Tatum Hawkins

LIS 452 – 4 credit hour

Final Project

Github: https://github.com/tatumhawkins/LIS452-Final-Project

Files:

- All Chapters.zip - the individual chapters of the text, full text taken from free online version
- Chapter 1 Practice.txt – chapter 1 text used for practice at the midpoint check, works with Vocab Quiz.py
- Email.jpg – screenshot of the email response received from the textbook author granting permission to use his text – he included some suggestions and I did use the one to allow for selecting a specific chapter rather than always just the whole book
- Final Vocab Quiz.py – my final code for this project, saved to be ready for someone to use for vocab practice – see the note at the end of this document for making it evaluator-friendly
- The Plan.md – note about what I'm doing in this project along with the list of steps I had planned at the proposal stage
- Vocab Quiz.py – the portions of the code I had prepared by the time of the midpoint check, works with Chapter 1 Practice.txt

Midpoint

The guts of this came together pretty quickly and I had them ready for the midpoint check. Isolating the glossary was really similar to what we did to isolate the table of contents in Dracula. Since this book has the word on a single line and the definition of the next single line, it was actually easier to get those lists than in Dracula. After a couple of missteps I realized I'd have to process the word and its definition in the same step of the loop because it was messing with the counters otherwise. Initially I thought that I would need a list of the tuples of (word, def), but it turned out that was extraneous. Anything I would have used it for could just as easily be done with the separate lists since the indices were coordinated.

I did have to learn a little about the random module for this. I could spend hours playing with that! There's a lot of things in daily life that I like to have randomization for, and there's a good chance that if I never to need to write another program again I will still come back to PyCharm just to write programs to randomize information. I know it can do a lot more than I used it for here, but I didn't want to use anything that I wasn't totally sure what it was doing.

For the midpoint check I used a single chapter that I chose and got the framework for the quizzing part going. Since I have no idea how to evaluate text for "linguistically equivalent," I only have the option to show the definition and ask the vocab word/phrase. Going the other way around would be way more annoying for the user as they'd have to supply the exact wording of the textbook definition.

The correct spelling is also required because I can't even imagine all of the possible spelling variations that could be entered. At the time of the midpoint I had it up to quizzing over a single word. Then it would say that the response was correct or that it was incorrect and supply the correct answer.

Final

I knew .pop could do something helpful! It took me multiple days to figure out *what* exactly and it felt like a major epiphany when I finally got it. I was struggling with picking a random word for the quiz and then removing it from both the words and defs list. That ended up changing the length of those lists and then everything had to be reset after each round of the quiz. It just seemed excessively busy and brutal to the lists. I finally figured out that I could make a masking list of just the indices and select from there. Then the value being used for each round could be popped out and the original words and defs lists would be static. This fixed a bunch of issues I had earlier. At various times I had screwed up the indices every time I picked a word or I was always removing the last word regardless of which one was chosen (so it was entirely possible to get the first word for all 26 rounds in chapter one).

The .extend method was a little bit of fanciness that I wandered over earlier in the semester and had a hell of a time finding it back for this. Since I couldn't remember the name of it, just the idea, it took a while to find. Until then, I was accomplishing the same thing with a little accumulator loop that just appended numbers to the indexing list.

My other *big* issue was the order of the loops. I managed to nest them inside out (and sideways and upside down). Things were repeating, but not the things I wanted. I was asking for the chapter every round and remaking the lists every round, but only asking one vocab question in each round. There is definitely something to be said for making each piece individually and then putting them all together at the end, but for me, it's much better to map it all out on paper first and then code from top to bottom. (I know that's not what we were supposed to be learning – I do write the chucks one at a time! I just have to do everything in order to have anything make sense.) Since I tried to just get the big chunks done for the midpoint, I ended up tangling up all of the processes when I went to put it all together.

Also, the power of defining functions! I did have this set up with over 600 lines of code at one point. I couldn't figure out how to change the chapter name, so I had a 29 part if-else structure with individual versions of the vocab game in each one with just the chapter number changing. Line 8 was a full day's thinking and reading and planning. Sad, but true.

I'm still finding it a bit odd to declare all of the definitions at the top and then have the motor at the bottom. I think that's just a difference of it not being compiled (that's what I gather form internet wisdom at least). Since I was doing C++ the last time I tried to learn programming, this still seems a bit upside down to me. This way does feel more like a mathematical proof though. After declaring all the variables, constraints, and theorems, the proof is usually just a few lines. Having the motor of the whole program be one input and one three-branch if statement feels almost anti-climactic considering how much thought has gone into this thing.

I did not end up including the part about finding additional words from the text from possible inclusion in the quiz. It was super messy and ended up making way more work for the user which seems like a bad idea for a “flash cards” style program. I think if I were to change/add anything, it would be to give the user an option to add extra words to a chapter quiz if they wanted to before the quizzing actually starts. It would take from sort of sentinel-loop I think. So they could keep adding words/defs until they ran out. Then append those lists to the lists from the glossary.

Note When you’re running the program, you can un-comment line 67 to see what the correct word is for each round. Also, there are several chapters with fewer vocab words so it’s easier to fiddle with. I used chapters 9, 10, and 23 a lot. (I did tweak chapter 12 a bit from the original copy/paste – the format code portions comes over really awkward.)