System Requirements Specification Index

For

Sequence Generator Console Application

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Sequence Generator Console
System Requirements Specification

1 PROJECT ABSTRACT

Numbspace Labs, a data analysis firm, requires a numerical sequence generation tool for their research projects. The company needs a Python console application that can efficiently generate custom number sequences, perform mathematical operations on these sequences, and output the results in various formats. This tool will be used by their analysts to quickly generate test data, identify patterns, and validate mathematical models across multiple research domains.

BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. Application must demonstrate the four ways to use range() function 2. System must generate different types of sequences (basic, custom, stepped, reverse) 3. Program must analyze basic sequence properties (sum, count, etc.) 4. System must handle input validation and error conditions

2 CONSTRAINTS

2.1 INPUT REQUIREMENTS

1. Sequence Parameters:
   - Must accept parameters for range() function
   - Parameters include: start, end, step values
   - Example: For range(5, 15, 2) - start=5, end=15, step=2
2. Sequence Types:

- - Basic sequence: range(n)
- - Custom sequence: range(start, end)
- - Stepped sequence: range(start, end, step)
- - Reverse sequence: range(start, end, -step)

2.2 Calculation Constraints

1. Basic Sequence Generation:
    - Must use range() with single parameter
    - Example: range(10) to generate [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    - Must return list of generated sequence values
2. Custom Range Iteration:
    - Must use range() with start, stop parameters
    - Example: range(5, 15) to generate [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
    - Must return list with custom start value
3. Stepped Sequence Generation:
    - Must use range() with start, stop, step parameters
    - Example: range(0, 20, 2) for [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
    - Must use step parameter to control increments
4. Reverse Sequence Generation:
    - Must use range() with negative step parameter
    - Example: range(10, 0, -1) for [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
    - Must handle decreasing sequences correctly

2.3 Output Constraints

1. Display Format:
    - Show "Sequence Generator System"
    - Show sequence type and parameters used
    - Show generated sequence and basic properties

1. Sequence Properties:
    - Show length, sum, and average of sequence
    - Show count of even and odd numbers
    - Simple formatted output for readability

4. TEMPLATE CODE STRUCTURE:

1. Generator Functions:

- generate_basic_sequence(count): Creates a sequence from 0 to count-1
- generate_custom_sequence(start, end): Creates a sequence from start to end-1
- generate_stepped_sequence(start, end, step): Creates a sequence with specified step
- generate_reverse_sequence(start, end, step): Creates a decreasing sequence
- analyze_sequence(sequence): Calculates properties like sum, even/odd counts
- display_sequence_report(sequence, properties, sequence_type): Formats and displays results

2. Main Section:

- Display menu with sequence generation options
- Get user input for sequence parameters
- Generate the requested sequence
- Analyze sequence properties
- Display formatted results
- Allow user to generate another sequence or exit

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Select sequence generation method from menu
3. Enter required parameters
4. View generated sequence and properties
5. Repeat or exit