Python Midterm & Final

The midterm and final are a chance for you to show me what you can do with what you have learned. The midterm will help you prepare a project and the final is your project. Please read the directions carefully as not following them will reduce your final score. Any questions or confusions should be directed to me at [broge2@uis.edu](mailto:broge2@uis.edu) or on the forums.

# Due Dates

**Midterm**: June 26th before 11:59PM Central Standard Time

**Final**: July 22nd before 11:59PM Central Standard Time

# **Midterm**

For your midterm I would like for you to think about a final project and give a description in a document file about your project. The document should be language agnostic. In other words, it should be a description of your project without being tied to any one language. The document must contain the following information

1. What is your program? Give a description of what your program is going to do.
2. How is it going to work? Describe how you expect the program to perform its operations.
   1. List out the functionalities and how the user will interact with such functions.
   2. Consider the users experience. How do you expect the user to interact with your program? How might the user break your application and what can you do to help mitigate that.
3. How will the data be represented? You don’t have to know how to handle the data yet but do think about what kind of data you expect to be working with.
   1. For instance, if you are working on a GPS application then you would work with longitude and latitude as well as time as a major part of your application. This is data.
4. Describe any problems you may experience and how you plan on handling it. It is good to think about what might happen, so you are prepared to handle it.

The midterm is to help you with the final project so don’t feel that you need to stick with how you planned to do something in the document if you discover a better way of doing it. If you have a hard time deciding on a final project, I can supply you with one. Also, even though we haven’t covered the material listed in the final project you can still include a general overview. For instance, we haven’t covered what a JSON object or a CSV file are so just say that you will save the data to a file that will be used to save x, y, and z data.

This document is known as a Software Design Document and it get it as crazy as you want. For some guidance, here are a couple of references you can use to help you write the document.

* <https://www.freecodecamp.org/news/how-to-write-a-good-software-design-document-66fcf019569c/>
* <https://www.toptal.com/freelance/why-design-documents-matter>

## What you must turn in:

The document file. It does not need to be a final draft. Whatever you turn in, I will give you some guidance on your project (with one exception). If I think you have too many features, I will ask you to reduce the number of features. If I think you don’t have enough features, I will ask you to add more. If I think your document file was put together at the last minute, I will send it back with no explanation and a bad grade (this is the exception.). Take time to think about your project and be proud of it. It is your work after all.

## Where you will turn in:

On Canvas.

# **Final**

Your final project must follow the specified requirements for full credit. The requirements are listed in the following table.

|  |
| --- |
| Choose at least one:   1. A GUI based project – The project uses a GUI frontend for user interaction. 2. A CLI based project – CLI stands for command line interface. The project uses the command line for user interaction. (Command line is either your Command Prompt in Windows or Terminal for Linux/Mac) 3. Both. You can have a command line and a graphical user interface. |
|  |
| Your project must have some form of data persistence.  Choose one:   1. Save data to a CSV file. 2. Save data to an SQLLITE database 3. Save data to a JSON object. 4. Another method you feel is more appropriate. |
|  |
| Your project must use a third party module not installed with standard Python. You can use as many as you would like to make your life easier. Examples of third party modules include:   * Matplotlib * BeautifulSoup * BioPython * Paramiko * Pandas * Numpy * Qt   Anything not part of Python’s standard library. |
|  |
| Your project should not be thrown into one Python module (A single Python script file). Break your project up into modules so you can easily debug it. |
|  |
| Use docstrings to describe your classes, functions, and methods. |

You may also include a form of networking or web scripting but is not required.

## What you must turn in

There are three things you must turn in. The first is of course the program. The second, is a video link of you demonstrating your program. I don’t care how you record your demonstration and where you upload it. You can upload to YouTube and share the link or use some other media. Do not send me the video file. We have a space cap for emails and video files are normally too big to be sent. The third piece is an updated version of your midterm which contains updates you performed while working on your project.

## Where you will turn in

Send me your project file, a link to your video, and your updated midterm through Canvas.

## Grading Rubric

Since the project is one of your decision, coming up with an objective way to grade becomes challenging. I have decided to break up the grading into multiple parts discussed here.

Grading your final project will be broken up into three parts. The parts to be considered are the code structure, functionality, and polish. This covers 3 aspects of software engineering.

### Code Structure

Splitting the code into appropriate modules, classes, and functions that makes sense is a tougher task that should be practiced more. This makes writing code easier for others. The use of UML diagrams is often used to organize these various aspects and helps direct your thinking and structure of your code. This will keep you on track. The other part to this includes programming patterns. This could be an entire class on its own. To help you, you can reference

<https://sourcemaking.com/design_patterns>

which provides common patterns and examples in Python. You don’t need to utilize programming patterns in your final project but knowing about them and using them will make your code structure much better.

Use UML diagrams to plan and programming patterns for extensibility, modularity, and ease of programming.

### Functionality

Does the program do what it is supposed to do. I will be going through each feature you wrote and testing with various inputs. You should do the same. Try to consider what a user might try to do or accidentally do to catch errors. Have someone sit down and use your program with basic instructions and see what they do on their own. Tell them to try different inputs and make sure to record what they do so you can go back and assess how to handle those various inputs if they caused problems. Any features you include is fair game to scrutiny and grading so if you have a feature that is incomplete and it’s not an important feature then don’t include it into your final solution if you don’t have the time to get it working.

If you are writing programs that don’t work, then no one will want to use your programs including yourself.

### Polish

This one is tougher and can be a bit more subjective. It is important, however, so I have broken this down even further. Polish in this case will consist of user experience, graceful error handling, and program feedback.

User experience is an entire topic that classes have been developed around. Often shortened to the initialism UX by software developers this is one of the most important aspects of software development. Consider using your operating system. If you had to put in a proper boot sequence by flipping switches every time you wanted to use your computer then you would probably not want to use that operating system anymore (unless you are an enthusiast or an early computer time traveler). Think about the software you use every day. Why do you use that particular software and not the competitors? What about it makes it easy for you to use? What negatives would you say that software has in terms of your experience using it? Does it take too many clicks to get to a feature? Is the user interface not well organized? Is the font too small or too big? I’m sure you can come up with more. User experience defines what software we use today and how well adopted it will be by the masses. Consider user experience while working on your project.

Graceful error handling means not crashing when a known error is thrown. Sometimes, software does crash but it should be for extreme or unforeseen cases. Common cases like missing files or division by 0 should be handled gracefully and allow the user to continue using the software without restarting it. Sometimes, these common errors turn to major issues such as the 1970 glitch on iPhones. Here is a video explaining this issue: <https://youtu.be/MVI87HzfskQ>

Program feedback is also important and could be included as part of user experience, but I separated it as it is something people don’t often think about. When you click a button on a webpage if you didn’t see any animations via your browser or in the application itself you would suspect nothing occurred and click again even if the program is doing something. Providing feedback to the user is important when they perform an action. Whether you write a GUI or CLI application, you can provide some form of feedback for processes that take time like a loading bar, a percentage indicator, or an animation indicating the user’s request is being worked on. Little things like this are often missed but are important for a well-received application.