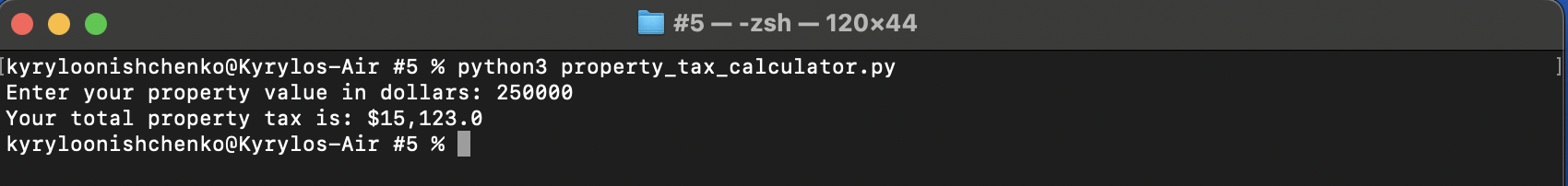
**Part 1:**

***property\_tax\_calculator.py***

******

In this program, I created two function one for calculating assessment value of the property and the other for calculating property tax itself. When get\_property\_tax function gets executed, it invoke get\_assesment\_value function inside itself in order to return a value and then manipulate produced value further in the calculations. As a result, that invocation serves as a nesting mechanism where one function may call the other function inside itself. That works perfectly as you try to structure you program by defining each specific function for each specific operation with the program.

**Flowchart *property\_tax\_calculator.py***



**Part 2:**

***calories\_calculator.py***

***Text

Description automatically generated***

This program involves defining two functions concatenated and invoked inside one main function. First function is get\_fat\_calories which calculates number of fat calories based on inputted value and specific formula and produces a number of fat calories consumed by the user during the day. Second function is get\_carbohydrate\_calories which calculates a number of carb calories based on inputted value and specific formula. The output of this function is number of carb calories consumed by the user during the day.

**Flowchart *calories\_calculator.py***

******

**Part 3:**

***revenue\_generated.py***

***Text

Description automatically generated***

This is a cool program that calculates incomes from basketball game from each specific tier. I know that you can write this program with if-elif-else conditionals, but I decided to implement dictionary mechanics that replicate the real-world behavior of an application like this, since they follow similar structure as real databases such as MySQL that contains of key/value pairs and produce values based on the requested key. Even though we do not learn all of Python the data type in this class (at least not in six weeks one), I decided to implement dictionary mechanics because it requires way less repetitive code to implement within each function as opposed to other approaches. Nevertheless, I also focused on debugging this application in order to prevent users from crashing it, determining most uncommon behavior patterns that may lead a program to traceback.

**Flowchart *revenue\_generated.py***