

## AI - ASSISTED CODING LAB ASSIGNMENT=1.3

NAME: POTHUNURI AKSHITHA

ROLL NO: 2403A510C2

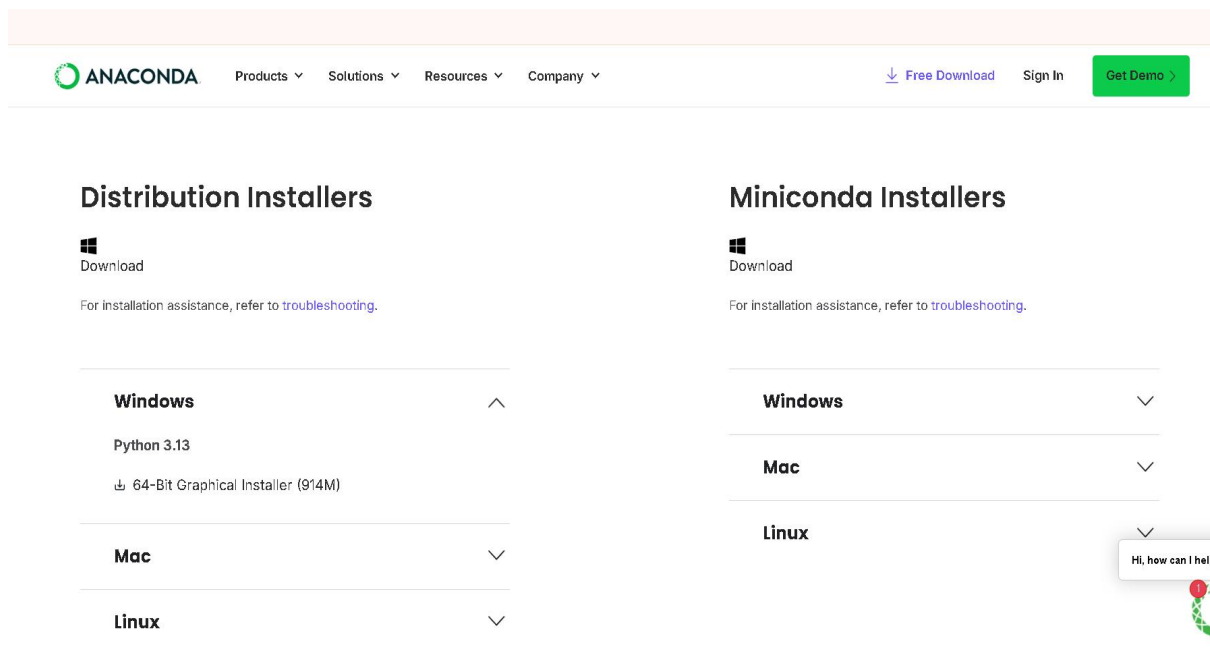
BATCH: 05

BRANCH: CSE

SUB: AI ASSISTED CODING

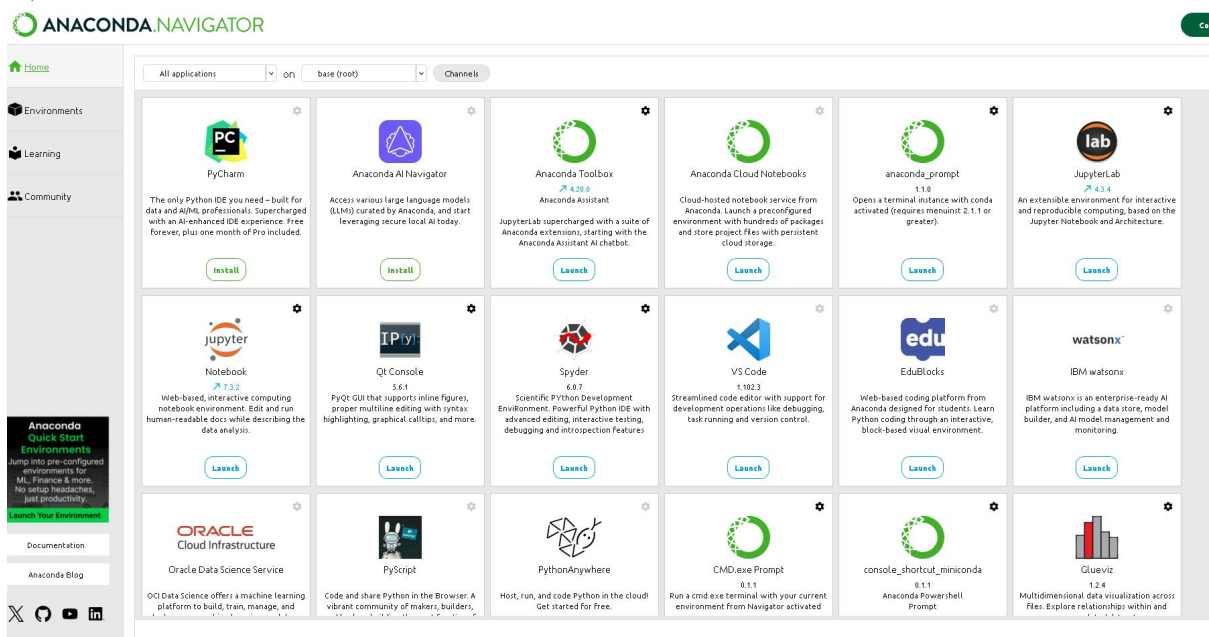
TASK - 01

Screen shot-1

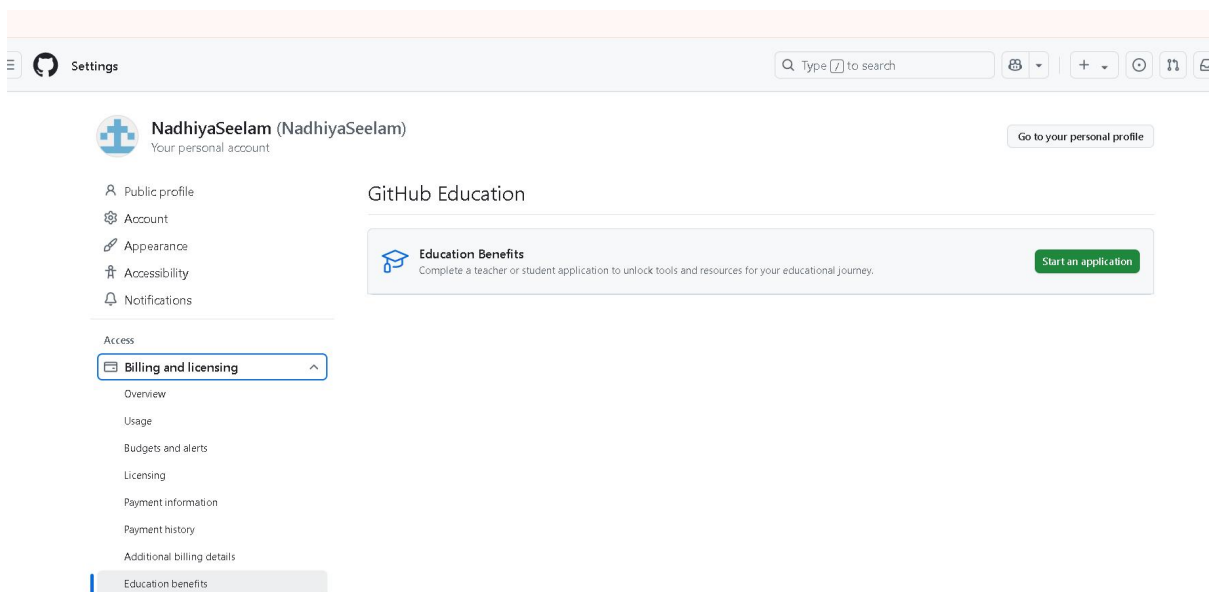


The screenshot displays the Anaconda website's installer section. At the top, the Anaconda logo is on the left, and navigation links for Products, Solutions, Resources, and Company are in the center. On the right, there are links for 'Free Download', 'Sign In', and a green 'Get Demo' button. Below the navigation bar, the page is divided into two main columns: 'Distribution Installers' and 'Miniconda Installers'. Each column features a 'Download' button with a Windows icon and a link to 'troubleshooting' for installation assistance. Under 'Distribution Installers', there are three expandable sections: 'Windows' (showing 'Python 3.13' and a '64-Bit Graphical Installer (914M)' download link), 'Mac', and 'Linux'. Similarly, the 'Miniconda Installers' column has expandable sections for 'Windows', 'Mac', and 'Linux'. A small chatbot window in the bottom right corner says 'Hi, how can I help?' with a snake icon.

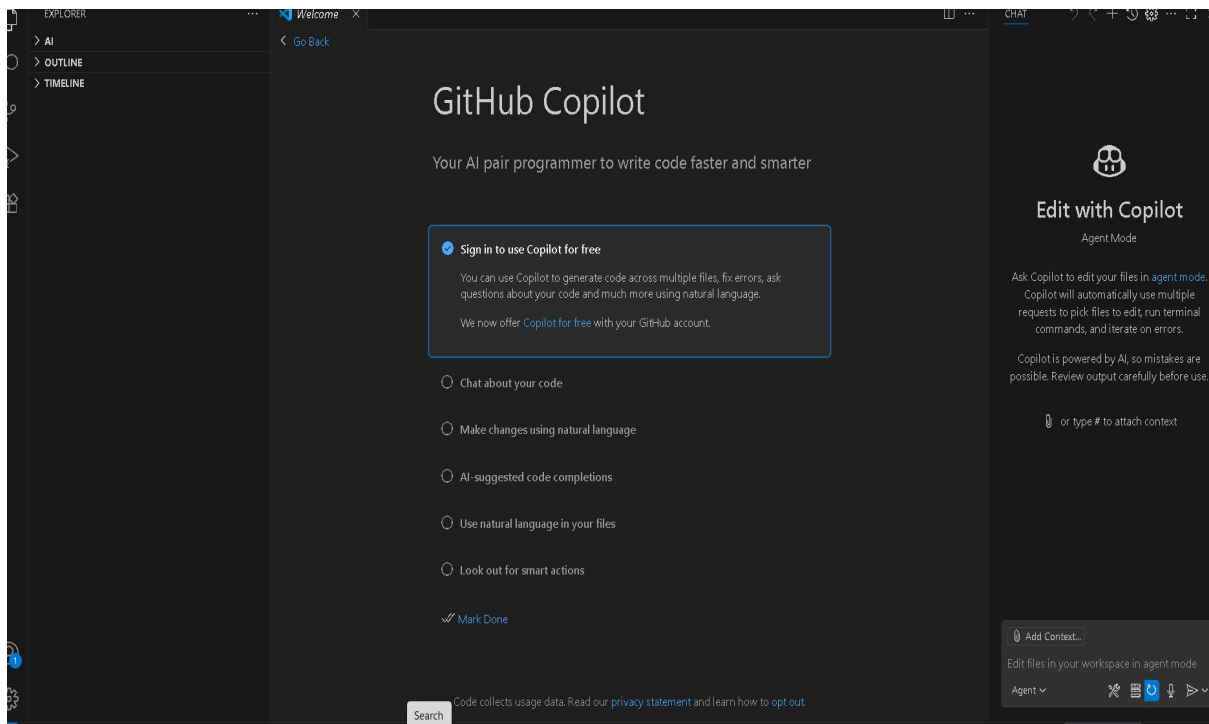
Screen shot-2



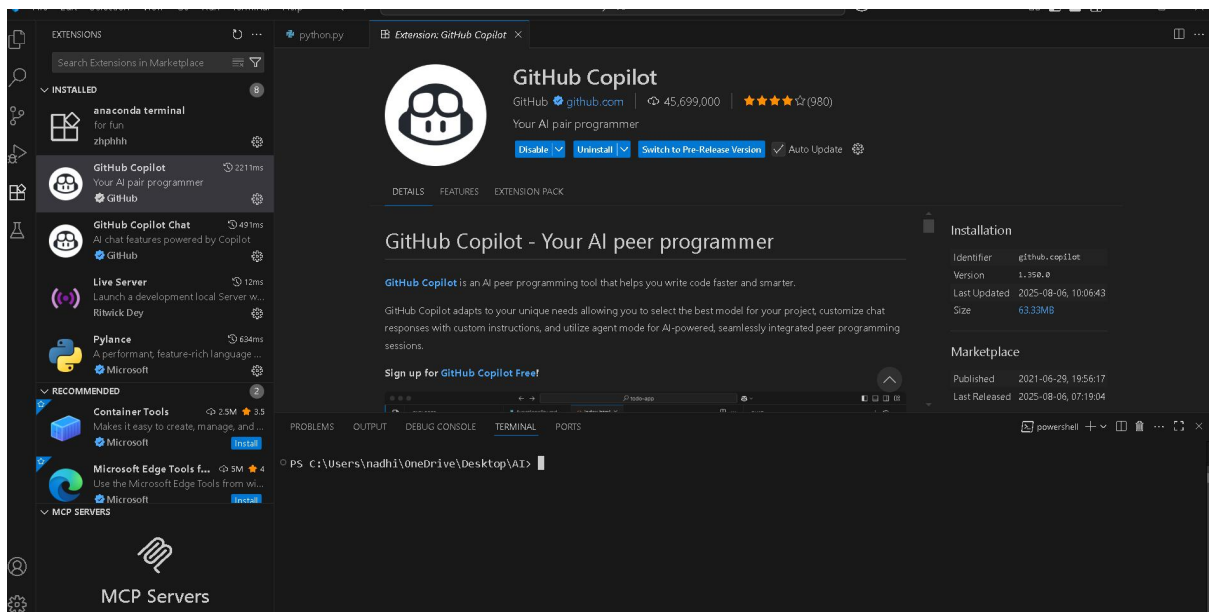
Screen shot-3



Screen shot-4



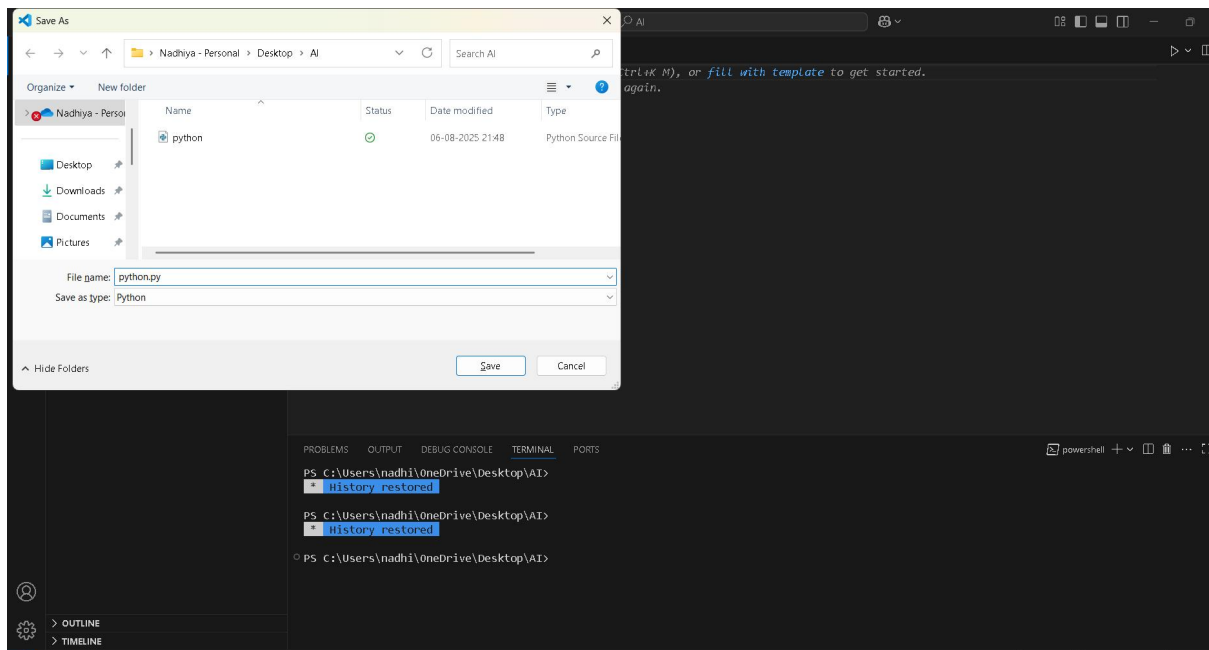
Screen shot-5



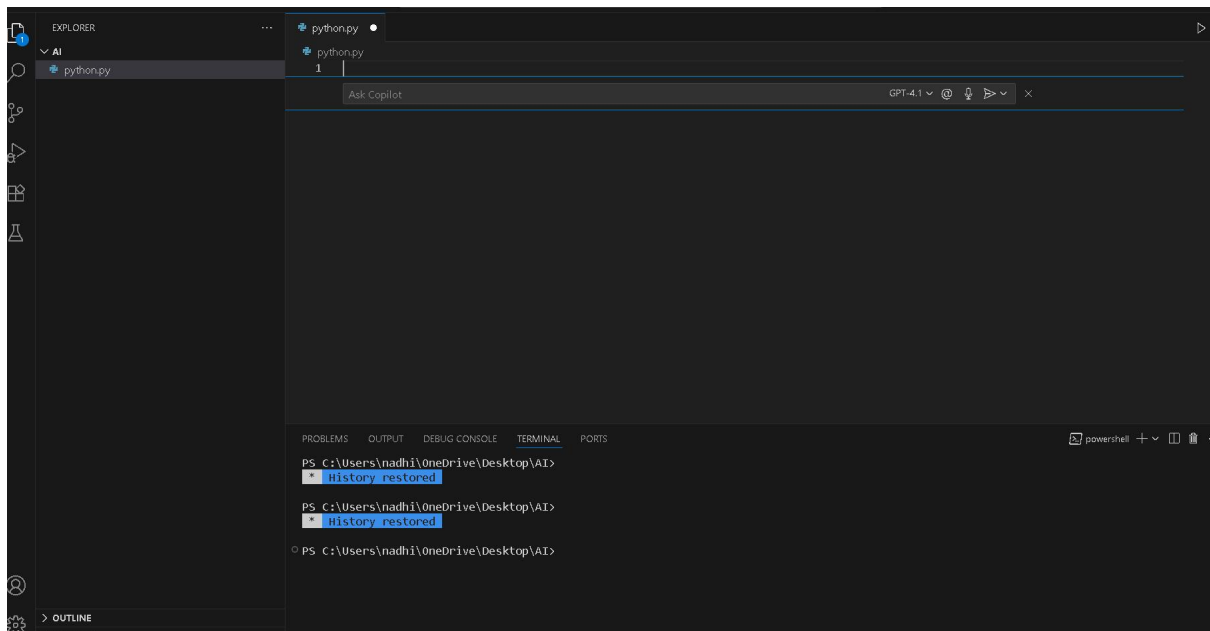
Screen shot-6



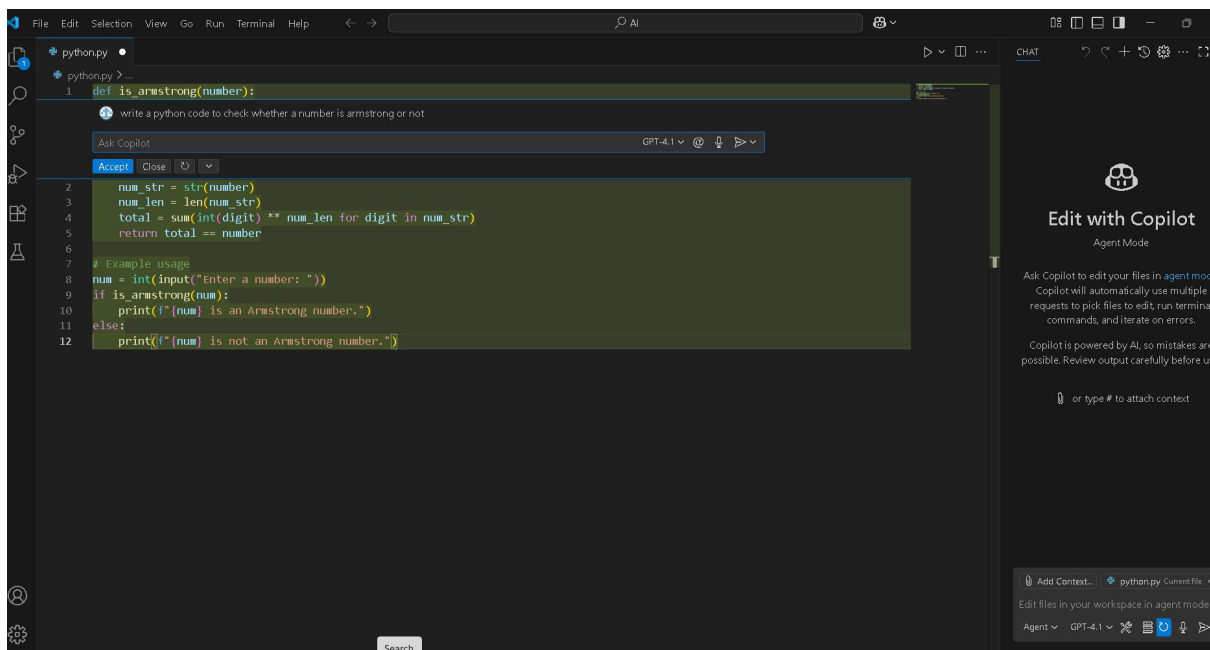
Screen shot-7



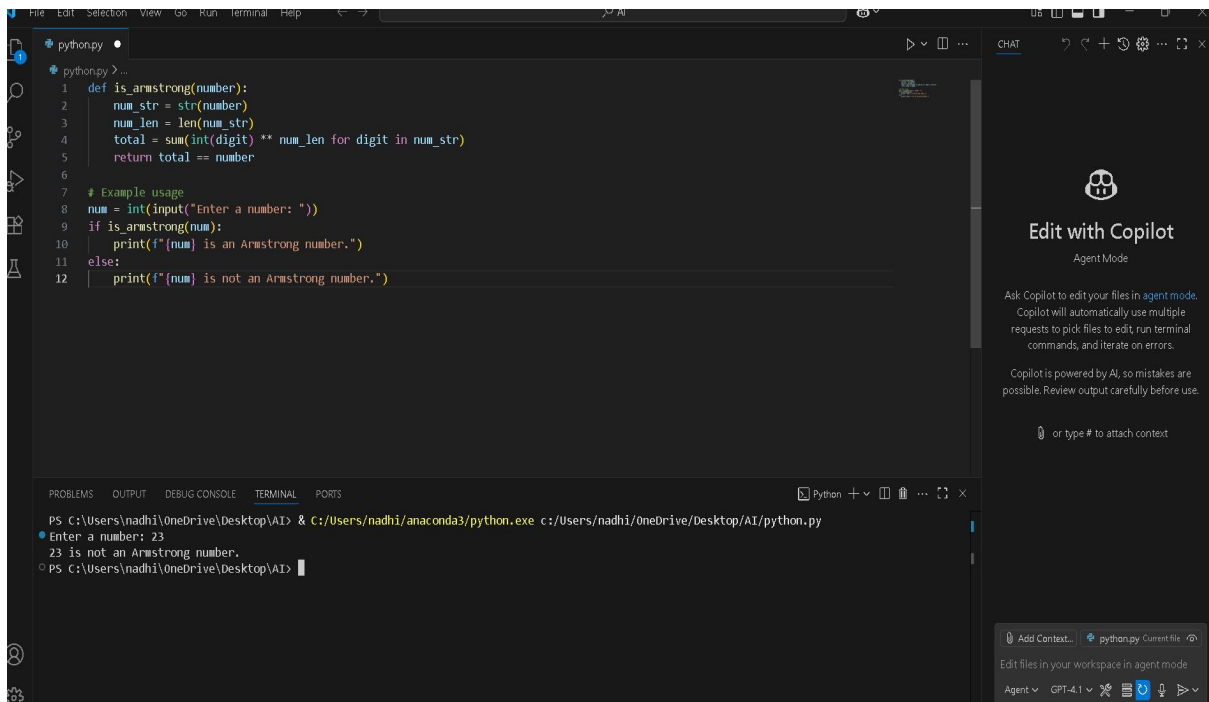
Screen shot-8



Screen shot-9



Screen shot-9



## TASK – 02

**Prompt :** Write a python code to check whether a number is prime or not.

```
... first_code.html python.py
python.py
1 def is_prime(n):
2     if n <= 1:
3         return False
4     if n == 2:
5         return True
6     if n % 2 == 0:
7         return False
8     for i in range(3, int(n ** 0.5) + 1, 2):
9         if n % i == 0:
10            return False
11    return True
12
13 # Example usage
14 num = int(input("Enter a number: "))
15 if is_prime(num):
16     print(f"{num} is a prime number.")
17 else:
18     print(f"{num} is not a prime number.")
```

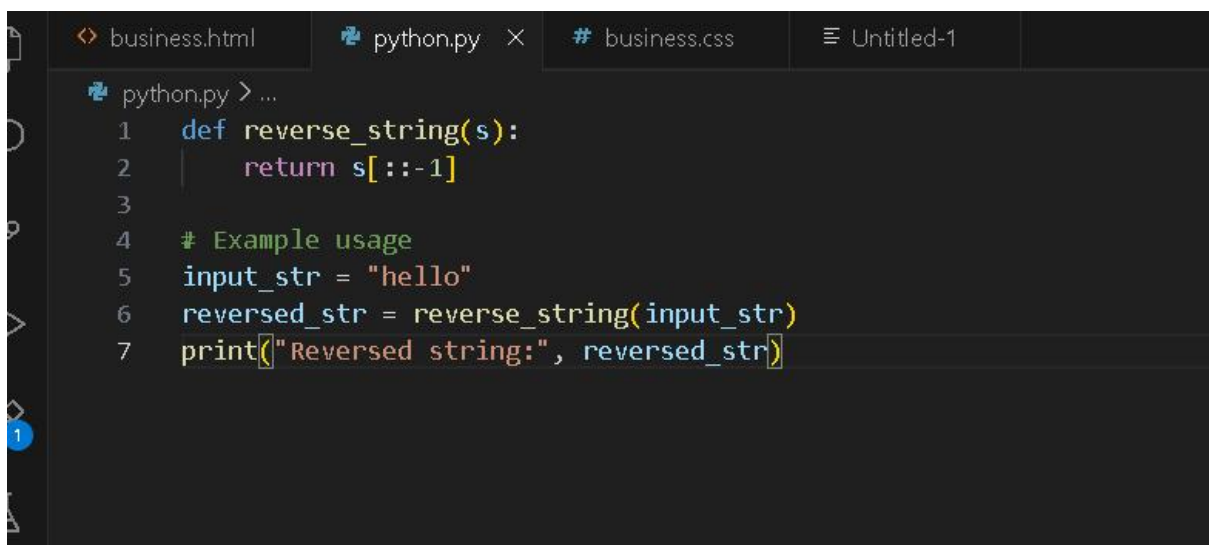
*Output: 2 is prime number*

*Explanation:*

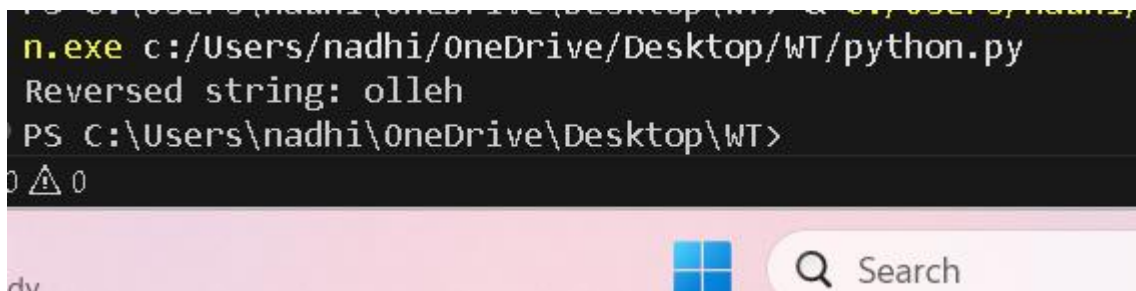
- The code checks if a given number is prime.
- A prime number is a number greater than 1 that has no divisors other than 1 and itself.
- The code usually:
  - Returns False if the number is less than or equal to 1.
  - Loops from 2 up to the square root of the number.

- If the number is divisible by any of these, it returns False.
- If no divisors are found, it returns True.

### Task-3:



```
python.py > ...
1  def reverse_string(s):
2      return s[::-1]
3
4  # Example usage
5  input_str = "hello"
6  reversed_str = reverse_string(input_str)
7  print("Reversed string:", reversed_str)
```



```
n.exe c:/Users/nadhi/OneDrive/Desktop/WT/python.py
Reversed string: olleh
PS C:\Users\nadhi\OneDrive\Desktop\WT>
```

### Explanation:

- The function reverse\_string(s) takes a string s as input and returns its reverse using slicing (s[::-1]).
- The example usage sets input\_str to "hello".



- It calls reverse\_string(input\_str), which returns "olleh", and stores it in reversed\_str.
- Finally, it prints Reversed string: olleh to the console

### Task-O4:

```
python.py > ...
1  # Recursive version of factorial
2  def factorial_recursive(n):
3      """
4      Calculate factorial of n recursively.
5      """
6      if n == 0 or n == 1:
7          return 1
8      else:
9          return n * factorial_recursive(n - 1)
10
11 # Iterative version of factorial
12 def factorial_iterative(n):
13     """
14     Calculate factorial of n iteratively.
15     """
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21 # Example usage
22 if __name__ == "__main__":
23     num = 5
24     print("Recursive:", factorial_recursive(num)) # Output: 120
25     print("Iterative:", factorial_iterative(num)) # Output: 120
```

```
PS C:\Users\nadhi\OneDrive\Desktop\WT> & C:/Users/nadhi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nadhi/OneDrive/Desktop/WT/python.py
Recursive: 120
Iterative: 120
```

### Explanation:

- This function calculates the factorial of n using recursion.
  - If n is 0 or 1, it returns 1 (base case).
  - Otherwise, it returns  $n * \text{factorial\_recursive}(n - 1)$ .

- **factorial\_iterative(n):**

This function calculates the factorial of n using a loop.

- It initializes result to 1.
- Then multiplies result by each number from 2 up to n.

- **Example usage:**

- If the script is run directly, it sets num = 5.
- It prints the factorial of 5 using both the recursive and iterative functions.
- Both methods output 120.

**Task-05:**

```
business.html python.py # business.css Untitled-1
python.py > ...
1 def find_largest(numbers):
2     if not numbers:
3         return None # Return None if the list is empty
4     largest = numbers[0]
5     for num in numbers[1:]:
6         if num > largest:
7             largest = num
8     return largest
9
10 # Example usage
11 nums = [3, 7, 2, 9, 4]
12 print("Largest number:", find_largest(nums))
```

```
PS C:\Users\nadhi\OneDrive\Desktop\WT> & C:/Users/nadhi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nadhi/OneDrive/Desktop/WT/python.py
Largest number: 9
PS C:\Users\nadhi\OneDrive\Desktop\WT>
```

## Explanation:

- **`find_largest(numbers):`**

This function takes a list of numbers and returns the largest value.

- If the list is empty, it returns `None`.
- It starts by assuming the first number is the largest.
- It then loops through the rest of the list, updating largest if it finds a bigger number.

- Finally, it returns the largest number found.
- **Example usage:**
  - A list `nums = [3, 7, 2, 9, 4]` is defined.
  - The function is called with this list, and the result is printed.
  - Output: Largest number: 9