AI Assisted Problem Solving Using Python

NAME : SYEDA FARHATH UMAMA

HT.NO : 2503B05143

Program Name: M.Tech (CSE)

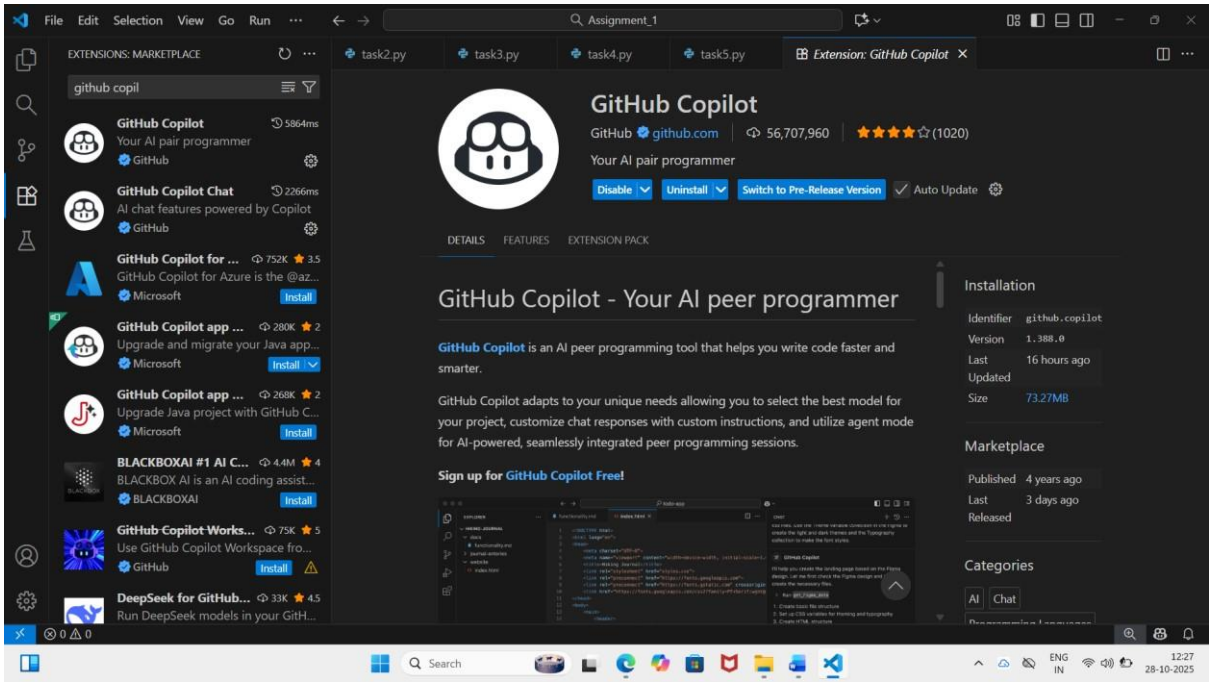
Lab 1: Environment Setup – GitHub Copilot and VS Code Integration

Task Description#1

● Install and configure GitHub Copilot in VS Code.

Expected Output#1

● Install and configure GitHub Copilot in VS Code.



Task Description#2

● Use Copilot to generate a is_prime() Python function.

Expected Output#2

● Function to check primality with correct logic.

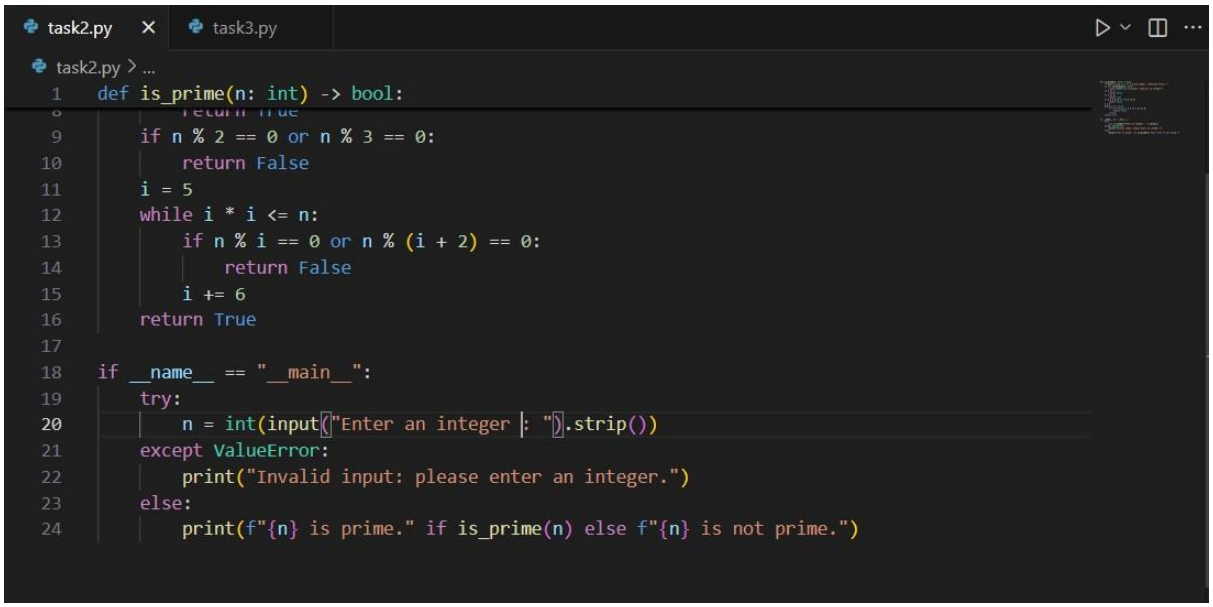
Prompt_1:

Create a function named is_prime() to check primality.

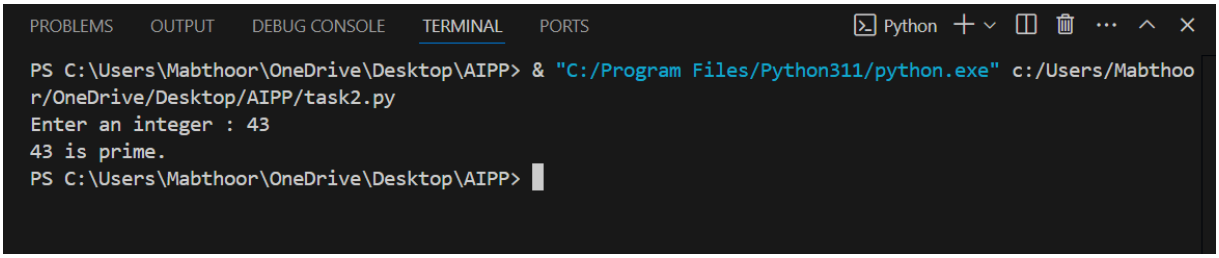
Prompt_2:

Now, update this code where the user can take the input from keyboard.

CODE:



OUTPUT:



Task Description#3

- Write a comment like # Function to reverse a string and use Copilot to generate the function.

Expected Output#3

- Auto-completed reverse function

Prompt_1:

Create a function to reverse a string and provide the auto completed reverse function output.

CODE:

task2.py task3.py

task3.py > ...

```python
# Function to reverse a string
def reverse_string(s: str) -> str:
    """Return a new string which is the reverse of s."""
    if not isinstance(s, str):
        raise TypeError("reverse_string() requires a string")
    return s[::-1]

def reverse_string_io() -> str:
    """Read a string from input, print its reverse, and return it."""
    s = input("Enter a string: ")
    rev = reverse_string(s)
    print(f"Reversed string: {rev}")
    return rev

if __name__ == "__main__":
    reverse_string_io()
```

OUTPUT:

```
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP> & "C:/Program Files/Python311/python.exe" c:/Users/Mabthoo
r/OneDrive/Desktop/AIPP/task3.py
Enter a string: python
Reversed string: nohtyp
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP>
```

Task Description#4

● Generate both recursive and iterative versions of a factorial function using comments..

Expected Output#4

● Two working factorial implementations

Prompt_1:

Generate both recursive and iterative versions of a factorial function where the output needs to return two working factorial implementations.

CODE:

task4.py > ...

```python
# Recursive factorial
def factorial_recursive(n: int) -> int:
    """Return n! using recursion."""
    if not isinstance(n, int):
        raise TypeError("factorial_recursive() requires an integer")
    if n < 0:
        raise ValueError("factorial_recursive() requires a non-negative integer")
    if n == 0 or n == 1:
        return 1
    return n * factorial_recursive(n - 1)

# Iterative factorial
def factorial_iterative(n: int) -> int:
    """Return n! using an iterative approach."""
    if not isinstance(n, int):
        raise TypeError("factorial_iterative() requires an integer")
    if n < 0:
        raise ValueError("factorial_iterative() requires a non-negative integer")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

def factorial_io() -> None:
    """Read an integer from input, print factorials computed iteratively and recursively."""
    try:
        n = int(input("Enter a non-negative integer: ").strip())
        if n < 0:
            print("Invalid input: please enter a non-negative integer.")
            return
    except ValueError:
        print("Invalid input: please enter an integer.")
        return

    print(f"Iterative: {n}! = {factorial_iterative(n)}")
    print(f"Recursive: {n}! = {factorial_recursive(n)}")
```

OUTPUT:

```
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP> & "C:/Program Files/Python311/python.exe" c:/Users/Mabthoor/
OneDrive/Desktop/AIPP/task4.py
Enter a non-negative integer: 6
Iterative: 6! = 720
Recursive: 6! = 720
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP>
```

Task Description#5

● Use Copilot to find the largest number in a list. Assess code quality and efficiency.

Expected Output#5

● A valid function with your review

Prompt_1:

Create a function to find the largest number in a list where the list need to be taken from keyboard.

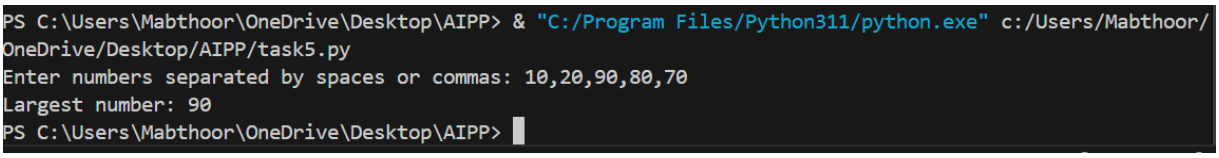
CODE:

```python
def find_max(numbers: list[float]) -> float:
    """Return the largest number in a non-empty list of numbers."""
    if not isinstance(numbers, list):
        raise TypeError("find_max() requires a list")
    if not numbers:
        raise ValueError("find_max() requires a non-empty list")
    max_val = numbers[0]
    for x in numbers[1:]:
        if x > max_val:
            max_val = x
    return max_val

def find_max_io() -> None:
    """Read a list of numbers from keyboard, print the largest, and return nothing."""
    s = input("Enter numbers separated by spaces or commas: ").strip()
    if not s:
        print("No input provided.")
        return
    import re
    parts = [p for p in re.split(r"[,\s]+", s) if p != ""]
    try:
        nums = [float(p) for p in parts]
    except ValueError:
        print("Invalid input: please enter only numbers separated by spaces or commas.")
        return
    try:
        largest = find_max(nums)
    except ValueError as e:
        print(e)
        return
    # Print as int when the number is an integer value
    if largest.is_integer():
        print(f"Largest number: {int(largest)}")
    else:
        print(f"Largest number: {largest}")

if __name__ == "__main__":
    find_max_io()
```

OUTPUT:



```
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP> & "C:/Program Files/Python311/python.exe" c:/Users/Mabthoor/
OneDrive/Desktop/AIPP/task5.py
Enter numbers separated by spaces or commas: 10,20,90,80,70
Largest number: 90
PS C:\Users\Mabthoor\OneDrive\Desktop\AIPP>
```