1. Problem Statement

The problem being presented is to take a traditional Mad Lib and translate it from paper to code. The program needs to accept user input based on prompts for certain parts of speech, then present the output as a story with the blanks filled in by the user input. The program must also be pep8 compliant.

* Requirements for this program are that it must prompt for user input with a message like, “Please insert a noun.” The program must then substitute that value into a prewritten story with blanks that are assigned to each piece of user input. Finally, the program then displays the story with the user input plugged in, which will create a complete Mad Lib story.
* Variables will be assigned value by the user input. For example, to fill in the first verb blank of the Mad Lib, first\_verb = input(“Type a verb: “) will allow the user to type a verb.
* The print() command will display the story with the variables presented as strings concatenated with the prewritten Mad Lib, and the result should be a complete story.

2. Planning

The planning process included writing the Mad Lib story prior to coming to lab and deciding on the techniques to use to implement the story as Python code. We went through the story and counted out which of each type of speech was presented, such as the first verb, second verb, etc. We then went into the .py file and began assigning values to the variables. We continued down the story filling in these blanks as the value of the user input based on prompts. As an example:

first\_verb = input(“Type a verb: “)

first\_noun = input(“Type a plural noun: “)

first\_pronoun = input(“Type a pronoun: “)

We continued down the list until all blanks had a prompt for user input. We then used the print() function to display the completed story with the filled in blanks:

print(“There once was a boy who liked to “ + first\_verb + “with “ + first\_noun)

If the program functioned properly, the user would insert a verb, let us use “run” as an example, and a noun, let us use “dogs”. The program should output the strings from the prewritten code concatenated with the user input to look like this: “There once was a boy who liked to run with dogs.”

3. Implementation and testing

When we first tested the code, it ran as it should asking the user for different grammar types and ending with displaying the final poem as a result. There were no spaces in-between the variables and strings though which happened because we didn't add spaces inside of our strings to make up for the variable inputs.

4. Reflection

As far as the basic function of the program goes, everything went smoothly. It did what it needed to do the first time we implemented and tested the solution. Unfortunately though, apon testing the code we found it to have many errors that were not aligned with pep8 compliance. The code had many errors including lines that were over 80 characters long and white-space in places there didn't need to be any. To fix this, we used carriage returns in each line that needed to be shortened which allowed us to meet the requirements for pep8 compliance. Once each long line was indented we added print functions to that line to keep the program consistent and working properly.

Unfortunately though we did not realize as beginning programmers that you do not add a plus sign at the end of a line (which there was a lot of once we shortened the long lines of code) because each new print function is basically added to the previous line of code. When there were plus signs at the end of each line and space in between that and the closing parenthesis + )

the pep8 checker would read “invalid syntax” and “whitespace”. It took a while to figure out but once the plus signs and plus signs were deleted along with the long lines of code being shortened, our code became pep8 compliant.

Another error we encountered was missing space between the variables and the strings. We should have accounted for the lack of space between variables and strings to make the program more user-friendly. Our variables were concise and let the code editor know exactly what they needed to edit, so the programming side of things went according to plan.