IS430 – Instructions for the Final Project

What are the Final Project goals?
The goals of the Final Project include:

- Make a start on a lifetime of self-directed programming projects that help you achieve your professional and personal objectives.
- Reinforce the Python programming skills learned during the course.
- Develop greater confidence in doing programming work on your own.
- Explore learning new Python skills needed for your project.
- Develop practical experience in controlling the scope of a programming project so that you produce a viable product by the deadline.

What kind of project can I choose?
You are free to choose from a wide range of topics that meet the project goals. Your project might address professional interests, personal interests, or even entertainment. I have provided links to resources in our Weekly Schedule that might help with identifying potential projects that might be of professional interest to MSLIS program students. I have also provided a link to my playlist on LinkedIn Learning for Python project books. Many of the projects described in these books are likely to have either a personal or entertainment interest. Please feel free to come to lab and discuss your project ideas and get some feedback and coaching from me.

How big does the Final Project need to be?
The size goal for the project is that it should contain the same amount of work as 2 coding assignments from the latter portion of the course. Most of these assignments had 4 or 5 exercises that involved small to medium-sized programs. By contrast, the Zelle Chapter 11 assignment had 2 exercises and involved a bigger, more complex, game simulation program. These examples should give you a good idea about how big your project should be. Please feel free to speak with me if you are having difficulty applying these guidelines in the case of your project.

Please be aware that your project concept could easily lead to a project that is considerably bigger than the appropriate size for a Final Project. In that case, you will need to do some planning to select the portion of the work that you will address in the Final Project. The remainder of the work would be left for you (or someone else) to do after the course is over. Projects might be divided by the product features included in the scope of this project, or they might be divided by the datasets that are processed within the scope of this project. It is okay for your Final Project plan to represent only the first step of a multi-step process that could be implemented in a series of projects. Please feel free to speak with me about ideas from choosing the appropriate scope for your project.

How should I design my software?
The design of the software that makes up your Final Project should follow the software design approaches that we used for coding assignments during the semester. If there is functionality that is likely to be used in more than one of your programs, create a common function that you can place in a library module, import the function into each program that needs it, then call it. If there are data records that need to be sorted, create a custom dataclass to hold the data while you are sorting it. If there are practical advantages to creating automated Pytest unit tests for your code as an alternative to manual unit test, then consider taking advantage of that opportunity as well. At one point, I considered making Pytest automated testing the challenge portion of this assignment. In the end, I decided to choose another challenge instead. Nevertheless, I am interested in seeing where you find practical advantages to using automated test code in your project.

At some point, you will need to decide how the functionality of your project solution will be broken up into runnable programs. During the semester, we explored two alternative approaches to this that you will want to consider adopting:

1. **A Pipeline of Processing Steps**: Many projects will work well when designed as a pipeline of simple processing steps. The design of these projects is likely to resemble that of the Jupyter Notebook coding assignment. Each processing step can be implemented with a program that you code and test on its own in PyCharm. Finally, you can create a Jupyter Notebook to document and implement the overall production workflow.

2. **A Complex Stand-Alone Program**: The design for other projects may be more like the game simulation program that we created in the Zelle Chapter 11 coding assignment. While you might place some of your code into a library module, the user will be running just one program from the command line. While these projects might include a Jupyter Notebook to hold documentation, the user will probably not be executing the code from the Jupyter Notebook. Instead, they will start it from the command line (or the PyCharm simulation of the command line that we see when we run programs with PyCharm).

As you might imagine, the design choices are many. Please feel free to discuss alternatives with me as you work on the design your project solution.

Will I need to learn further Python skills?
You might. At this point, you have learned basic Python, and you are well equipped to learn more. Many attractive projects might require you to master new Python packages (from the Python Standard Library or from third party providers). A good example of the kind of libraries that you might decide to use are those that were covered in the Severance Chapter 11 tutorial on data organization. When your project includes learning more Python, it is both more rewarding and bigger. It is more rewarding in that you are growing your capabilities. Yet, the learning workload makes your project bigger. You will need to manage all the demands of your project to make sure that you reach your main goal within your deadline. From the perspective of finishing your project on time, learning more Python is a side goal – a worthy goal – but a

side goal. The trick is to sign up for just enough new Python learning to allow you to tackle an interesting project without over-committing.

What if I need to change direction?
You can expect that you will need to do some direction changing during the project. Because this is a creative endeavor, your ideas about the details of what you are creating are likely to evolve. Also, it is difficult to foresee all of the obstacles that will emerge along your path. The goal is to get to a good destination that meets the requirements for the Final Project. It is okay (and even expected) that the exact destination will shift over the life of your project. This often leads to tough choices during the project. Please know that I am available to help coach you through those choices.

What should be included in the deliverables for the project?
Like the coding assignments that you submitted during the semester, the main deliverable for the Final Project assignment is a zipped-up PyCharm Project that you will submit to Canvas. To make your work on this easier, I have created a starter version of that PyCharm project that you can download from the Weekly Schedule. The starter file is:

- `surname_givenname_is430_final_project.zip`

Unzipping the starter file will yield the following PyCharm project directory:

- `surname_givenname_is430_final_project`

Using the Refactor > Rename feature of PyCharm, change the name of the directory and the project to include your actual name rather than the model text provided.

The starter version of the project includes two features that you should retain in your project and complete:

- The `data` subdirectory
  Please place your data in this subdirectory, following the practice that we used on most of our coding assignments.

- `main_final_project_notebook.ipynb`
  This notebook provides a structure for the written part of your Final Project submission. By filling out the notebook sections, you will provide us with the information that we need to fully appreciate your work and evaluate it fairly.

  The sections in the notebook contain instructions that are set in *italic type* to guide you through completing the document. Feel free to delete these *italicized* instructions when you have finished authoring your own content. That should help to make the submitted version of the notebook more readable.

To what extent do I need to follow specific coding practices in this project?
In the Final Project, I will be expecting you to follow the good programming practices that we have adopted in the course. Here is a quick summary of good practices that we have covered:

- Include a Python Docstring that describes the intent of the program.
- Place your highest-level code in a function named *main*.
- Include a final line of code in the program that executes the *main* function.
- Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function and the code surrounding that function.
- Choose names for your variables that are properly descriptive.
- Define CONSTANT_VALUES and use them in place of *magic numbers*.
- Always use f-strings for string interpolation and number formatting.
- When processing items from Python lists and tuples, unpack the values into variables with meaningful variable names to avoid using indexed expressions in your code.
- Open all files in a WITH block to assure that they are closed before the conclusion of the program.
- Remember that your program should behave reasonably when it is not given any input. This might be the result of the user pressing enter at a console prompt. Or it might be the result of the user providing an input file that is empty.
- Model your solution after the code that I demonstrate in the tutorial videos.
- Create a sub-directory named *data* within your PyCharm project to hold data files.
- Remember to submit all data files with your PyCharm project – including the files that were provided as starter files to this assignment.
- All functions that are not *main()* should have descriptive, action-oriented names.
- All functions should be of reasonable size.
- All functions should have high *cohesion*, and low *coupling*.
- Remember to test your program thoroughly before submitting your work.
- Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by *if, else*, and *elif* conditions in your program (boundary value tests).
- Use of the *break* statement is allowed but not encouraged.
- Use of the *continue* statement is forbidden.
- Regular expression patterns should be expressed as Python *raw strings*
- Your finished code must be refactored to meet all good program design practices covered in this course.
- When needed, custom Python classes should be created using Python Dataclasses using the approach demonstrated in our course.
- Where it creates a practical advantage, manual unit tests should be replaced by automated unit tests created with Pytest.
- Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
- Python programs that are called from a Jupyter notebook should not prompt the user for input. Instead, configuration values should be set in a notebook code cell, then passed explicitly as parameters to a called Python function.

- Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. This may be achieved using either manual unit testing techniques or by using Pytest automated unit testing.
- Unit testing scenarios tested directly in PyCharm should not be repeated in the testing of the Jupyter notebook. Testing of the Jupyter notebook code should test the workflow that the notebook implements.
- Jupyter notebooks should not include extensive Python code placed in the notebook code cells. When Python code in notebook code cells grows beyond the nature of configuration code, it should be placed into a .py library module file, imported into the notebook, then called from the notebook.
- The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Which technology tools should I use for the Final Project?
Use the tools that we used for coding assignments that we did during the semester. These include Anaconda and PyCharm. If you need help identifying further tools to use during your project, please contact me to discuss alternatives.

How do I submit the Final Project?
Follow the process that I demonstrated in the tutorial video on submitting your work. This involves:

- Locating the properly named directory associated with your project in the file system.
- Compressing that directory into a single .ZIP file using a utility program.
- Submitting the properly named zip file to the submission activity for this assignment.

How should I name my Final Project submission files?
The starter files for this assignment were discussed earlier in these directions. Per those directions, please rename the PyCharm directory and project to include your name. Also, there is one Jupyter Notebook included in the starter files. While you are expected to edit that file to describe your project submission, please do not rename the file.

Consistent with the earlier instructions on PyCharm project naming, your PyCharm project name should have the following form:

```
surname_givenname_is430_final_project
```

If this were my own project, I would name my PyCharm project as follows:

```
trainor_kevin_is430_final_project
```

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

```
surname_givenname_is430_final_project.zip
```

If this were my own project, I would name the zip file as follows:

```
trainor_kevin_is430_final_project.zip
```

When is the Final Project due?
Please submit this assignment by the date and time shown in the Weekly Schedule.

Last Revised
2025-11-12