Final Project

As part of completing COGS18, you must complete an independent project. This final project is worth 25% of your grade for the course.

Objectives

The broad objectives of this project are for you to:

- Choose a topic that you are interested and work on code related to that idea
- Be able to plan out what is required for a computational solution to your chosen topic - Design and write an implementation of your project plan in Python code
- Practice following best practices for coding style, documentation and code testing
- Work with different coding tools, including using Jupyter notebooks and Python files

Project Schedule

- W7: Project Description released
- W8: By the end of this week, you must have a project topic / outline
- W8-10: Work on the project
  - Coding Lab during Weeks 9 and 10 will provide you with time to work on and get help with your Final Project.
- **Finals Week: projects due (11:59 PM, Wednesday, December 11th)**

Getting Started

We strongly encourage you to discuss potential project ideas on Piazza, and with your TAs and IAs, and with the instructor through office hours and coding labs. This will give us a chance to give you feedback and help guide your project plans.

Suggestions for working on projects:

- Work consistently
  - Slow and steady is a much friendlier way to develop code than a last minute sprint If you get stuck, pause, and ask for help
  - Seek advice when you are unsure, and seek it early and often
- Talk to course staff and each other about projects
  - Ask questions on Piazza
  - Ask questions to course staff
  - Visit office hours
  - Talk to other students about projects, and help each other

Project Topics

For your project, you can choose a topic that extends an application from one of the assignments, or propose your own topic.

Possible topics:

- Encryption (A2)
- Chatbots (A3)
- Artificial Agents (A4)
- Data Analysis

- Choose your own adventure (propose and develop your own project idea)

There are more detailed project ideas in the `ProjectIdeas` notebook.

Taboo Topics

While you're allowed to work on a topic of your choosing there are two "taboo topics" - or topics that are not permitted for the COGS18 final project. These two topics are off the table for two reasons: (1) There are a lot of examples of these out on the Internet. While you're allowed to borrow code from elsewhere, we want to ensure that you get the chance to implement code on your own and aren't limited by what others have done and (2) It gets boring to grade these when we get a whole bunch of them that all look super similar.

This quarter, the Taboo Topics for the COGS18 final project are:

- Hangman
- Drawings from Turtle

Now, if you have a great or interesting idea that puts a new spin on Hangman or are super interested in doing something with Turtle, reach out to Professor Ellis and ask for an exception. You'll have to explain your idea. And, if approved, you'll be able to work on your taboo topic of choice!

Project Scope

The main goal of the project is to demonstrate that you can design and implement code to fulfill some task that you choose. That is, the goal of the project is to plan and write some code that solves a novel problem, one for which the solution is not pre-specified (as compared to and different from the assignments).

There is not a specific amount of code that you have to write for the project. In terms of project scope, your project must implement some new thing, that you design and write the code for. To do so, you are expected to write new code that creates or adds some functionality, and that does so using good code practices. As a rule of thumb, note a project that appropriately responds to this call will have at least a set of new functions, will use code constructs such as loops and conditionals, will import code from available modules when needed, and will do all of this organized into a well constructed organization, in which the code follows proper coding style, is documented, and includes at least one code test.

As the focus of the project is working with Python code, there is **no** prioritization for trying to perform complex tasks. On the contrary, in the face of a complex task in which the work ends up being trying to understand the complex topic you are working on, instead of focusing on working directly with Python code, a simpler project topic is preferred.

Project Approach

As you work through your project, we recommend taking an approach in which you use a modular design and **prioritize creating a minimal viable product**, using a rapid prototyping approach. These terms will be more fully described in class - but to a first approximation this approach suggests that you organize your design as separate components (modular design), that are independent of each other. Focus first on the core component of the project (minimal viable product) - what piece of it is required such that you have a working implementation of the basic idea that you are targeting. As you work through this, try things out quickly (rapid prototyping) - write some code, check if it works, figure out why or why not, and iterate onto a next version. Once you get the core of the project working, you can add any extra pieces that would be nice and useful to extend the scope of the project, knowing that you have the core idea working.

Project Requirements

To complete the project you must submit a zip file containing your project, which will include Python code (3.6 or 3.7) that implements your project. Your project must include (at least) one Jupyter notebook and one python file (a module file and/or a python script). The Jupyter notebook file will include a description of the project, organized in markdown cells.

The actual code for the project can be organized into either the notebook, and/or into python files, as either module files and/or scripts. Exactly where and how you organize the code can and should be decided by the project topic itself. For example, projects that are more focused on functions and classes may be more organized into a module organization. For some projects, the code may be well organized into a script that executes your project. For other projects, with more interactive components, much of the code may be presented within the Jupyter notebook.

Projects are individual, and each person needs to submit a project including (at least some) unique and original code. Projects can also include code that comes directly from the assignments, and/or from working with other students on project ideas, but this should be clearly marked and attributed in your submissions. More details on external code are included below.

Submitting Your Project

You will be submitting a zip file containing your project to TritonEd. This zip file must contain a directory structure of the form:

MyProjectFolder/

- ProjectNotebook.ipynb
- my_module/
  - functions.py
  - test_functions.py
- scripts/
  - my_script.py
- requirements.txt

Note that not all of these files / sections are mandatory, and they do not have to use the names specified here - file names can be whatever you want, as long as the organization is clear and what kind of thing each file is is clear. **You must include, at minimum, a Jupyter notebook, and some part of your code should be organized into a module, with a test file**. The scripts section is optional, and should be included if it makes sense to organize your function into an executable script. The requirements text file (optional) should list any modules you use in your project that are not included in the anaconda distribution.

How to create a zip file

First, make sure everything is organized into a folder, as you want it. Then:

- On Mac: left click on the folder, and click 'Compress'.
- On Windows: left click on the folder, go to 'Send To' and select 'Compressed (zipped) folder'.

In both cases, this will create a zip file, with the name 'FolderName.zip'.

You can upload this file to TritonEd as your project submission.

Grading Rubric

Component	Grade Value
Concept	5%

Component	Grade Value
File Structure	5%
Project Description	10%
Approach	20%
Project Code	30%
Code Style	10%
Code Documentation	10%
Code Tests	10%

Concept (5%)

Your project is directed to fulfill some task that meets the requirements of the project specification. Note that concepts are *not* graded on complexity. The goal of the project is to choose a task that you can complete in code, and you should choose and work on a task that is interesting to you. As long as your idea is responsive to the project outline, as described above, it will be considered valid.

File Structure (5%)

Your project includes a clear file organization, and includes at least one Jupyter notebook and one python file (not including the test file), organized into a directory structure.

Project Description (10%)

Within the Jupyter notebook, there is a description of the project. This description should be written in human language, and provide a self contained description of the project. This can be organized as, for example, a couple paragraphs of text, and/or bullet points, etc.

Approach (20%)

The approach you designed and implemented to create your project meets these requirements:

- You use relevant modules (from standard library and external sources)
- The algorithms / approaches used are well chosen for the task at hand
- The algorithms / approaches chosen are properly implemented

Project Code (30%)

The code you wrote to implement your design will be graded on the following requirements:

- Uses appropriate variable types (int, floats, strings, lists, tuples, dictionaries, etc)
- Uses appropriate code constructs (operators, conditionals, loops, error handling, etc)
- Code is organized into functions (necessary), and classes (optional)
  - You do not necessarily need to use classes (only if they make sense for the project)
- The code executes, without raising any errors, on expected inputs
- There is a clear modular organization to the code

Code Style (10%)

Structurally, we will be evaluating the following criteria:

- Uses blank lines to separate code elements and logical structure
- Uses good indentation & spacing
- Has 1 statement per line, and max line length of 79 characters

In terms of naming, we will be evaluating the following criteria:

- Uses descriptive names
- Follows naming conventions
- CamelCase for class definitions & snake_case for variables, functions and methods

Code Documentation (10%)

Documentation will be evaluated in terms of:

- Includes docstrings (numpy format) on all custom classes, functions, and methods
- Includes in line comments that explain the code

Code Tests (10%)

You must include at least one code test in your project, organized into a test file, in a test folder, and that uses pytest. This test should be a unit test of one of your custom functions and/or classes in your project, and should test that it executes, and test (at least some) expected outputs of the function, given some specified inputs.

External Code

Your project may include external code - that can be code from the course (assignments, materials, and/or coding labs), code from fellow students, that you perhaps worked on together, and/or code from other sources. Note that this code is not considered original code for the project, and will not be graded. Please follow the following guidelines when doing so.

Course Code

You are welcome and even expected to take code from the course, in particular code from the assignments.

Code co-developed with other student(s)

You may work with other students on aspects of the project, if you are working on similar topics. For any code that was developed together, you can include this code in your project, and indicate that it was written together with another student. Note that each student will be graded primarily on the unique code that is submitted as part of their project.

Third Party Code

You may include third party code, external from the course, including demo code and available snippets of code you find available.

Provide Attribution

Clear attribution of code is important. Including external code, that you did not write, but failing to indicate that is code you did not write and passing it off as your own is (and will be considered) plagiarism. To cite code indicate in the docstring for the module / function / class or code segment that this code is external code, and include the source of the code.

Modified Code

It may also be the case that you originally start with code from an external source, but through your project you modify that code - perhaps refactoring it, reorganizing the code, and/ or editing and updating the code organization and functionality. If this is the case, also provide attribution and a link to source, but if you significantly update the code organization and/or functionality (refactoring the code, and/or updating functionality more than just updating naming, style and documentation), then you can and should indicate that this adapted code is to be considered part of your graded project code.