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:

Assignmnet-1

Check_if_prime_or_nor.py

```python
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

while True:
    num = input("Enter a number (or type 'quit' to exit): ")

    # Check if user wants to quit
    if num.lower() == "quit":
        break

    # Check if input is a valid number before converting
    if num.isdigit() or (num.startswith('-') and num[1:].isdigit()):
        num = int(num)
        if is_prime(num):
            print(f"{num} is a prime number.")
        else:
            print(f"{num} is not a prime number.")
    else:
        break # Exit on invalid input
```

Output screenshot:

Function to check primality with correct logic.

PROBLEMS DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER

```
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

```python
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

while True:
    num = input("Enter a number (or type 'quit' to exit): ")

    # Check if user wants to quit
    if num.lower() == "quit":
        break

    # Check if input is a valid number before converting
    if num.isdigit() or (num.startswith('-') and num[1:].isdigit()):
        num = int(num)
        if is_prime(num):
            print(f"{num} is a prime number.")
        else:
            print(f"{num} is not a prime number.")
    else:
        break # Exit on invalid input
```

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

```python
# Recursive version of factorial
def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)

# Iterative version of factorial
def factorial_iterative(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Example usage:
if __name__ == "__main__":
    num = 5
    print("Recursive:", factorial_recursive(num))  # Output: 120
    print("Iterative:", factorial_iterative(num))  # Output: 120
```

Output: Two working factorial implementations

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

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

Code:

largest_number.py

largest_number.py > ...

```python
def find_largest_number(numbers):
    """
    Returns the largest number in a list.
    Args:
        numbers (list): List of numeric values.
    Returns:
        The largest number in the list, or None if the list is empty.
    """
    if not numbers:
        return None
    return max(numbers)

# Example usage:
if __name__ == "__main__":
    sample_list = [3, 5, 2, 8, 1]
    largest = find_largest_number(sample_list)
    print(f"The largest number in {sample_list} is {largest}")
# Review:
# - Uses Python's built-in max() function, which is efficient (O(n) time).
# - Handles empty lists gracefully by returning None.
# - Code is concise, readable, and leverages standard library features.
# - 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>
```