COM S 127 - Assignment #1

Please see the 'Course Schedule' on the Syllabus for this assignment's due date.

Assignment Objective

The objective of the assignment is for students to practice printing text strings, gathering input from the user with the `input()` function, and concatenating strings and input together.

The purpose of this assignment is for students to practice putting code into a file called `zanyTexts.py`, saving that file, and being able to run that file at will from the command prompt.

Instructions

A "Zany Text" is a humorous short 'story' created by inserting words, entered into the program via a text prompt, into a sentence that is missing one or more of its original words. The 'catch,' is that you do not know what the sentence is or which words are missing - just the parts of speech for the words (e.g. noun, verb, color, etc.).

You will need to make four (4) "Zany Texts" in total in your file. They will be executed one after the other such that you will gather input for the first one, and then the first one will print out. You will then gather input for the second one, and then the second one will print out, etc. There should be exactly four (4) "Zany Texts" in the completed assignment and they should all be original. Do not copy the example below.

Additionally, the `zanyTexts.py` script should output a 'header,' including the name of the program ("Zany Texts!"), the student's name, and the class/ section they are in as per the `Example Output` below.

Example Output

```
Zany Texts!

By: Matthew Holman
[COM S 1270 1]

Zany Text #1

noun: cat
adjective: silly
adjective: happy
adjective: blue
past tense verb: ran
noun: house

Once upon a time, there was a cat. It was a silly, happy, blue cat. And, one day it ran all over the house!
```

etc.

You will output your four original "Zany Texts" in the same 'run' of the program as the provided example. Do not make multiple scripts - they all 'live' in the same script together.

Additional Instructions

Students' work should be their own completely original effort, and should reflect a degree of industriousness that goes beyond 'copying and pasting' the instructor's example output (seen above) four times. While the student's "Zany Text" stories can be anything they like, they should be different from the example the instructor provides. To that end, it should be noted that simply changing the instructor's example ever so slightly is insufficient. This is true in either the 'input gathering' phase, or the final print-out phase of each "Zany Text." While this is a subjective decision to a certain degree, the best way to avoid any problems is to make four completely original "Zany Texts." **As such, this assignment, as simple as it is, may, in fact, take more than 'three seconds' to complete.**

IMPORTANT NOTE: The user will enter their own input/ words in the terminal. These words are their choice. The student's code will only ask for parts of speech such as nouns, verbs, etc. Do *not* try to enter your own words into the `input()` function prompt strings. That will not work. For example: `noun1 = input("shark")` will not actually enter the word 'shark.' *You* may want to enter the word 'shark,' but that is actually the user's choice - they can actually enter anything they like by way of the terminal.

If you need a 'refresher' on the different 'parts of speech,' please review the material at the following link:

https://help.butte.edu/departments/cas/tipsheets/grammar/parts_of_speech.html

Additionally, students do *not* need to compensate for the crashes that occur when the user enters a value of the incorrect type. (E.g., The program asks for an integer, but the user enters a letter instead.)

It is required for the student to write their name, the date they started working on their script, and a short explanation of what their code does at the top of their file. For example:

```
# Matthew Holman              6-6-2024
# Assignment #1
#
# This is a silly game called 'Zany Texts!' wherein the user
# enters words, and a fun story results!
```

Students' work should be their own, completely original effort. Meaning - they should not just copy my code explanation above, but should try to come up with one of their own. Further, the actual Python script itself can be programmed in any way (excluding cheating ala ChatGPT) so long as the output resembles that seen above.

NOTE: If you use commas inside your print statements, (ex: `print("Hello", "there!")`) you may get undesired spaces in your final printouts. Do not worry about these for now - they will not affect your grade. However, if you like, you can try experimenting with the `+` operator for string concatenation (ex: `print("Hello" + " " + "there!")`). Try to see how this works with the variables you assign the text to from the `input()` function.

Frequently Asked Questions (F.A.Q.)

- **Q1**: Hey there - I'm a total 'lazy-bones' who hates fun. What is the minimum level of effort I can put into this and 'call it a day?' I want to be done with this in three seconds! (Why, no, now that you mention it, I have *no idea* why academic, personal, and professional success just seem to elude me. It's clearly all a grand conspiracy arrayed against me.)

- **A1**: You need to make four original 'Zany Texts' - none of which should match what the instructor has provided in the example output. Try to use your imagination.

- **Q2**: Ok - I made *one* new one. Can I just copy/ paste that a few more times and make a few little changes? It's been nearly two whole entire seconds since I started and I want to be done!

- **A2**: No - none of the 'Zany Texts' you make should bear any resemblance to one another or the example.

- **Q3**: So, when you say that `"none of the 'Zany Texts' you make should bear any resemblance to one another or the example"`, what the *really* means is that *something* in my "Zany Texts" should always `"____ all over the ____!"`, right? I mean, you definitely can't tell that I just lazily copy/ pasted the example and didn't bother to change it up very well, right?

- **A3**: No - we can tell. They should all be different.

- **Q4**: B... b... but... I *really* want something in my 'Zany Texts' to `"____ all over the ____!"` every single time! This is the artistic expression in my poor tortured soul!

- **A4**: As this is the 'instruction following class,' please follow the instructions and make them all different. You are free to follow the dictates of your artistic aspirations on your own.

- **Q5**: When the instructions at the top of this document say `"you will need to make four (4) 'Zany Texts' in total in your file"`, what that *really* means is to only do one (1) and then 'call it a day,' right? Although it *seems* like the the instructions say `"there should be exactly four (4) 'Zany Texts' in the completed assignment"`, that really means that only having one (1) is sufficient - right? Doesn't one (1) count as four (4)?

- **A5**: No - you must do four (4) brand new ones.

- **Q6**: All of this seems to indicate that I might have to do a minimal amount of typing. That's not fair! I *hate* typing! There shouldn't be any typing involved in computer class!

- **A6**: If you hate typing, then computer class is definitely the right place for you - in the same way that if you hate painting, a painting class is definitely what you want to take. Alternatively, if you hate creative writing, then taking a creative writing class is absolutely where you should be! (In seriousness, you will need to get used to the idea of typing a **lot** in computer class.)

- **Q7**: Do I *really* have to say the words "Zany Text?" That's so stupid and embarrassing! I'm so cool that I shouldn't have to do that!

- **A7**: Obviously, you can do whatever you want. However, the use of "Zany Text" as a title, filename, description of what we're doing, etc., will not be changing - either now or in the future.

Programming Advice

Notice how you can take in input from the user with the `input()` function. For example:

```
word1 = input("Please Enter a Word: ")
```

This code takes in input from the user through the terminal and places that input 'inside' the `word1` variable. You can then use this variable to print output to the screen. For example:

```
print("This is the word you entered: " + word1)
```

or

```
print("This is the word you entered:", word1)
```

Notice, both of the above examples work. However, in the example with the 'plus sign' operator, the space between the text and the contents of the `word1` variable had to be entered manually. In the version with the 'comma,' the space was added automatically.

You can run your code in the terminal (use **CTRL+`** (PC) or **Control+`** (Mac) to make it visible at the bottom of VS Code) with the following command:

(PC): `python zanyTexts.py`

(Mac): `python3 zanyTexts.py`

If it is not working, you are likely in the wrong folder. In VS Code, go to `'File' -> 'Open Folder'` and navigate to the exact folder where you put the `zanyTexts.py` file.

Special Notes

NOTE: This assignment should not be terribly difficult. However:

- Completing this assignment may require the student to start their work 'before the last minute.' Please plan accordingly.
- The student's script **CANNOT** crash under any circumstances except for those circumstances noted below. Any portions of the student's code where the script crashes will receive a zero (0) for that aspect of the script.
  - Understand, when the word 'crash' is used in the instruction above, what this means is 'crashing under expected use cases given the level of knowledge attained in the class.'
    - We do not currently have the ability to enforce user input types (although we will later in the semester). As such, if you ask for an integer as input (e.g., 1), and the user provides a letter (e.g., 'a'), and it crashes, then this is *not* a problem *yet*.

NOTE: The student is turning in their CODE - NOT the OUTPUT of their code.

- The student's code will be run by the TAs, and the output of those runs, along with the code itself, is what will be evaluated.

NOTE: Assignments turned in in any other format other the specified file types will not be accepted.

- Screenshots of code **will not** be accepted.
- `.sln` files are **not** code files - they contain **no** Python code and **will not** be accepted.
- `.zip`, `.rar`, `.tar.gz`, and other compressed files **will not** be accepted unless otherwise specified.
- If a student's submission is not in a `.py` file, when so specified, and if the submission is not accompanied by **all** the files needed to run the submission, the submission will not be graded.
  - **THIS WILL LEAD TO THE STUDENT RECEIVING A ZERO (0) ON THE ASSIGNMENT.**
  - **Students will *NOT* be allowed to re-submit their work after the final deadline in this case.**