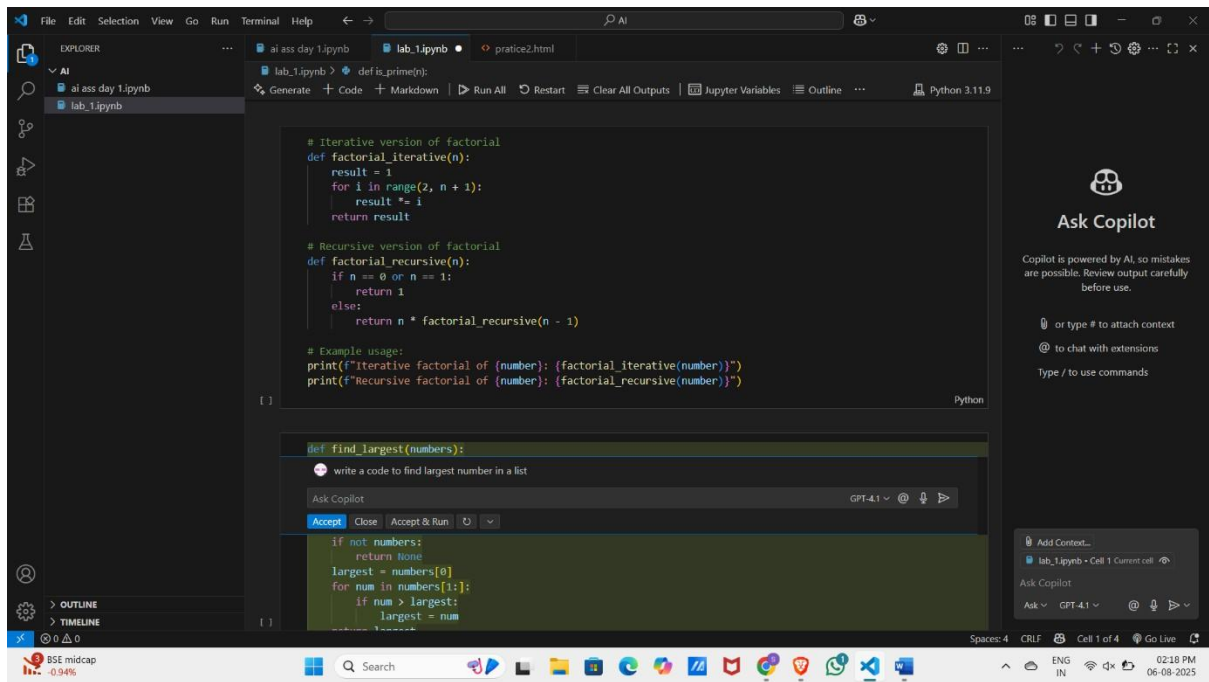
The screenshot shows a Visual Studio Code editor with a dark theme. The Explorer sidebar on the left shows a file named 'lab\_1.ipynb'. The main editor area displays Python code. The first code block defines a function 'reverse\_string(s)' that returns 's[::-1]'. Below it, an example usage is shown: 'sample = "hello"' followed by 'print(reverse\_string(sample))'. The output of this code is 'olleh'. The second code block contains two functions for calculating factorials: 'factorial\_iterative(n)' and 'factorial\_recursive(n)'. The iterative function uses a loop to calculate the factorial, while the recursive function uses a base case and a recursive call. Example usage for both is provided at the bottom. The right sidebar features the 'Ask Copilot' panel, which includes a warning that Copilot is powered by AI and may make mistakes, and options to attach context or chat with extensions. The bottom status bar shows 'Spaces: 4', 'CRLF', and 'Cell 1 of 4'.

```
def reverse_string(s):  
    return s[::-1]  
  
# Example usage:  
sample = "hello"  
print(reverse_string(sample))  
  
[2] ✓ 0.0s  
... olleh  
  
# Iterative version of factorial  
def factorial_iterative(n):  
    result = 1  
    for i in range(2, n + 1):  
        result *= i  
    return result  
  
# Recursive version of factorial  
def factorial_recursive(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * factorial_recursive(n - 1)  
  
# Example usage:  
print(f"Iterative factorial of {number}: {factorial_iterative(number)}")  
print(f"Recursive factorial of {number}: {factorial_recursive(number)}")  
  
[ ]  
  
def find_largest(numbers):
```

**Result-** olleh

**3)Problem statement-**Generate recursive and iterative versions of a factorial function using comments

**Prompt-**write a program recursive and iterative versions of a factorial function using comments

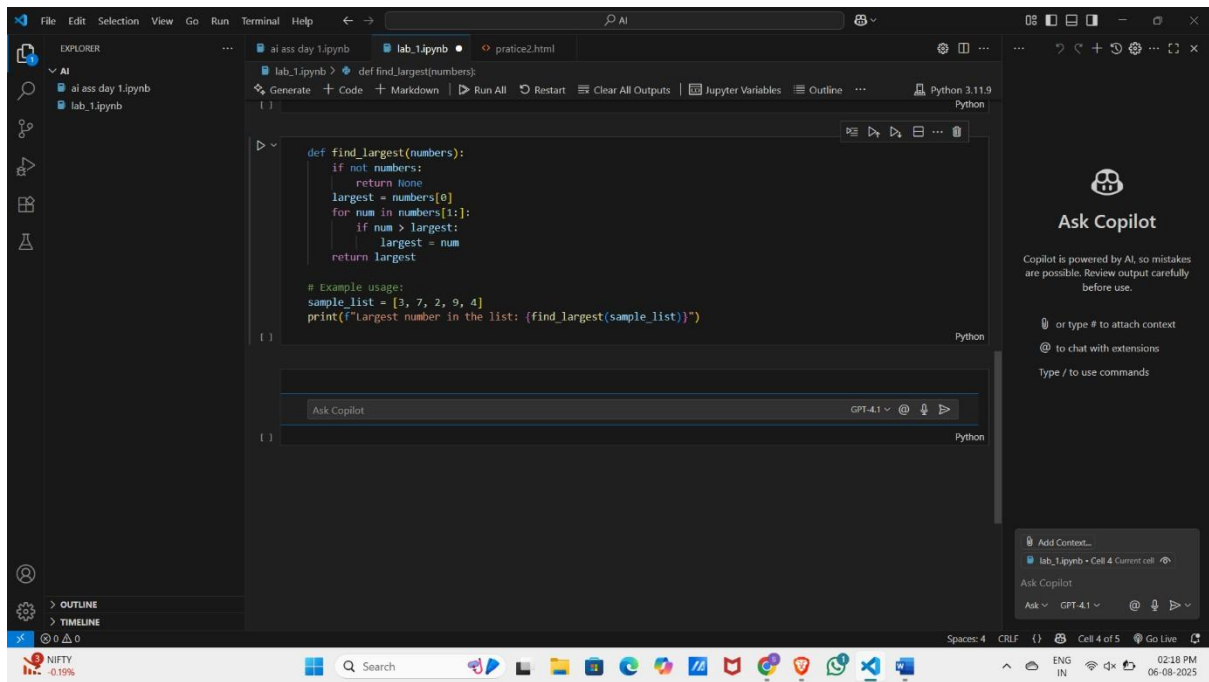


**Result**-Iterative factorial of 17: 355687428096000

Recursive factorial of 17: 355687428096000

**4)problem statement-** Generate a python code, use co-pilot to check the largest number in the list. Take input dynamically from the user.

**Prompt**-write a code to find largest number in a list



**Result-**Largest number in the list: 9