COGS18: Project Grading Rubric

- In general, we are not aiming for particularly harsh grading. If they spent time and effort engaging in the project, they should get a passing grade.
- Sections aren't all or none - unless they totally didn't do something, give them some proportion of the section, matching the extent to which they did or didn't do the required thing.
- Keep in mind the students' context. Though you might know a better way to do something, it is not necessarily fair to expect them to know that. Keep the course materials in mind in terms of evaluating what they should know how to do.

Concept (total: 5 points)

Points	Requirement
5	They chose a project topic from either the assignment list, or an independent topic they can write code for

File Structure (total: 5 points)

Points	Requirement
1.5	There is at least one Jupyter notebook file
1.5	There is at least one python file
2	Files are interpretable organized (not necessarily same as template)

Project Description (total: 10 points)

Points	Requirement
2.5	There is a project description in a Jupyter notebook
7.5	The project description is a self-contained explanation of the project. It provides a description of what they are trying to do.

Approach (total: 20 points)

Points	Requirement
5	The design they chose responds to the project topic they outlined
5	Original code meets project requirements (3 functions; 2 Classes w/ 1+ method; 2 functions + 1 Class)
10	The code / approaches / algorithms they used for their project are appropriate for the task they trying to complete

Code (total: 30 points)

Points	Requirement
5	Variables & Code Constructs: they use variables and control flow constructs in their code, as needed

Points	Requirement
5	Functions & Classes: code is organized into functions; Use of classes is optional
5	Modular Organization: at least some code is organized into an external module, and this file is well constructed
5	Imports & Libraries: they import code from the module, and use external libraries if & when appropriate.
10	The code is functional - it does what it is supposed to, with no major errors or bugs, and works as intended.

Style (total: 10 points)

Points	Requirement
2.5	There is good code layout - spacing between code segments, etc
2.5	There is proper indentation and spacing
2.5	Names used are descriptive
2.5	Names follow the right naming conventions (snake_case, CapWords)

Documentation (total: 10 points)

Points	Requirement
7	There are numpy format docstrings on functions & classes Needs: first line sentence, parameters & returns sections
3	There are (at least some) inline comments that explain the code

Tests (total: 10 points)

Points	Requirement
2.5	Has asserts that are used to test some property of the code
2.5	These are organized into a well formed test file
2.5	Tests are organized into well formed test functions
2.5	The tests execute

Fudge points (up to 10 points)

Use these to offset projects that are strong but lose points due to the rubric. Explain why these were granted in the comments section. You can award up to 10 points to offset points lost to the rubric when you feel a project was strong. This should NOT be used as extra credit. I anticipate these being applied sparingly. Most projects will have 0 fudge points. Project should not be >100 points after applying fudge points.

Extra Credit (up to 4 points)

Awarded for going above and beyond. Students should have an explanation of how they went above and beyond in their notebook to be eligible for extra credit (however, if this is missing and you feel their project is very strong, feel free to award extra credit). This could include using packages/modules not discussed in class, learning something new, or generally going well beyond the requirements of the project. The max possible project score should be 104.