# Final Project
# 02-601: Programming for Scientists

## Learning Objectives

The homework assignments in this course are designed to help reinforce the introductory concepts of programming. However, because they are weekly assignments, they are unable to delve very deeply into a single idea. The purpose of this project is for you to "take the next steps" as a programming scientist and have the opportunity to design and document a larger, well-documented project as well as to grow comfortable explaining your work to an uninitiated audience via both a presentation and an essay.

## Project components

We expect the following deliverables as part of a successful project. Each deliverable includes the percentage of the project grade allocated to it. See "Important Dates" for due dates; materials can be submitted on Canvas.

### Deliverable 1: Project Proposal (5%)

You should write a project proposal of at most one page. This proposal should:

- clearly state the scientific problem that your project will aim to address;
- explain why your project is interesting scientifically and computationally;
- discuss briefly your approach to achieving this goal and why you think this is feasible;
- identify any data or external resources you will use for the project.

It's fine if some of these change during the course of the project, but the proposal should make an attempt to plan out the project. This will help me help you shape the project to be successful and avoid a scope that is either too narrow or too broad.

### Deliverable 2: In-Person Check-in #1 (5%)

We will require a short (approximately 10-minute) in-person check-in to briefly describe project progress and ask questions related to progress.

### Deliverable 3: Project Progress Report (5%)

By the progress report due date, you should provide a 1-page description of the progress you have made so far on your project, describing what you have done, what you plan to do in the remaining time, and identifying any problems you are encountering.

### Deliverable 4: In-Person Check-in #2 (5%)

We require a second short in-person check-in to briefly describe progress, explain what steps have been taken to resolve the problems in the written progress report, and ask questions related to finishing the project.

### Deliverable 5: Project Presentation (20%)

A short (5-10 minute, depending on time availability) in-class presentation to your peers at the end of the semester explaining what you have done and demonstrating your results. You should explaining your work to an audience that is unfamiliar with your project and demonstrate your results. The maximum length of the presentation will be subject to the number of students in the course.

### Deliverable 6: Final Essay (30%)

A medium-length (minimum 5-page) essay. This essay should start from first principles, explaining the technical background of the scientific problem, the computational problems formulated from this biological problem, and the algorithm(s) needed to solve the problem. It should then include a discussion of the results of applying the computational approach to a real dataset. The essay should be clearly written, self-contained, and contain citations to relevant previous work. Any algorithms that you implement or software that you run should be fully explained on a high level (i.e., do not paste your code into the essay).

### Deliverable 7: Final Code and Coding Demonstration (30%)

A medium-sized (minimum 1000-line) coding project that is well commented, clearly modularized, follows style guidelines, is accompanied by a recorded coding demonstration, and is documented so that it is easy to run. (We suggest not to start coding until your project has been approved.)

You should provide readable, well-organized code as part of the final project. The code should be self-contained and include a readme that is well-documented and allows the user to run the code quickly and effortlessly, with a specification of any additional packages needed. You should submit to Canvas but may want to host your code on Github as well.

You should also provide a short (maximum 5-minute) recorded demonstration of your code. This does not need to cover every aspect of your program but should demonstrate how the code can be used to produce a final result. Provide a link to the code demonstration in your readme.

## Important Dates

- **Sunday, September 20:** due date for project proposal
- **week of October 12:** in-person check-in #1
- **Sunday, November 1:** due date for written project progress report
- **week of November 16:** in-person check-in #2
- **Tuesday, December 1:** presentation slides due

- **Wednesday, December 2, Monday, December 7 and Wednesday, December 9:** in-class presentations
- **Friday, December 11:** due date for final essay with results included and final code with coding demonstration

## Project Examples

The project is deliberately open-ended; you may choose to complete a project on any scientific problem that you find interesting.

### Past Projects

A small sample of potentially interesting topics from past projects are below.

- How does disease propagate through a population under different conditions?
- Can we simulate a greenhouse computationally?
- How do we model shame in evolutionary game theory?
- How can I assess whether to put solar panels on my house?

### Unacceptable Projects

Examples of projects that do not fit the description and will not be accepted for further work would include the following:

- Reinventing the wheel of a standard package, such as writing functions from a `math` package.
- A project that has no immediate scientific scope, such as implementing a video game.
- A project relying too heavily on existing packages. An example would be an R pipeline that does not solve any problem or provide any novel analysis but simply plugs the output of one function into the next.

### Potential Projects

Examples of some potential projects that spring to my mind that could be fun are below (most are from biology). A disclaimer that they may need to be adapted along the way to fit the demands of the project.

- Build your own genome assembler, perhaps one that incorporates data from "Hi-C" data finding chromatin contacts at different points of the genome.
- Build and analyze co-expression graphs from RNA-sequencing data.
- Construct a family of evolutionary trees (perhaps using multiple tree construction algorithms) for a variety of multiple alignments on the same collection of taxa to obtain a collection of "gene trees"; then, design an approach that reconciles these differing trees into a single "species tree" for the collection.

- Evolve cellular automata to "solve" some interesting problem, and reflect on the structure of the solutions (compare to modularity experiment from 02-680).
- Analyze The Cancer Genome Atlas expression data (`https://cancergenome.nih.gov`) to find which expression patterns are associated with differing cancer types.
- Build a simulator demonstrating the birth of stars from nebulae and rigorously examining the patterns formed.

## Group Option

I am willing to allow students to form groups (maximum size of three students) to complete the projects. The written component will be expected to be very thorough, and the coding component will expand by 1000 lines for each additional team member (e.g., a group with four students would need to code at least 4,000 lines).

Note that although I encourage group communication, especially at the planning and coordination stage, it should be quite clearly indicated in the code which group members have completed which components of the code.

Barring exceptional circumstances, each member of the group will receive the same grade for the work completed by the group.

A note: group projects have in the past seemed to require more work to coordinate than completing an individual project.

## Grading Guidelines

The following guidelines will be applied for assessing each component of student projects. These are not a hard and fast rubric but are things to keep in mind as you complete your project.

### Code Expectations

- Does the code have the appropriate length?
- Is the code reasonably sophisticated or is it wandering?
- Is the code designed top-down and partitioned modularly?
- Is each function commented?
- Are there comments throughout the code explaining key ideas?
- Is a readme provided that is easy to follow in order to help the user get the code up and running quickly, including precise commands to run the code?

### Essay: Written Quality

- Is there a clearly written introduction that explains the scientific problem for a lay audience?
- Are all figures clearly explained via captions and referenced appropriately from the main text?

- Is the essay structured logically with a clear flow from the beginning of the article to the end?
- Is an abstract provided that clearly articulates the high-level aims and results?
- Does the quality of the exposition accurately reflect the ability of a master's student in a course in SCS?

## Essay: Quality of Scientific Work

- Does the essay have the requisite length?
- Are any computational problems addressed in the project very clearly formulated?
- Are the key algorithms used explained on a high level for a wide audience without resorting to copying code into the document?
- Is a thoughtful results section included that is interpreted in the context of the scientific problem introduced and whose results are explained without resorting to appealing to a figure or dataset?

## Presentation

**Note:** Some of the items below are taken directly from the"3 Minute Thesis" competition guidelines (`https://library.cmu.edu/3mt`).

- Does the presentation end on time?
- Does the presentation provide an understanding of the background to the question being addressed and its significance?
- Does the presentation clearly describe the key results of the research including conclusions and outcomes?
- Does the presentation follow a clear and logical sequence?
- Does the speaker convey enthusiasm for their project?
- Does the speaker capture and maintain their audience's attention?
- Does the speaker avoid scientific jargon, explain terminology and provide adequate background information to illustrate points?
- Does the speaker have sufficient stage presence, eye contact and vocal range; maintain a steady pace, and have a confident stance?
- Does the speaker spend adequate time on each element of their presentation, or did they elaborate for too long on one aspect or was the presentation rushed?
- Do the slides enhance the presentation? Are they clear, legible, and concise?

## External Resources

We require a project proposal because we want to make sure that you are on the right track. If you struggle with the scientific component of your project, please let us know so that we can help.

To help with your written and oral communication skills, the university has a central service called the Global Communication Center (GCC). You should make use of their services throughout your time at CMU; check them out at `https://www.cmu.edu/gcc/`.