SSEP 2024: Intro to Programming with Python

Summer 2024

Assignment 02: Mad Libs

*This is a **pair programming assignment find another person to work with on the assignment.***

Assignment Goals:

- Gather and store user input
- Manipulate input with mathematical and string operators
- Print information to the user

Notes

It is your responsibility to write a program that is well structured (i.e. modular), and easy to read. Your file should start with a header that has your name(s), the date, and a brief description of what your program does. Variable names should be descriptive. Comments should be used appropriately to document your code.

Quick Programming Tips:

- ***Not sure what to do?*** Talk through the assignment instructions with your partner and write out in English what specific tasks your program needs to do. Then, pseudocode.
- ***Stuck on how to program your solution?*** Try out 2 different ideas then ask for help if you are still stuck!
- ***Have a seemingly invisible bug?*** Use print statements throughout your code. Before running the code, think about what you expect to print if the code is working. Then see if what you expected is what prints.

Program Specification (AKA Spec)

For this assignment, you will write a program that plays Mad Libs with the user.

The text you will use for your MadLib is the following:

> Congrats <NAME>!
> You've been invited to a <ADJECTIVE> party! The theme of the party is <NOUN>. Be sure to wear a <ADJECTIVE> costume that screams <X>. The party will be on <MONTH> <DAY>, <YEAR>. We expect <NUMBER> people, so please bring <Y> <FOOD>s so that everyone can have some. We cannot wait to see you there <Z>

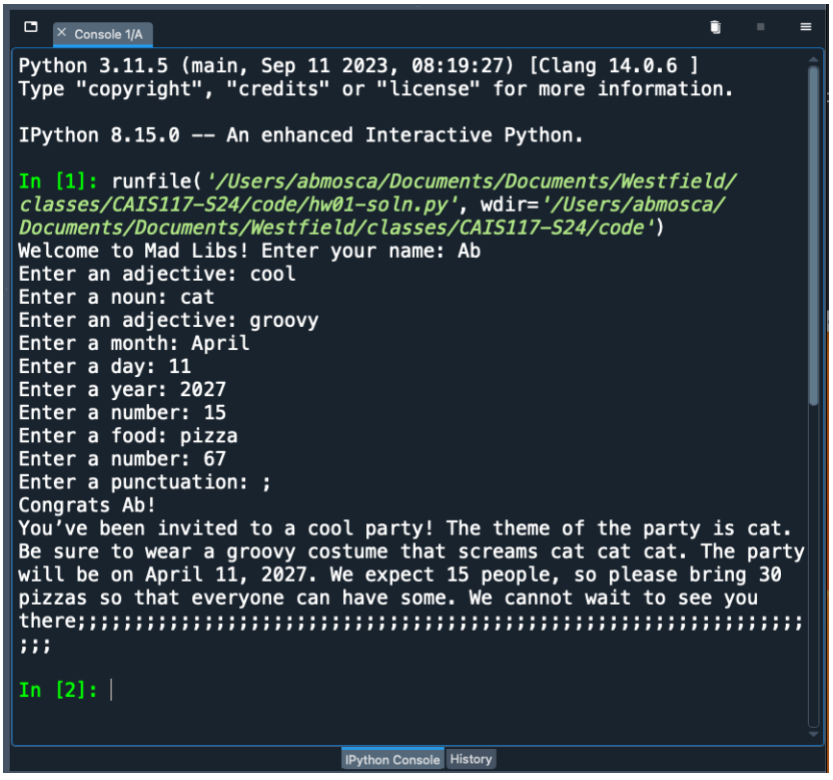
Note the spacing above. "Congrats <NAME>!" is on one line, followed by the rest of the text on a single new line.

All of the variables surrounded by angle brackets (<>) above should be collected from the user. For example, to get <NAME> you should ask the user to input their name. To get <ADJECTIVE> you should ask the user to input an adjective.

For <X>, <Y>, and <Z>, do the following:

- For <X>, repeat the first <NOUN> input 3 times with a space between each
- For <Y>, multiply the <NUMBER> input by 2
- For <Z>, ask the user for punctuation and a number, then put that punctuation that many times

An example of how your program should look to the user is below:



Submission

Be prepared to demonstrate and discuss your code with the class. After giving every one time to work we will come together and discuss:

1. Solutions you came up with
2. Roadblocks you hit along the way
3. What you would do differently if you were to do the assignment again

Curious to know how your code would be graded in a college class? Here's an example rubric:

	Missing / Not Complete (0)	Approaching (2)	Meets (4)	Exceeds (5)
Readability	Assignment is unreadable or not submitted.	Assignment includes formatting, but significant improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment includes formatting, but minor improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment is well formatted and easy to read. Headers, documentation (comments), and descriptive variable names are all included.
Computational Problem Solving	No code is included in the assignment, or the code included is unreadable.	Problem solving approach could use significant improvements. Specifically, better decomposition of the problem, and/or increased modularity.	Problem solving approach is solid but minor improvements could be made with respect to decomposition of the problem, and/or increased modularity.	Problem solving approach is solid. Problem is decomposed into manageable pieces and code is modular.
Implementation	Nothing has been implemented, or most of assignment has not been done.	Code does not run consistently or efficiently. Some outputs match expected outputs. All parts of the assignment are completed except for a few small parts.	Code mostly runs consistently and efficiently. Most outputs match expected outputs, and all parts of the assignment are completed.	Code runs consistently and efficiently. Outputs match expected outputs, and all parts of the assignment are completed.