



**School of Computer Science and AI, SR University**

## **AI Assisted Problem Solving Using Python**

**AcademicYear: 2025-2026**

**Assignment Type: Lab**  
Assig\_No. 1

Name: **Kassim Sesay**

Enrol No. **2503B05101**

Course: Msc Computer Science

**Submitted to:**

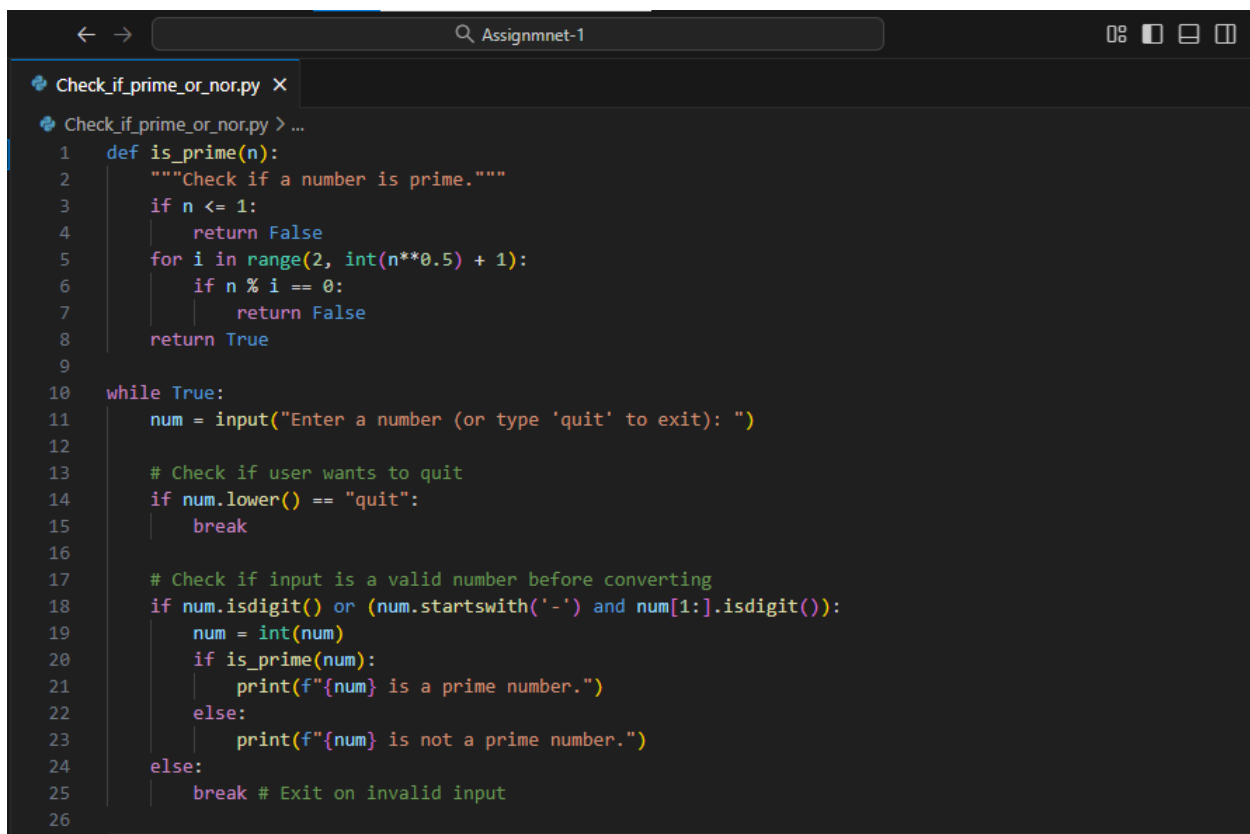
**Name: Venkataramana Veeramsetty**

**Title: Course Instructor**

Date: 27 Oct 2025

Q 2: Use Copilot to generate a `is_prime()` Python function.

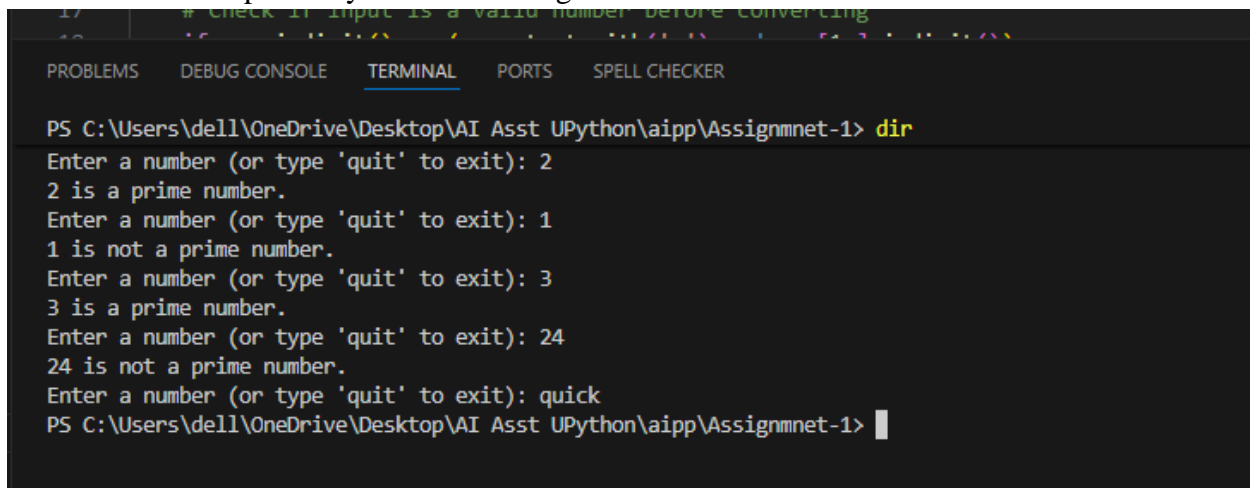
Code:



```
1 def is_prime(n):
2     """Check if a number is prime."""
3     if n <= 1:
4         return False
5     for i in range(2, int(n**0.5) + 1):
6         if n % i == 0:
7             return False
8     return True
9
10 while True:
11     num = input("Enter a number (or type 'quit' to exit): ")
12
13     # Check if user wants to quit
14     if num.lower() == "quit":
15         break
16
17     # Check if input is a valid number before converting
18     if num.isdigit() or (num.startswith('-') and num[1:].isdigit()):
19         num = int(num)
20         if is_prime(num):
21             print(f"{num} is a prime number.")
22         else:
23             print(f"{num} is not a prime number.")
24     else:
25         break # Exit on invalid input
26
```

Output screenshot:

Function to check primality with correct logic.



```
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1> dir
Enter a number (or type 'quit' to exit): 2
2 is a prime number.
Enter a number (or type 'quit' to exit): 1
1 is not a prime number.
Enter a number (or type 'quit' to exit): 3
3 is a prime number.
Enter a number (or type 'quit' to exit): 24
24 is not a prime number.
Enter a number (or type 'quit' to exit): quick
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1>
```

Q 3: Write a comment like # Function to reverse a string and use Copilot to generate the function.

Code:

```
Check_if_prime_or_nor.py X
Check_if_prime_or_nor.py > ...
1  def is_prime(n):
2      """Check if a number is prime."""
3      if n <= 1:
4          return False
5      for i in range(2, int(n**0.5) + 1):
6          if n % i == 0:
7              return False
8      return True
9
10 while True:
11     num = input("Enter a number (or type 'quit' to exit): ")
12
13     # Check if user wants to quit
14     if num.lower() == "quit":
15         break
16
17     # Check if input is a valid number before converting
18     if num.isdigit() or (num.startswith('-') and num[1:].isdigit()):
19         num = int(num)
20         if is_prime(num):
21             print(f"{num} is a prime number.")
22         else:
23             print(f"{num} is not a prime number.")
24     else:
25         break # Exit on invalid input
26
27
```

Output: Auto-completed reverse function

```
PROBLEMS  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1> dir
Enter a string to reverse:
SR University
The original string is: SR University
The reversed string is: ytisrevinU RS
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1> |
```

Q4: Generate both recursive and iterative versions of a factorial function using comments..

Code:

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

Output: Two working factorial implementations

```
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aiapp\Assignmnet-1> python factorial_function.py
Recursive: 120
Iterative: 120
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aiapp\Assignmnet-1> 
```

Q 5: Use Copilot to find the largest number in a list. Assess code quality and efficiency

Code:

```
largest_number.py X
largest_number.py > ...
1  def find_largest_number(numbers):
2      """
3      Returns the largest number in a list.
4      Args:
5      |   numbers (list): List of numeric values.
6      Returns:
7      |   The largest number in the list, or None if the list is empty.
8      """
9      if not numbers:
10         return None
11     return max(numbers)
12
13 # Example usage:
14 if __name__ == "__main__":
15     sample_list = [3, 5, 2, 8, 1]
16     largest = find_largest_number(sample_list)
17     print(f"The largest number in {sample_list} is {largest}")
18
19 # Review:
20 # - Uses Python's built-in max() function, which is efficient (O(n) time).
21 # - Handles empty lists gracefully by returning None.
22 # - Code is concise, readable, and leverages standard library features.
23 # - For very large lists, this is as efficient as possible in Python.
```

Output: A valid function with your review

```
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1> python largest_number.py
The largest number in [3, 5, 2, 8, 1] is 8
PS C:\Users\dell\OneDrive\Desktop\AI Asst UPython\aipp\Assignmnet-1> |
```