**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: Whitney Perry**

**Date: 7/18/2024**

**Final Replit Program Share Link:** <https://replit.com/join/zlmfkzljwd-whitneyperrypro>

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 will be building a computer program game for Mad Libs. This game will be solving boredom with silliness by asking for a series of random words to the user and inputting those words into a story the user will not see until the end. The user will be asked to supply these random words via parts of speech prompt (noun, verb, adjective, etc.). The user will continue to enter random words until all blank words in the story and filled in. At this point, a stored story will print with the user inputted words filled in the various blank areas of the story. Once the story has been printed, the program will ask the user if they would like to play again or quit. |

## 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. Scenario 1: Player follows the game instructions.    1. Request for user to enter a noun.    2. User enters the noun “cat”.    3. Request for user to input a verb.    4. User enters the verb “dancing”.    5. The game outputs a story: “The cat is dancing at my school!” 2. Scenario 2: Player doesn’t follow the game instruction.    1. Request for user to enter a noun.    2. User enters the non-noun “bouncing”.    3. Request for user to input a verb.    4. User enters the non-verb “yellow”.    5. The game outputs a story: “The bouncing is yellow at my school!” 3. ~~Scenario 3: Player wants the game to generate words for the story.~~    1. ~~Request for user to enter a noun.~~    2. ~~User enters nothing.~~    3. ~~Program generates noun “sun”~~    4. ~~Request for user to input a verb.~~    5. ~~User enters nothing.~~    6. ~~Program generates verb “skydiving”~~    7. ~~The game outputs a story: “The sun is skydiving at my school!”~~ |

## 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. |
| Function play()  Print (“Mad Libs is a hilarious game of storytelling for the whole family. Players will fill in the blanks with words without actually being able to read the sentence that they are given. The players will be asked for certain types of words, such as nouns, verbs, or adjectives to fill in the blank. The players will write their words. Once finished, the story is read, using their words, leading to loads of fun!”)  Print (“Definitions:   * Noun – is a word that names a person, place, thing, or event. Example: man, child, city, Paris, dog, bee, book, flower, hat. * Verb – is a word that denotes an action or state of being. Example: walk, run, remember, listen, play, read, yell, relax. * Adjective – is a word that describe or modifies a noun. Example: careful, charming, exciting, fearless, dry, dirty, delicious. * Adverb – is a word that modifies a verb, an adjective, or another adverb. Example: quickly, sadly, gleefully, very, sometimes, boldly, soon, repeatedly. * Preposition – is a word that shows the relationship between a noun or a pronoun in a sentence. Example: above, below, between, with, during, except, upon, against, across. * Interjection – is a word that shows an intense feeling or emotion. Example: wow, tikes, uh oh, ouch, shh, whew, help, aha, tsk-tsk, drat, um, oops.   ”)  Print(“Type play to play the game or quit to stop the program.”)  Generate random number to ID the pick a random story.  Load variables:  Word 1 … Word 20  Load story variable.  Load variable type of speech list  Load variable word list  #All counters will be greater than 0 predefined for each story  Load PartsOfSpeech word counter.  For PartsOfSpeach Word counter = 0 to PartsOfSpeech Word  While PartsOfSpeech word counter != 0  Print(“Give me a PartsOfSpeach Type.”)  Store user input  PartsOfSpeech Type counter minus 1  End While  End For  Print(Story)  Print(“Type play to play again or quit to stop the game.”)  Story 1:  Dear Santa. I have been a very **Adjective1** **Noun1** this year. I always help **Person1** with chores around the **Noun2**. It’s my job to **Verb1** the **Noun3** and take out the **Noun4** everyday. I really hope that I am on the **Adjective2** list this year. I have done a lot of **Adjective3** things, so I think that I deserve it. I even helped **Person2** feed the **Animal1** while they were on vacation. I have a few wishes this year. I would love to see a **Adjective4** new **Noun5** underneath the tree with my name on it. It would make me the happiest **Noun6** on the **Noun7**! Oh, and if you could put a **Adjective5** **Noun8** inside of my stocking, that would be **Adjective6** too! **Interjection1**, I love the holidays!  Story 2:  There are many **adjective** ways to choose a/an **noun** to read. First, you could ask for recommendations from your friends and **pluralnoun**. Just don’t ask Aunt/Uncle **Person1** he/she only reads **adjective** books with **articleofClothing**-ripping goddesses on the cover. If your friends and family are no help, try check out the **noun** Review in The **City** Times. If the **PluralNoun** featured there are too **adjective** for your taste, try something a little more low-brow, like The **PluralNoun** Magazine or The **PluralNoun** Digest. You could also choose a book the **Adjective**-fashioned way. Head to your local library or **A Place** and broswe the shleves until something catches your **Part Of Body**. Or, you could save yourself a whole lot of **Adjective** trouble and log on to www.book**Place**.org, the **adjective** new website to **verb** for books. With all the time you’ll save not having to search for **plural noun**, you can read at least **number** more books! |

## 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 had difficulty getting the counter to work. I looked online for some good examples and found that enumerate simplifies counters for lists. * I realized that I wanted a menu so I took inspiration from the salary program from the coursework and modified it to fit my program * I had trouble understanding how an f string worked to get the word index I was looking for. * I found that you cannot format a string within an f string so I added the title format before the story. * Replit seems to think that my f strings are too long but when I used the AI code fix suggestion, it just broke my code. It seems that despite the error, it works just fine * My lists were showing an error of being too long. I originally fixed it by using .append to the list but the AI code fix suggested just doing a return carriage instead every third word. That made the code look much cleaner. * I didn’t like how the console was cutting off my words so I found that I could import textwrap to improve readability. * I had to do many tests to make sure that the right word type aligned with the story properly. I made some minor modifications to add more user input. * To improve this in the future, I might add the story and the types of words to a file to make the code even cleaner. I would be able to turn reading the story into a function instead. |

## 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. |
| import csv  import sys  import textwrap  FILENAME = "rules.csv"  #exiting the program  def exit\_program():  print("Terminating program.")  sys.exit()  #Read the rules from the file  def read\_rules():  try:  rules = []  with open(FILENAME, newline="") as file:  reader = csv.reader(file)  for row in reader:  rules.append(row)  return rules #this fills out the rules in a list from the rules.csv file  except FileNotFoundError:  print(f"Could not find {FILENAME} file.")  exit\_program()  except Exception as e:  print(type(e), e)  exit\_program()  #Display the rules  def list\_rules(rules):  for i, rules in enumerate(rules, start=1):  print()  print(textwrap.fill(f"{i}.{rules[0]} - {rules[1]}", 70)) #this prints the rules  print()  print("-----------------------------------------")  print()  display\_menu() #this brings us back to the menu  #Play the Christmas Story  def play\_christmas():  word = []  items = ['Adjective', 'Noun', 'Person'  ,'Noun', 'Verb' ,'Noun'  ,'Noun', 'Adjective','Adjective'  ,'Person', 'Animal', 'Adjective'  ,'Noun', 'Noun', 'Noun'  ,'Adjective', 'Noun', 'Adjective',  'Interjection'] #this instructs the user what type of word to enter    for index, item in enumerate(items):  word.append(input(item + ": "))  # this requests the user to input a word for each item in the list  # and stores it in the word list.    ChristmasStory = f"Dear Santa, I have been a very {word[0]} {word[1]} this year. I always help {word[2]} with chores around the {word[3]}. It’s my job to {word[4]} the {word[5]} and take out the {word[6]} everyday. I really hope that I am on the {word[7]} list this year. I have done a lot of {word[8]} things, so I think that I deserve it. I even helped {word[9]} feed the {word[10]} while they were on vacation. I have a few wishes this year. I would love to see a {word[11]} new {word[12]} underneath the tree with my name on it. It would make me the happiest {word[13]} on the {word[14]}! Oh, and if you could put a {word[15]} {word[16]} inside of my stocking, that would be {word[17]} too! {word[18]}, I love the holidays!"  #this stores the story in a variable called ChristmasStory    print()  print("Christmas Story")  print(textwrap.fill(ChristmasStory, width=50)) #this prints the story  print()  print("-----------------------------------------")  print()  display\_menu() #this brings us back to the menu  #Play the Book Recommendation Story  def play\_book():  word = [  items = ['Adjective', 'Adjective', 'Plural Noun'  , 'Person', 'Genre' ,'Article of Clothing'  , 'Noun', 'Plural Noun','Adjective'  , 'Plural Noun', 'Plural Noun', 'Adjective'  , 'Place', 'Part of Body', 'Adjective'  , 'Noun', 'Adjective', 'Verb'  , 'Plural Noun', 'Number'] #this instructs the user what type of word to enter  for index, item in enumerate(items):  word.append(input(item + ": "))  # this requests the user to input a word for each item in the list  # and stores it in the word list.  word[6] = word[6].title()  word[9] = word[9].title()  word[10] = word[10].title()  BookStory = f"There are many {word[0]} ways to choose a/an {word[1]} book to read. First, you could ask for recommendations from your friends and {word[2]}. Just don’t ask Aunt/Uncle {word[3]} he/she only reads {word[4]} books with {word[5]}-ripping goddesses on the cover. If your friends and family are no help, try check out the '{word[6]} Review' in 'The City Times'. If the {word[7]} featured there are too {word[8]} for your taste, try something a little more low-brow, like 'The {word[9]} Magazine' or 'The {word[10]} Digest'. You could also choose a book the {word[11]}-fashioned way. Head to your local library or the {word[12]} library and browse the shelves until something catches your {word[13]}. Or, you could save yourself a whole lot of {word[14]} trouble and log on to www.book{word[15]}.org, the {word[16]} new website to {word[17]} for books. With all the time you’ll save not having to search for {word[18]}, you can read at least {word[19]} more books!"  print()  print("Book Recommendation Story")  print(textwrap.fill(BookStory, width=50)) #this prints the story  print()  print("-----------------------------------------")  print()  display\_menu() #this brings us back to the menu  #Display Menu  def display\_menu():  print("MAD LIBS")  print()  print("MENU - TYPE A COMMAND TO PLAY")  print("rules - Learn the rules of the game")  print("christmas - Play Christmas game")  print("book - Play the Book Recommendation game")  print("exit - Exit program")  print()  display\_menu()  rules = read\_rules()  while True:  command = input("Command: ")  if command.lower() == "rules": #User types in rules to show rules  list\_rules(rules)  elif command.lower() == "christmas": #User types in christmas to play the christmas game  play\_christmas()  elif command.lower() == "book": #User types in book to play the book game  play\_book()  elif command.lower() == "exit": #This exists the program  break  else:  print("Not a valid command. Please try again.\n")  print("Ending Mad Libs Program") |

## 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/zlmfkzljwd-whitneyperrypro> |