SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 4

Program	**: B. Tech (CSE)**
Specialization	**: CSE**
Course Title	**: AI Assisted Coding**
Course Code	**: 23CS201PC302**
Semester	**: 3 -2**
Academic Session	**: 2025-2026**
Name of Student	**: Eshwar**
Enrollment No.	**: 2403a51l26**
Batch No.	**: 51**
Date	**:20-01-2026**

Submission Instructions:

(All instructions should be followed strictly to avoid deduction of marks)

1. Use the same file to complete the assignment and don't change the settings.
2. Minimum 10 screen shots of your account should be taken to showcase your work.
3. **File Format:**
   - Submit your assignment as a PDF document (pdf). Ensure the file is named according to the following convention:

     `BNo_StudentName_AI_Coding_A1.`

     `Sample`: **B10_Rohit_22A523421_A1**
4. Fill all the entries mentioned on top section.
5. Mention your AWS Academy Virtual Lab Account details as shown in the next page.
6. **Don't write on this page.**
7. All answers should be answered from next page only.

Lab assignment 4.2:

Prompt: Zero-shot: Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime:

ai_coding.py > ...

```python
#  Zero-shot prime number check

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

# Sample test
num = 17
print(f"Is {num} prime? {is_prime(num)}")
```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe" c
Is 17 prime? True
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

```python
#  # Zero-shot AI-generated function to check prime numbers

def is_prime(n):
    """
    Checks if a given number is prime.
    Returns True if prime, False otherwise.
    """
    if n <= 1:
        return False  # Numbers <=1 are not prime
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False  # Found a divisor, not prime
    return True  # No divisors found, number is prime

# Example usage
print(is_prime(7))   # Output: True
print(is_prime(12))  # Output: False

```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe"
True
False
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

- **The function checks if the number is less than or equal to 1 (not prime).**
- **then tests divisibility from 2 up to the square root of n to improve efficiency.**
- **If no divisors are found, the function returns True, confirming the number is prime.**

Task 2:

Prompt:

One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a

function that calculates the sum of elements in a list.

ai_coding.py

```python
# One-shot AI-generated function to sum list elements

def sum_list_elements(lst):
    """
    Returns the sum of all elements in a given list.
    """
    total = 0
    for num in lst:
        total += num
    return total

# Example usage
example_input = [1, 2, 3, 4]
print("Input:", example_input)
print("Sum of elements:", sum_list_elements(example_input))

```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe"
Input: [1, 2, 3, 4]
Sum of elements: 10
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

- **The function sum_list_elements initializes a total at 0.**
- **It iterates over each element in the list, adding it to the total.**
- **Finally, it returns the total sum, producing correct output guided by the one-shot example**

Task 3:

Prompt:

Few-shot: Give 2–3 examples to create a function that extracts digits from an alphanumeric string

ai_coding.py

```python
# Few-shot AI-generated function to extract digits from alphanumeric string

def extract_digits(text):
    """
    Returns a string containing only the digits from the input alphanumeric string.
    """
    digits = ""
    for char in text:
        if char.isdigit():
            digits += char
    return digits

# Sample tests
print("Input: 'abc123' -> Output:", extract_digits("abc123"))
print("Input: 'a1b2c3' -> Output:", extract_digits("a1b2c3"))
print("Input: '2023xyz' -> Output:", extract_digits("2023xyz"))
```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe"
Input: 'abc123' -> Output: 123
Input: 'a1b2c3' -> Output: 123
Input: '2023xyz' -> Output: 2023
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

- **The function extract_digits iterates through each character of the input string.**
- **It checks if the character is a digit using isdigit() and appends it to a result string.**
- **The function returns only the digits, accurately extracting numbers from any alphanumeric input.**

Task 4:

Prompt:

Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string

```python
# Few-shot AI-generated function to count vowels

def count_vowels(text):
    """
    Returns the number of vowels in the input string.
    """
    vowels = "aeiouAEIOU"
    count = 0
    for char in text:
        if char in vowels:
            count += 1
    return count

# Sample tests
print("Input: 'hello' -> Vowels:", count_vowels("hello"))
print("Input: 'world' -> Vowels:", count_vowels("world"))
print("Input: 'OpenAI' -> Vowels:", count_vowels("OpenAI"))
```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe" c
Input: 'hello' -> Vowels: 2
Input: 'world' -> Vowels: 1
Input: 'OpenAI' -> Vowels: 4
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

- **The function count_vowels checks each character in the string to see if it is a vowel.**
- **It increments a counter for every vowel found and returns the total count.**
- **Providing examples in one-shot or few-shot prompts helps AI understand the expected input-output format, improving accuracy.**

Task 5:

Prompt:

Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in min() function.

ai_coding.py

```python
# Few-shot AI-generated function to find the minimum of three numbers

def find_min(a, b, c):
    """
    Returns the smallest of three numbers without using min().
    """
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c

# Sample tests
print("Input: (5, 2, 8) -> Minimum:", find_min(5, 2, 8))
print("Input: (1, 7, 3) -> Minimum:", find_min(1, 7, 3))
print("Input: (10, 4, 6) -> Minimum:", find_min(10, 4, 6))
print("Input: (7, 7, 2) -> Minimum:", find_min(7, 7, 2))

```

Output:

```
PS C:\Users\Eshwar\OneDrive\Desktop\python> & "C:\Program Files\Python314\python.exe" c:
Input: (5, 2, 8) -> Minimum: 2
Input: (1, 7, 3) -> Minimum: 1
Input: (10, 4, 6) -> Minimum: 4
Input: (7, 7, 2) -> Minimum: 2
PS C:\Users\Eshwar\OneDrive\Desktop\python>
```

- **The function compares the three numbers using conditional statements.**
- **It returns the smallest number without relying on Python's built-in min() function.**
- **Few-shot examples guide the AI to handle all possible scenarios, ensuring correct output for any combination.**