Department of Computer Science and Engineering

PES University, Bangalore, India

Python for Computational Problem Solving (UE24CS151A)

Level-2(Orange) Problem Statements

Addressed to: All faculty members

General Guidelines to form Level-2 Problems

- **Define the Real-World Context**
    - Choose a relatable real-world scenario, such as budgeting, basic data analysis, gaming mechanics, Budget tracker or expense manager, Simple quiz game, Inventory or stock manager for small businesses.
    - Explain the context briefly and outline any assumptions or simplifications made for the problem.

- **Focus on Practical Applications of Python Skills**
    - Reading from CSV Files (File handling)
    - Filtering the data after reading from CSV files using looping constructs and conditional statements.
    - Implementation of Callbacks using built-in functions.
    - List and String manipulation and Concatenation.
    - Creating Functions and calling Functions and to modularizing the code.
    - Importing and Running Tkinter module.
    - Creating Tkinter windows and basic window styling (Resize, Color and Title)
    - Creating Labels and destroying labels.
    - Global Variables
    - Creating Checkboxes and deselecting them through code.
    - Creating Buttons and giving functions in the command attribute of the button

- **Encourage Problem Decomposition**
    - Break down the problem into smaller tasks or functions to foster a structured approach to coding.
    - Provide examples or hints on identifying tasks within the larger problem (e.g., "Define a function to calculate the total cost given a list of expenses").

- **Incorporate Basic Data Analysis**
  - Use datasets relevant to the context, encouraging students to analyze or summarize data.
  - Simple tasks could include calculating averages, finding maximum/minimum values, or summarizing categories.

- **Test for Edge Cases**
  - Ensure students consider edge cases (e.g., an empty list, extremely high or low values).
  - Provide examples of inputs that might lead to unexpected outputs or errors, guiding students to anticipate these cases.

- **Introduce Tkinter Layout Management**
  - Familiarize students with Tkinter layout methods (pack, grid, place) for organizing the interface effectively.
  - Provide examples of simple layouts to help students manage the positioning and alignment of widgets
  - Suggest using labels, entry fields, and buttons in a clear, easy-to-read layout to build simple User-Friendly Interfaces.
  - Demo the expected output.

-END-