**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: Andre Echevarria**

**Date: 10/9/2024**

**Final Replit 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? |
| I am going to solve the problem for users who want to manage daily chores efficiently. Most of them can't remember their to-do list, which further leads to missed deadlines and stress. This program lets a user create, view, modify, and delete activities or tasks from their to-do list.  Data for the description involves describing a task with due date and priority that the user will base their inputs on. The program will prompt the users to add tasks and control them through a simple command-line interface.  Outputs shall be listing current tasks, confirmation messages any time a task gets added or deleted, and also error messages in case the user tries to perform invalid operations. |

## 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? |
| Example 1: A task being added to the list Input Data: Person inputs a task and a due date  Steps:  1. Prompt user to enter description of task.  2. Capture description as entered by the user  3. Ask user to provide due date.  4.Capture due date as entered by user  5. Add this task description and due date to the list of tasks  6. Acknowledge back to user that this is added in the list.  Result: Saves the task in the to-do list and notifies the user that an operation has been successfully executed.  Example 2: Removing a task from the list  Input Data: User enters the number of task to remove  Steps:  1. Display current to-do list with numbers of tasks.  2. Ask user to input number of the task he would like to remove.  3. Receive a number entered by the user  4. Check if that number is present in a list.  5. In case of its presence remove the task from a list.  6. Let the user know that the task has been deleted. If it's not in the list, let the user know that.  Result: The list displays the removed task specified, or an error message is shown in case the task number doesn't exist. |

## 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. |
| <Write your pseudocode here>  BEGIN Program  INITIALIZE an empty task list  WHILE user wants to continue  DISPLAY menu options (Add task, View tasks, Delete task, Exit)  PROMPT user for a choice  IF choice is "Add task" THEN  PROMPT user for task description  STORE description  PROMPT user for due date  STORE due date  ADD task (description, due date) to task list  DISPLAY confirmation message  ELSE IF choice is "View tasks" THEN  IF task list is empty THEN  DISPLAY "No tasks available."  ELSE  DISPLAY the task list with task numbers  ELSE IF choice is "Delete task" THEN  DISPLAY the task list  PROMPT user for task number to delete  IF task number exists THEN  REMOVE task from task list  DISPLAY confirmation message  ELSE  DISPLAY "Task number not found."  ELSE IF choice is "Exit" THEN  BREAK  ELSE  DISPLAY "Invalid option, please try again."  END WHILE  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 Replit. 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? |
| I knew if it the program was running based upon the error codes or if after the decisions were made it would exit when asked.  The only error I encountered was the one below and it was solved when I added a “:” to line 19. |

## 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. |
| # To do list Program  # Set an empty to store tasks task\_list = []  # Main loop about whether the program will keep working, depends on user choice while True:  # Showing user menu  print("Menu Options:")  print("1. add a task")  print("2. view tasks")  print("3. delete task")  print("4. exit")   #Getting user's choice  choice = input("enter your choice (choose a number 1-4):")   # Option adding task  if choice == "1":  description = input("enter task description:")  due\_date = input("enter due date: ")   # Add the task to the list and confirm  task\_list.append((description, due\_date))  print("task added!")   # Option viewing tasks  elif choice == "2":  if not task\_list: #check if the list is empty  print("No tasks available.")  else:  print("Tasks:")  for idx, task in enumerate(task\_list):  print(f"{idx + 1}. {task[0]} (Due: {task[1]})")   # Option deleting task  elif choice == "3":  if not task\_list:  print("No tasks available.")  else:  print("Tasks:")  for idx, task in enumerate(task\_list):  print(f"{idx + 1}. {task[0]} (Due: {task[1]})")  task\_index = int(input("Enter the task number to delete: ")) - 1  if 0 <= task\_index < len(task\_list):  deleted\_task = task\_list.pop(task\_index)  print(f"Deleted task: {deleted\_task[0]} (Due: {deleted\_task[1]})")  else:  print("Invalid task number.")   # Option exiting the program  elif choice == "4":  print("Exiting the program.")  break   else:  print("Invalid choice. Please try again.") #invalid inputs |

## PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the Replit 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. |
| https://replit.com/join/aebghxgyea-andreechevarria |