Python Assignment

CS253 – 2025, IIT Kanpur

Deadline: April 18, 2025

Total Marks: 60

Important Note on Plagiarism Policy

Plagiarism is strictly prohibited. Students must ensure that all submitted work is their own. Any form of copying, unauthorized collaboration, or use of AI-generated solutions without proper understanding will be considered a violation of academic integrity.

- Direct copying from peers, online sources, or unauthorized material will result in **zero marks** for the assignment.
- Proper documentation and code comments are encouraged to demonstrate understanding.

Ensure that your submission reflects your own effort and comprehension.

Contact

Any queries regarding this assignment should be directed to **Moinuddin** (moinuddin23@iitk.ac.in) and Souvik Mukherjee (Email: souvik@cse.iitk.ac.in). Please mention CS253 in the subject line while sending any email.

Best of Luck!!

1. Introduction to Python Programming

1.1 Smart Text-Processing Tool [10 Marks]

In a small town in India, a young software engineer named **Amit** is developing a smart text-processing tool for his community. The tool is designed to assist local writers, poets, and journalists in formatting their content efficiently using a special numeric key.

The tool takes two inputs:

1. A **unique registration number** n (a positive integer) assigned to each writer.
2. A **content string** s (a non-empty string containing only letters) representing the writer's text.

To ensure dynamic content processing, the tool follows these rules:

1. **Even Registration Number:** If n is **even**, the tool **reverses the content** to reflect a "mirror perspective," inspired by ancient Sanskrit scripts.
2. **Odd Registration Number:** If n is **odd**, the tool converts all **vowels to uppercase** (to highlight their importance) and **consonants to lowercase** (to soften their impact), just like in poetic recitations.
3. **Word Pattern Extraction:** The number of **1s in the binary representation** of n determines a substring length k. The tool extracts all substrings of length k from s and stores them for analysis.
4. **Lexicographical Ordering for Clarity:** If the **bitwise AND** between n and the **length of** s results in zero, Amit's tool sorts these extracted substrings in **lexicographical order** (helpful for dictionary-based archiving). Otherwise, it lists them in **reverse order** (often used for artistic effect in poetry).

Amit ensures that his program can handle multiple requests from different writers by implementing **proper control flow**, **input validation**, and **exception handling**.

Function Definitions

The following functions are used to implement Amit's Smart Text-Processing Tool:

Function Name	Purpose
`validate_input(n, s)`	Ensures n is a positive integer and s is a non-empty string containing only letters.
`process_string(n, s)`	Applies transformations based on the parity of n. If n is even, it reverses the string. If n is odd, it converts vowels to uppercase and consonants to lowercase.
`count_set_bits(n)`	Counts the number of 1s in the binary representation of n, which determines the substring length k.
`extract_substrings(s, k)`	Extracts all substrings of length k from s.
`sort_or_reverse(n, s, substrings)`	Sorts substrings lexicographically if $n\&\text{len}(s) = 0$, otherwise reverses the order of substrings.
`main()`	Manages user input, calls necessary functions, and ensures multiple test cases can be handled.

Example Input and Output

Input 1:

```
n = 10
s = "bharat"
```

Processing Steps:

- n is even, so s is reversed: `"tarahb"`
- Number of 1s in binary 10 (`1010`) is 2. So, substrings of length 2 are extracted: `"ta", "ar", "ra", "ah", "hb"`
- Bitwise AND between n and len(s): $10\&6 = 2 \neq 0$
- Substrings are printed in reverse order: `"hb", "ah", "ra", "ar", "ta"`

Output 1:

```
hb ah ra ar ta
```

Input 2:

```
n = 7
s = "kavita"
```

Processing Steps:

- n is odd, so vowels become uppercase, consonants become lowercase: `"kAvItA"`
- Number of 1s in binary 7 (`111`) is 3. Substrings of length 3: `"kAv", "AvI", "vIt", "ItA"`
- Bitwise AND between n and len(s): $7\&6 = 6 \neq 0$
- Substrings are printed in reverse order: `"ItA", "vIt", "AvI", "kAv"`

Output 2:

```
ItA vIt AvI kAv
```

1.2 Rock-Paper-Scissors [8 Marks]

RPS is a popular hand game played between two players. The rules of the game are as follows:

- Rock beats Scissors.
- Scissors beats Paper.
- Paper beats Rock.

Imagine you and your friend want to decide who gets the last slice of pizza. Instead of arguing, you both agree to play a fair Rock-Paper-Scissors game. The winner gets the pizza, and if it's a tie, you play again.

Write a Python program that simulates a Rock-Paper-Scissors game between a human player and the computer. Your program should:

1. Allow the user to input their choice (Rock, Paper, or Scissors).
2. Generate a random choice for the computer.
3. Determine the winner using the rules of the game.
4. Display the choices of both the user and the computer.
5. Allow the user to play multiple rounds until they decide to quit. [By inserting the **'Quit'** command (case independent)].

Constraints

- The user's input should be case-insensitive (e.g., "rock", "Rock", or "ROCK" should all be valid).
- The program should handle invalid inputs and prompt the user to enter a correct choice.
- The program should display the total score (wins, losses, and ties) after each round.
- Use the `random` module to generate the computer's choice.

2. Tony's Research and Computational Challenges

In a research institute in Bengaluru, Tony, a data scientist, is working on various numerical computations to assist researchers and analysts. His work involves optimizing algorithms for efficiency and accuracy in real-world applications. One day, Tony encounters the following computational challenges:

2.1 Summing Prime Numbers for a Cryptographic Algorithm [7 Marks]

A cybersecurity expert approaches Tony with a problem. In order to develop a new cryptographic system, they need to compute the sum of the first n prime numbers efficiently. Tony must devise a fast algorithm to generate prime numbers and compute their sum. [Study and use **"Sieve of Eratosthenes"** method]

Function to implement:

- `sum_of_primes(n)` - Computes the sum of the first n prime numbers.

2.2 Analyzing Temperature Fluctuations [10 Marks]

A meteorologist studying climate change provides Tony with temperature readings collected over m days from a remote weather station in the Himalayas. To analyze the temperature trends, Tony is asked to compute:

- The mean temperature
- The median temperature
- The standard deviation
- The variance (calculated as **sample variance**, not population variance, using **Bessel's correction** for an unbiased estimate).

These calculations will help determine the fluctuations and patterns in the temperature data.

Functions to implement:

- `analyze_temperatures(temperatures)` - Computes the mean, median, standard deviation, and variance of a given list of temperatures.

Important Considerations:

- Use the appropriate mathematical/statistical library (e.g., `math` or `statistics`).
- Handle edge cases such as an empty dataset or a single temperature reading ($m = 0$ or $m = 1$), ensuring division by zero errors are avoided.

2.3 Solving a System of Linear Equations in Physics [10 Marks]

In a physics laboratory, researchers are working on a set of equations that model the movement of particles in a magnetic field. These equations form a system of linear equations of the form:

$$AX = B$$

where A is an $N \times N$ matrix and B is an $N \times 1$ vector. Tony is asked to efficiently compute the solution X.

Function to implement:

- `solve_linear_system(A, B)` - Solves the system of linear equations $AX = B$.

Important Considerations:

- Use of NumPy is allowed, and prompt an error in case of singular matrices.

Tony successfully implements the program and helps the cybersecurity expert, the meteorologist, and the physicists with their respective computational challenges.

3. Shivam's Data Science Experiment [15 Marks]

Shivam, a data scientist at a weather research institute in Pune, is working on a project to analyze climate patterns using synthetic data. His goal is to generate and visualize random datasets to study trends in environmental data.

To achieve this, Shivam needs to:

1. **Generate a synthetic dataset** representing environmental observations:
   - A dataset with N random points for two numerical variables X and Y.
   - X values should be uniformly distributed within a user-defined range.
   - Y values should be computed using a randomly generated mathematical function of X, following the form:

     $$Y = Af_1(BX) + Cf_2(DX) + Ef_3(FX)$$

     where:
     - A, B, C, D, E, F are randomly generated constants.
     - f_1, f_2, f_3 are randomly chosen from {sin, cos, tan, log (valid for positive X), square, cube}.
2. **Generate visualizations** to analyze patterns:
   - A **scatter plot** of X vs Y.
   - A **histogram** of X with an appropriate number of bins.
   - A **box plot** of Y to detect outliers.
   - A **line plot** of sorted X values against their corresponding Y values.
3. **Enhance clarity** by adding:
   - Proper plot titles.
   - Axis labels for better understanding.
   - Legends (if applicable).
4. **Allow user customization**:
   - The number of data points (N).
   - The range of values for X.
5. **Ensure reproducibility** by setting a random seed.

Functions to Implement

To accomplish the tasks efficiently, Shivam decides to implement the following functions:

- `generate_dataset(N, x_min, x_max)`: Generates N random points for X and computes Y using a randomly generated function.
- `plot_scatter(X, Y)`: Creates a scatter plot of X vs Y.
- `plot_histogram(X)`: Plots a histogram of X.
- `plot_box(Y)`: Generates a box plot for Y.
- `plot_line(X, Y)`: Creates a line plot of sorted X against Y.
- `set_random_seed(seed)`: Ensures reproducibility by setting a fixed seed.

By implementing these functions, Shivam ensures that his institute can generate meaningful insights from climate data, improving their predictive models.