# Project 6: Final Project Spec

Version 1.1. Last Updated: 10/29/2021
Due: Wednesday, 11/30/2022 at 11:59 PM PT

**Contents:**

**Submission Guidelines:**

1. **Download your README as a PDF**. **.pages and .docx files are NOT accepted, NOR are Google Drive or Docs links.**
2. Download your project code as an .xml file (Snap!) or package all the elements (.py files, images, sounds, and anything else used by your project) as a .zip file (Python).
3. Rename the .zip or .xml file *Partner1_Partner2_FinalProject* if you're working in a pair. For example, NancyPelosi_AngelaDavis_FinalProject would be a valid project name.
4. Upload **BOTH** your README (PDF) and project file (.xml or .zip) to the Final Project assignment on Gradescope

**Recommended Timelines**

- The date listed below is the **suggested deadline** for each component of the final project
- This timeline assumes that you are done with proposal review on 11/15. However, **it is recommended that you start working on the project ASAP after your proposal is approved**

***Groups with 3 Features:***

Feature 1: Friday 11/18
Feature 2: Tuesday 11/22
Feature 3: Friday 11/25
Full game coded: Tuesday 11/29
README: Wednesday 11/30

***Groups with 2 Features:***

Feature 1: Friday 11/18
Feature 2: Wednesday 11/23
Full game coded: Tuesday 11/29
README: Wednesday 11/30

## Introduction

For your final project, you'll be turning in both code and a README. The README, which is a document that describes your project and its features, is meant to be an assignment where you'll write up documentation to practice explaining your work to someone else. This project is a chance for you to implement an idea driven by your own interest — best of luck!

## Technical Requirements

There are certain technical and style requirements that we want you to be able to demonstrate in the implementation of your final project, regardless of what kind of project you're doing, which are listed below. For some of the rubric items, you can also mix and match (see rubric for more details). Here is a list of some of the technical requirements you must include:

- **Non-trivial list:** Your project should include at least one non-trivial list, meaning a list whose function could not be simply replaced with another implementation. Often this can be a tracker of high scores, or part of a game board.
- **Custom block/function:** Your project should include at least one block or function that you define yourself. There is no restriction on what type of block this should be, whatever suits your project is what you should do!
- **Set of two (2) tests:** Your project should include at least one set of test cases for one of your custom functions. A "set" for us means two test cases with expected outputs and accompanying comments explaining why those are important test cases to pass. Remember that test cases should return True (to indicate successful functionality). It might be easiest to test reporter blocks or functions that return some value, but if you're testing functions that don't return/report, set up some inputs to call the function on and then indicate how the result demonstrates the expected behavior.
    - *Snap! projects:* Make sure to use the testing block included in Project 3 for reporter or Boolean blocks.
    - *Python projects:* Refer to this video for how to write your own tests. If you're stuck, ask in OH!
- **Script/local variable:** Your project should include **at least one non-global variable**.

## Style Requirements

It's best to keep up good style (commenting, specific variable names, non-repetitive code) as you work your way through implementing a project, but that doesn't always happen. This is a list of what style markers you should aim for as you're coding, but we also recommend that you do a final sweep for these before you submit!

- **Commenting** — At minimum, we're looking for clear and coherent comments for the following purposes:
    - *Describing how to start/run the program:* Often, this is the green flag for Snap!, and typing a command into the terminal for Python. Make sure to specify what to do, even if it seems obvious! The readers are not as familiar with your project as you are. **Make sure this is in both your code as a comment and in your README file.**

    - *For each function or block you've written:* An explanation of its inputs + its outputs or what data it changes if it's a command block or does not return anything.
    - *For each test case (not each tested block):* An explanation of what that test case is testing, and what it means for your project overall that your specific test cases pass.
        - Think of edge cases that your function needs to pass, e.g. the merge column block must be able to merge a column of [2, 2, 4, 0] to become [4, 4, 0, 0] without also merging the 4 and 4, so putting the former list in as input for the block's test case and the latter as the expected output should yield true if your merge column function works.
    - Here's a guide to commenting for reference!
- **Variables & functions** —
    - *Variable scope:* Make sure you're only using global variables when absolutely necessary (i.e., when they need to be referenced at multiple points in the program, or by multiple functions). Try to use script/local variables where possible.
    - *Naming convention:* Keep function and variable names clear, by specifying what they do or represent. A good rule of thumb is to give things names in a way that would still make sense to you if you needed to explain this project to someone else in a year. For example, a variable could be called "highScore" if it keeps track of the highest score so far.
- **Abstraction & generalization** — These are key computing ideas for a reason! Do your best to apply these principles as you code. Often, this will look like one of the following:
    - *Generalizing code*: If you can use a for-loop or a HOF or a helper function to prevent copy-pasting similar code, do it!
    - *Levels of abstraction:* Do your best to break down functions or features into smaller, concise pieces, so that each function can be focused on achieving a specific task or subtask.
        - If you find yourself with repetitive code, consider how you can abstract the part that is repeated.

## README Requirements

Your README is meant to serve as a guide to your project, and will be used by the readers as a reference for grading to ensure that you've achieved all of the requirements. Instructions for the general sections of the README are included in the template. An additional **optional** section, the Bug Writeup, is explained in detail below.

**The template (including instructions) for the final project README is here.** Please make a copy of the Google Doc to work on, and make sure to submit it as a PDF.

**[Optional] Bug Writeup:** Sometimes, despite your best efforts, the project deadline rolls around and you've still got a bug big enough that one or more of your features isn't working correctly. This happens in industry as well — think about app updates you've downloaded!

Not to fret, you can earn back up to 50% of the points originally lost for a non-functional feature with a thorough write-up. (Points given will depend on the size of the bug; this is meant to limit the points penalty for not implementing a bug-free project feature.)

Here's an example of what we're looking for in a bug writeup, based on a function you wrote for the OOP lab.

## Submission Guidelines

You should submit two things to the Gradescope final project assignment:

- **README:** This should be written on a copy of the template above, and be submitted as a PDF. We will not accept READMEs submitted as .docx files or Drive links, etc.
- **Project code:** This will be either a .xml file or a .zip folder. Your instructions will vary depending on what language your project is in.

*Snap! projects:*

1. Download the .xml file with your project code as you did for Projects 1-3, using File > Export Project.
2. Rename the file *FirstPartner_SecondPartner_FinalProject.xml* and upload directly to Gradescope. (Adjust accordingly if you are working solo or in a group of three.)
3. Upload the README PDF and the .xml file to the Final Project assignment on Gradescope.

*Python projects:*

1. Put the .py file(s) containing your project code along with any other media (images, sounds, etc.) that are used in the project into a ZIP folder.
2. Rename the ZIP folder *FirstPartner_SecondPartner_FinalProject.zip* and upload the ZIP file to Gradescope.
3. Upload the README PDF and the .zip file to the Final Project assignment on Gradescope. (Feel free to Google "How to make a .zip file" / ask for help on Ed.)

**If you run into any issues submitting, go to a lab or office hours or make a private post on Ed**

## Rubric & Grading

Your project will be graded for style and technical correctness according to the following rubric. After you get your grades back, you can submit regrade requests via Gradescope (if you feel we missed something in your original file submission).

| | |
|---|---|
| **Logistics (10.0 pts)** | |
| +1.0 | README and file/zip folder named correctly: *Name_Name_FinalProject* |
| +2.0 | README contains features description |
| +2.0 | README contains complexity justification |
| +2.0 | README contains variable justification |
| +2.0 | README contains blocks/functions table |
| +1.0 | README or code comments describe program controls |
| **Features (30.0 pts)** | |
| +15.0 / feature | Groups of 1-2, who implemented two (2) features |
| +10.0 / feature | Groups of 2-3, who implemented three (3) features |
| - 4.0 / bug | For each major bug (project-breaking/crashing) |
| - 2.0 / bug | For each minor bug (errors only on specific input or subpart of feature) |
| + 2.0 / bug | [Optional] Bug writeup for a major bug (see README) |
| + 1.0 / bug | [Optional] Bug writeup for a minor bug (see README) |
| **Technical Requirements (22.0 pts)** | |
| + 4.0 | Req. 1: Custom block / function |
| + 4.0 | Req 2: Non-trivial list |
| + 4.0 | Req 3: 2 tests for one custom block |
| + 4.0 | Req 4: Custom block / Non-trivial list / 2 tests for one custom block |
| + 4.0 | Req 5: Custom block / Non-trivial list / 2 tests for a custom block |
| + 2.0 | Req 6: 1 or more script or local variable(s) |
| **Style Requirements (8.0 pts)** | |
| + 3.0 | Useful and coherent comments (-1.0 if incoherent but present) |

| | |
|---|---|
| + 3.0 | No repetitive code / good use of helper blocks or functions |
| + 2.0 | Clear and useful variable and block names |

## Appendix I: Resources

Since this is the first project you're developing completely from an independent idea, we're collecting resources for you to use. When further resources are compiled, we will post them here as well as on Ed, so you'll get an email notification when that happens.

**Python Code:**

- **Testing in Python:** Check out this video, also linked above.
- Further documents made by our course staff about testing in Python coming soon, will be linked here.

**README template:**
https://docs.google.com/document/d/1j2EEyYHTXd-tOzOabpLWfRJvVA5sBMtd0XMeOLYh1gM/edit?usp=sharing

**Bug writeup:**
https://docs.google.com/document/d/1bwjFuaOQwpZ6-vFEffHKzm90nPOH-bFeQyV6iNcEu-k/edit?usp=sharing

**Guide to commenting:**
https://docs.google.com/document/u/1/d/e/2PACX-1vSrtiqOPprVJ327uvCj9aRGXJYGP9hjOMJyz_vSxK237PNnTCYumvE24QYcbmA_Xy8voUPOqHzXtZOO/pub?embedded=true