Taurean Liss

13 October 2020

Geospatial Programming

Week 3 Notes and Homework

Chapter 5 reading

5.1 – Floor division and modulus is something I learned about on edabit.com. After looking ahead to the first exercise I know that I will need to use these to complete it.

5.2 – 5.6– I have encountered Boolean expressions before Also, I am very familiar with using logic like this. I understand how fast this can get complicated though. Most of the programming games I have tried tend to focus on mastering these types of commands.

5.7 – Nesting conditionals is one of the concepts that seem obvious to me. The biggest challenge is keeping track of all the branches of the “tree”.

5.8 – Recursion has always been difficult for me to wrap my head around is usually about where I get the most frustrated. It is one of those things that probably makes more sense the more you just practice writing code down.

5.9 – This doesn’t seem to hard and might make understanding recursion a little bit easier.

5.10 – This makes it seem like infinite recursion might be something you have to avoid at all costs. It makes me wonder if there is a circumstance where it might actually be helpful.

5.11 – I was waiting for when this would come up. Trying this in IDLE made me realize this was pretty intuitive way to input keystrokes.

5.12 – This section describes one of the more confusing aspects of programming for me. When I first started fooling around with programming on my own, I became very discouraged by the types of error messages that I got. My main goal in programming is to not get so discouraged by this and to try to learn and intuit the meaning behind most error messages I get.

Chapter 5 exercises

5.1 - This exercise was not too difficult. Note: the book does not specifically say if the time given since the epoch is in seconds or not, and I had to independently verify that. The math on this exercise was pretty basic multiplication and division.

5.2 – This was a pretty straight forward problem. I had some difficulty getting inputs to be entered into the functions correctly but that was mostly due to me being new to the syntax.

Chapter 6 reading

6.1 - I understand what fruitful functions are a little better now. I have used “return” a lot on edabit.com because most of the beginning problems revolve mastering it. I can see that this will definitely become useful.

6.2 – This is well-written tutorial on working on code that ends up being complex but starts out very simply. I like how easy it was to follow.

6.3 – This builds off the last sections nicely. It doesn’t really teach anything new but brings everything else together.

6.4 – This is not exactly new to me. I find it interesting how minimally you have to write out True/False statements most of the time so that you do not actually need to write True or False.

6.5 – This section expands on 5.8. Also, this is not the first time I have seen factorials used to explain how recursion works.

6.6 – Just a bit of helpful advice involving trusting that certain functions are working correctly and how that might help when using recursive functions.

6.7 – Just another recursive example.

6.8 – This section describes a powerful tool to check what type of argument is being used called ‘ininstance’. Also, the concept of guardians is described.

6.9 – Some more good strategies for checking work and making sure that the current arguments are being used.

Chapter 6 exercises

6.3 -

1. The ‘middle’ function does not return anything for two or one characters or empty quotations. In retrospect, this was probably the biggest hint in order to successfully write a a recursive function for part two of this exercise.

2. I was not sure what the book wanted here because it did not really explain lists yet until the first part of this exercise. I know from elsewhere that the easiest way to do this is by checking to see if the reverse of the list equals the original list by doing this :

word == word[::-1]

The book does not specifically ask for a recursive function, but I tried to write one anyway. I took me awhile to realize that the easiest way to accomplish this was to use functions from the first part of 6.3. I tried to write it as an if/else statement at first until I realized that was making it more difficult for myself. I looked at the solution and instantly realized what I was doing wrong and decided I could not come up with a better way of doing it than what was in the book. I played around with the books solution and decided that it looked better with the previous functions taken out and combined into the main function.

GitHub - <https://github.com/Tliss1984/Geospatial-Programming>

I did not use my northeastern ID to make this account.