Introduction

Assignment VI consisted of three LAB lessons and one extensive section of code that used the concepts of classes and functions.

**CDInventory.py**

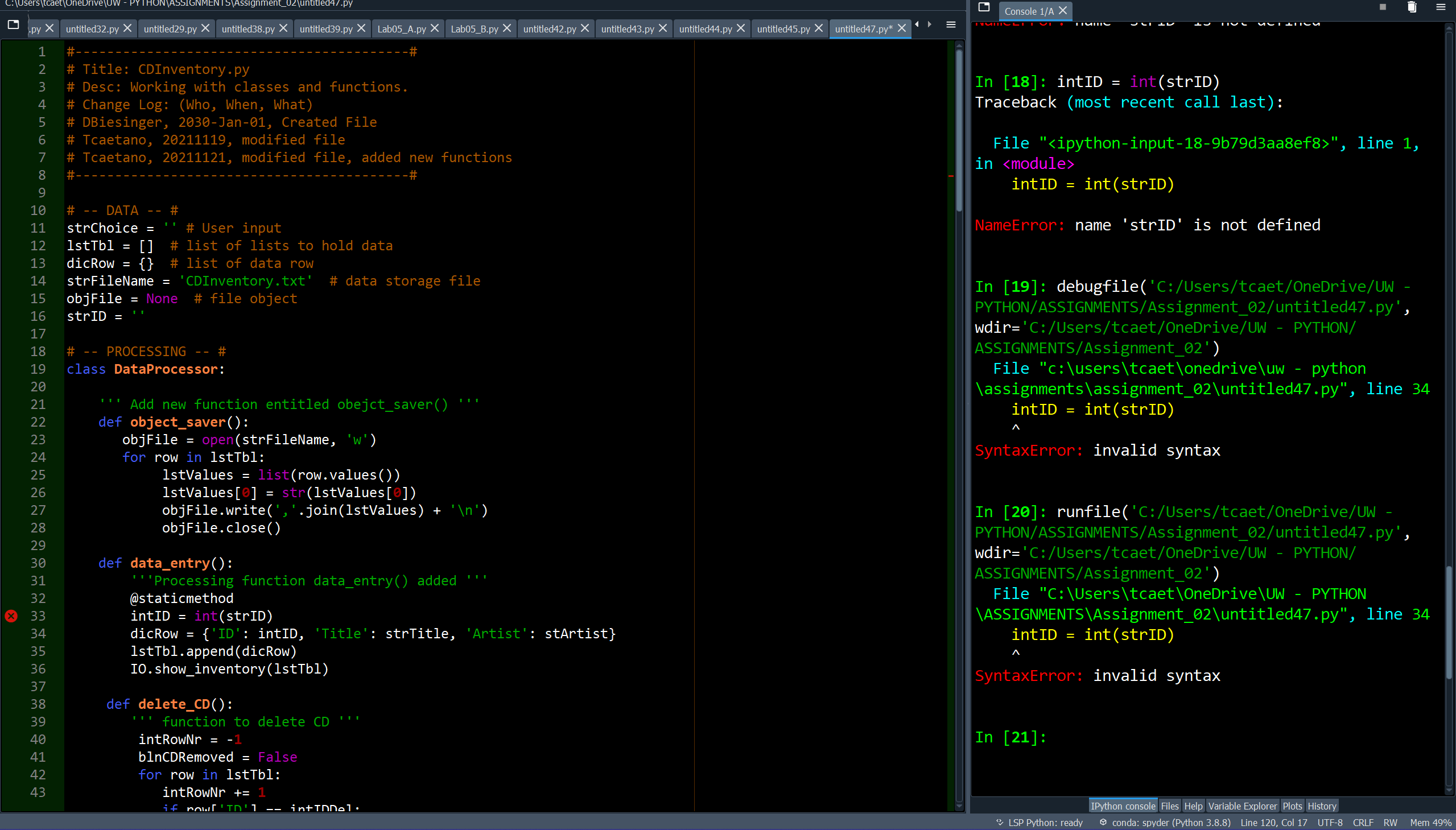
*I do not like to fail but this assignment has been a failure. I have tried repeatedly to make this work, but it has been overwhelming. The instructor’s comment that working with someone else’s code can be challenging, it taken for all it’s worth. I was not expecting to have such a large programme to modify – I much rather would have preferred it to be something that had to be done mostly from scratch especially since this is programming in infancy.*

*Adding classes to this made it very complicated and perhaps they could have been eased in a bit. I don’t think that it is effective to have #TODOs that involve shifting large portions of code around, but I could be wrong.*

*The CDInventory from last week albeit difficult to grasp in the beginning, finally began to make sense and the use of functions finally clicked in my head.*

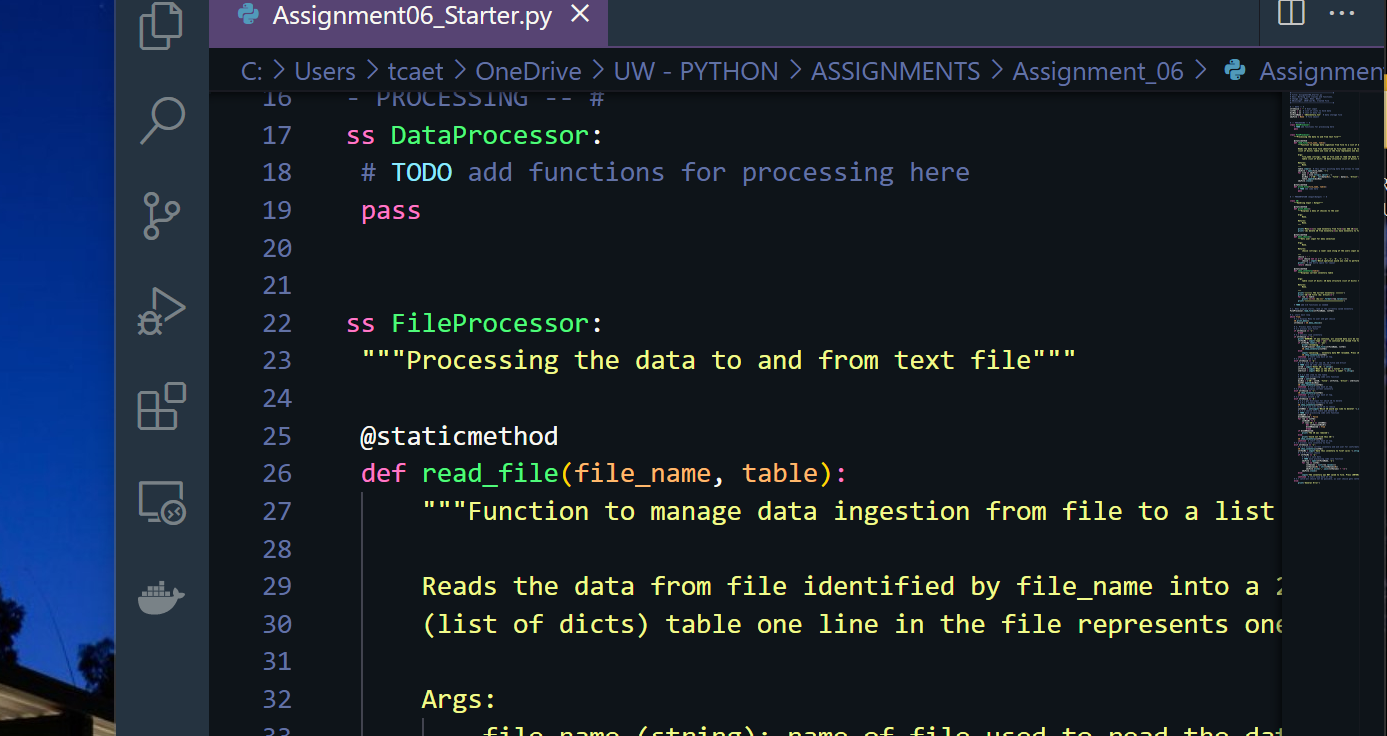
*This assignment has not – Are future assignments based on this code? I sincerely hope not.*

*I have attempted to follow the instructions given in the #TODO sections and I have created new functions that include that code.*



Note: Whilst only **In [21]:** is shown on the console, it is not a true indicator or the entries on the console – I had to restart Spyder.

*Let me give you another example of something I find confusing – Please notice the image below:*



*There are classes entitled DataProcessor and FileProcessor and then a comment which reads:*

* *“”” Processing the data to and from text file “””*

*DataProcessor and FileProcessor sound the same to me. Data are files and files contain multiple datum which are data. The inherent difference between these two classes is vailed - the choice of words is cryptic or ambiguous in my mind.*

*The comment on processing data to and from text file could have been re-written – it took me a while to understand that you meant the .txt file saved with the data entered by the user. Perhaps using the definite article would have helped but then again, I am being too picky. I have three foreign languages stirring my brain and I get thrown off by the way English writers explain a concept sometimes.*

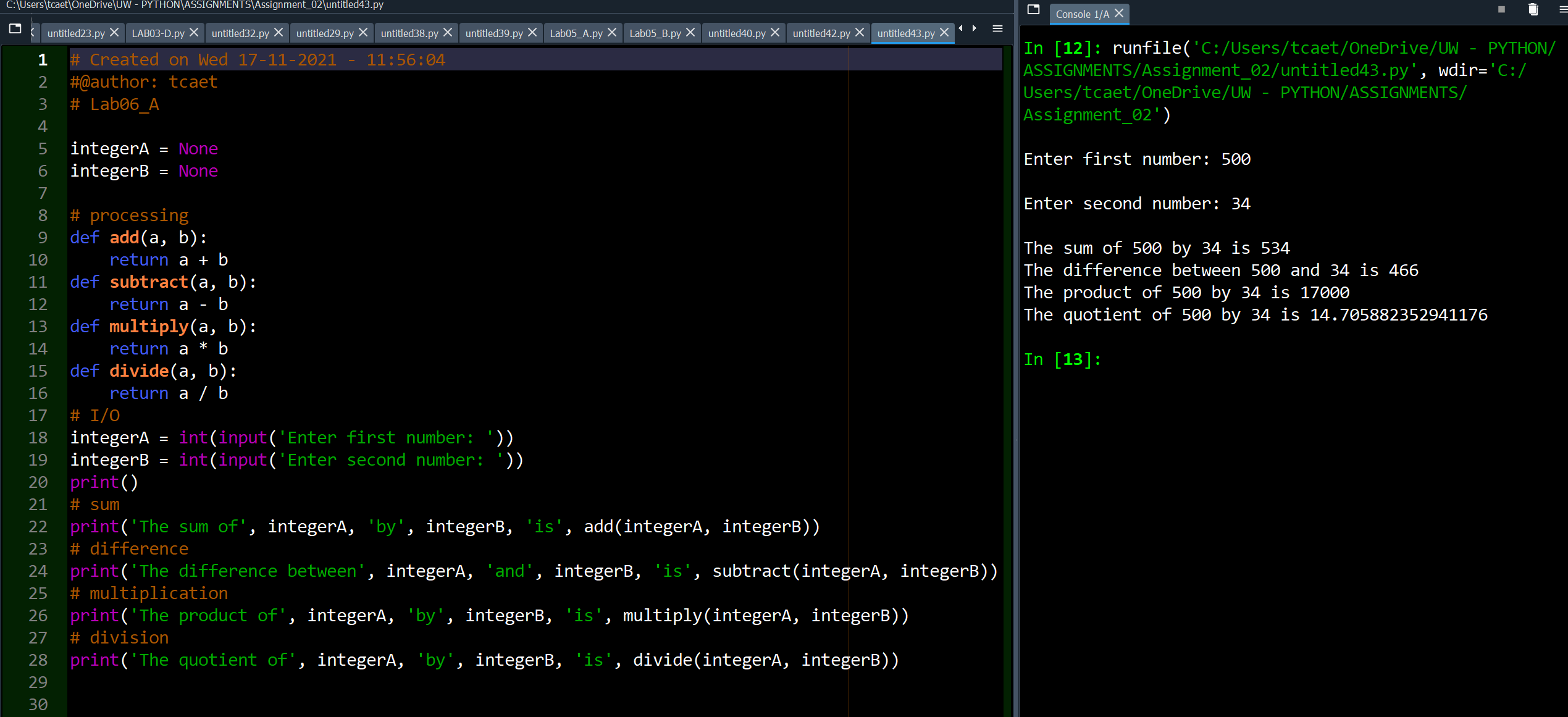
*Ultimately it is my failure to understand what is being asked that has prevented me from making this code work.*

*The LABs were very useful and I learnt quite a bit by playing with the code.*

**LAB 06-A: Working with functions:**  
In this LAB, you create a function each for calculating the Sum, Difference, Product and Quotient of two numbers. Use  
attributes to pass the values into the functions. And use return values.  
• Create a script called Lab06\_A. Add the code from listing 2 above.  
• Modify the script to work with attributes and return values.  
• Test the script and write down how the code works.

*This was confusing to understand in the beginning until I finally realised what the DEF var (x, y) does to the rest of the code. The variables a and b in the defined function aren’t showing up in the code below #I/O – I thought that these vars would have to show up again. The vars a and b are defining a set of rules to be followed later in the programme. Took me a while to figure this out despite all the literature I went through. Why can’t these bloody programming books and websites just say that… and then move on to more technical definitions?*

*Once I call the function multiply(), the rules set in the definition apply to any var inside it’s brackets – genius, I love it.*

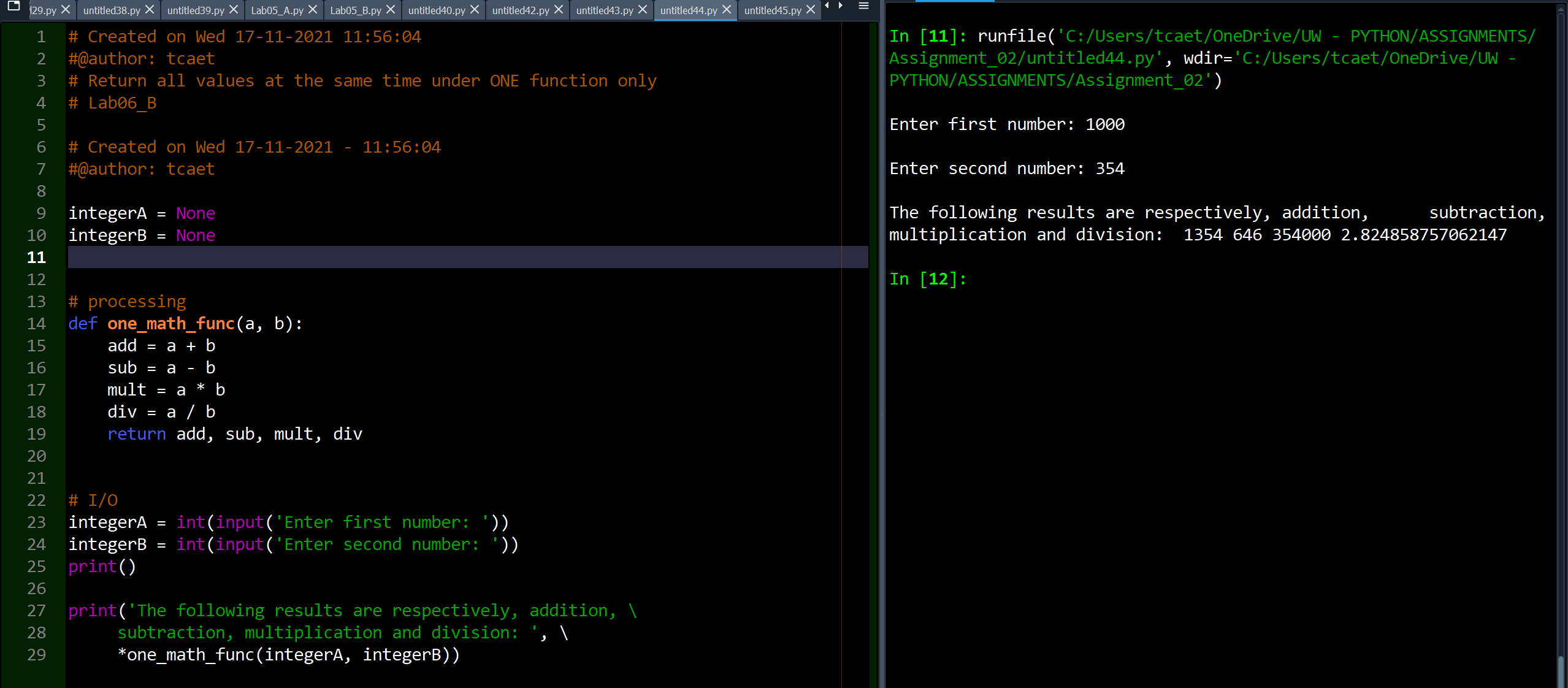


**LAB 06-B: Returning Tuples**  
In this Lab, you create a function that returns multiple values.  
• Make a copy of the script from LAB 06-A and save it as Lab06\_B

• Modify the script so all four calculations are done in one function. This function returns all four calculations results at the same time.  
• Test the script and write down how the code works.

*This one is still a bit of a challenge – I was able to define one function for all the operations as requested. The results are a tuple which shows the correct information – the formatting nevertheless is subpar. How to I get to display the results line by line? I will keep trying.*

*The function one\_math\_func() has two variables a and b. These two variables are given a set of mathematical rules through the creation of new variables which attach four maths operations to them. These four new variables, add, sub, mult and div are now part of ONE function only. When calling two new inputted into this function, variables a and b assume those values which are under the rules set on the one\_math\_func().*



**LAB 06-C: Creating a Class of Functions**  
In this Lab, we’ll continue with the code from LAB 06-A. You’ll create a class that has our four functions for doing the  
calculations.  
• Make a copy of the script from LAB 06-A and save it as Lab06\_C.  
• Add code and docstring to create a class called SimpleMath  
• Add code and docstring to create a function get\_sum  
• Add code and docstring to create a function get\_diffference  
• Add code and docstring to create a function get\_product  
• Add code and docstring to create a function get\_quotient  
• Add code for the I/O to capture user input and format your output that it resembles below figure.  
• Test the script and write down how the code works.

*This was very fun to figure out. It’s interesting to see a CLASS created that includes multiple functions that can be called through the CLASS. When programmes get excruciatingly large, calling classes will be of value as it “pulls out” a whole section of code that may be rather complex. The example below is a simple one, but it exemplifies well the use of a class.*

*As in the example on LAB-B, this programme could have worked by simply calling the functions. This would be accomplished by indenting the last three print statements at the bottom one tab to the right and deleting the CLASS SimpleMaths. The use of a CLASS, nevertheless, shows its usability for larger projects.*

