**Python Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Segun Akinshipe**

**Date: 03 – 12 - 2024**

**Final IDE Program Share Link:**

Complete the following template. Fill out all entries using complete sentences.

## PART 1: Defining Your Problem

|  |
| --- |
| **Task**  State the problem you are planning to solve.  **Requirements**   * Describe the problem you are trying to solve for. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| The problem we are solving is the requirement for a productive method of handling everyday duties. Many people find it difficult to remember their to-do lists, which causes them to forget obligations and miss deadlines.    This program will allow users to:   * **Add tasks**: Users can input new tasks they need to complete. * **View tasks**: Users can see a list of all current tasks. * **Delete tasks**: Users can remove tasks that have been completed or are no longer necessary.   **Inputs**:   * Task descriptions (strings) * User commands (add, view, delete)   **Outputs**:   * A list of tasks displayed to the user * Confirmation messages for adding or deleting tasks |

## PART 2: Working Through Specific Examples

|  |
| --- |
| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.  **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Python) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| 1. **Adding a Task**:    * User inputs the command add followed by the task description.    * The program stores this task in a list. 2. **Viewing Tasks**:    * User inputs the command **view**.    * The program displays all tasks currently in the list. 3. **Deleting a Task**:    * User inputs the command delete followed by the task number.    * The program removes the specified task from the list and confirms deletion. |

## PART 3: Generalizing Into Pseudocode

|  |
| --- |
| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.  **Requirements**   * Write pseudocode for the program in English but refer to Python program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.   **Help with writing pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert it to code later.   + <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>   + <https://www.wikihow.com/Write-Pseudocode>   **Inspiration**  When writing your entry below ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Python? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| START PROGRAM  INITIALIZE empty task list  WHILE true  DISPLAY menu options (add, view, delete, exit)  GET user input  IF user input is "add"  GET task description from user  ADD task description to task list  DISPLAY confirmation message  ELSE IF user input is "view"  IF task list is empty  DISPLAY "No tasks available"  ELSE  FOR each task in task list  DISPLAY task with its index  ELSE IF user input is "delete"  GET task index from user  IF index is valid  REMOVE task from task list  DISPLAY confirmation message  ELSE  DISPLAY "Invalid index"  ELSE IF user input is "exit"  BREAK loop  END PROGRAM |

## PART 4: Testing Your Program

|  |
| --- |
| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.  **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from your IDE. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify what approach was the one that worked.   **Inspiration**  When writing your entry below ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| 1. **Adding Tasks**:    * Test by adding multiple tasks and verify they appear in the list.    * Check for edge cases like adding an empty task. 2. **Viewing Tasks**:    * Test viewing when there are no tasks and ensure it displays an appropriate message.    * Verify that all added tasks are displayed correctly. 3. **Deleting Tasks**:    * Test deleting a valid task and ensure it is removed from the list.    * Attempt to delete an invalid index and check for error handling.   Errors Found:   * If trying to delete a non-existent index, the program should handle this gracefully without crashing.   Fixing Errors:   * Implement checks for valid indices before attempting deletion. |

## PART 5: Commenting Your Program

|  |
| --- |
| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.  **Requirements**   * The purpose of the program and each of its parts should be clear to a reader that does not know the Python programming language.   **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer. |
| **# Task Manager Program**  **# Initialize an empty list to store tasks**  tasks = []  **# Start an infinite loop for user interaction**  while True:  # **Display menu options for user commands**  print("Options: add, view, delete, exit")  command = input("Enter command: ")  **# Adding a new task**  if command == "add":  task\_description = input("Enter task description: ")  if task\_description: # Check if not empty  tasks.append(task\_description) # Add to tasks list  print("Task added successfully.")  else:  print("Task cannot be empty.")  **# Viewing all current tasks**  elif command == "view":  if not tasks: # Check if the list is empty  print("No tasks available.")  else:  for index, task in enumerate(tasks):  print(f"{index + 1}: {task}") # Display each task with its index  **# Deleting a specific task by index**  elif command == "delete":  try:  index\_to\_delete = int(input("Enter task number to delete: ")) - 1 **# Convert to zero-based index**  if 0 <= index\_to\_delete < len(tasks):  removed\_task = tasks.pop(index\_to\_delete) **# Remove from list**  print(f"Task '{removed\_task}' deleted successfully.")  else:  print("Invalid index.")  except ValueError:  print("Please enter a valid number.")  **# Exiting the program**  elif command == "exit":  break  **# End of program** |

## PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the IDE share link to your full program code.  **Requirements**   * The program must work correctly with all the comments included in the program.   **Inspiration**   * Check before submitting your touchstone that your final version of the program is running successfully. |
| < |