**Background**

Complete the following exercises. Your answers must be submitted via Canvas. Answers to Questions 04 and 05 may be typed in this document or handwritten and scanned. Answers to Questions 01, 02, 03, and 06 should be submitted **in their own .py file which is named as specified in the question**.

Note that for each code section, you may assume that the in-code documentation is accurate and reflects the both the correct approach and desired design for the piece of software.

**Section 1: Using Documentation for Design by Contract**

1. Consult the file pdf\_tools.py included with this assignment. This file contains multiple functions which can be used to manipulate PDFs. The function Delete() has been fully documented. The functions Split() and Merge() have had their functionality documented at a high-level but not with respect to Design by Contract. Document the functions Split() and Merge() in a similar style using the following guidelines:
   1. Label each precondition, post-condition, or invariant individually
   2. Split complex conditions into multiple preconditions, post-conditions, or invariants
   3. Any use of a variable name should start with an @ symbol (e.g.: @L or @pages)

If you want to run the code, you can download course\_catalog.pdf which has been included with this assignment and can be used to see specific behavior. **The file containing your solution to this question should be named Answer01.py and points will be deducted if it is not.**

**Note**: This is not a standard documentation style. However, you may experience several different styles during your career and you must follow whatever standard is laid out for you.

**Section 2: Using Exceptions for Design by Contract**

1. Using your Answer01.py file as a basis, alter the file to implement the necessary checking for implementing a Design by Contract using Python assert statements. Make sure to do this with all three functions (Delete(), Split(), and Merge().) Be sure that error messages are populated appropriately and descriptively. **The file containing your solution to this question should be named Answer02.py and points will be deducted if it is not.**
2. Using your Answer01.py file as a basis, alter the file to implement the necessary checking for implementing a Design by Contract using if statements. Make sure to do this with all three functions (Delete(), Split(), and Merge().) Be sure that error messages are populated appropriately and descriptively. **The file containing your solution to this question should be named Answer03.py and points will be deducted if it is not.**

1. Consider your implementations in Answer02.py and Answer03.py. Which implementation do you feel is better for this specific code base? Why? Defend your analysis using software design principles from the text or other sources.

**Note:** There no “correct” answer to this question. Either could be argued. I am interested in your analysis and defense of your analysis.

I believe that the implementation in Answer03.py is better for this code base. This is because the code is easier to read with being able to distinctly see what error message is being raised and also the reason way easier than assert statements. This implementation also allows the use of other Errors like TypeError and FileNotFoundError which can describe the problem a lot easier than a custom error message or just an AssertError. It also allows for multiple errors to be raised at once which allows for better error checking then the implementation in Answer02.py

1. Consider your implementations in Answer02.py and Answer03.py. Is this level of defensive programming warranted for this module? Why or why not? Defend your position.

I would say that the level of defensive program in both answers is warranted for this module. This is because this script manipulates files on the computer which, if handled incorrectly, could cause the module to potentially delete or override important files on the computer. This level of defensive programming also allows for easier bug checking by raising error messages which makes it easier to debug any issues with the code without the potential for small unnoticeable errors to come up.

**Section 3: Using Exceptions and Resources Responsibly**

1. Using **either your Answer02.py or Answer03.py as a basis**, refactor your code so that exceptions for file handling are handled more appropriately. Be sure to capture only the necessary lines within your try statement, run subsequent lines only when appropriate, catch only the appropriate exceptions, and print out an appropriate error message as part of the exception handling. **The file containing your solution to this question should be named Answer06.py and points will be deducted if it is not.**