CS 5001 Term Project (Spring 2022 SF)

Term Projects encourage students to further explore and practice key ideas from the course, through authentic, tangible activities relevant to personal goals and interests. Projects also provide an opportunity for students to demonstrate mastery of the material through more compelling, less stressful activities than traditional timed assessments. Humans learn by doing, and by reflecting on the process of doing. Unlike a traditional final examination, creating a project tends to feel more like the career work you hope to soon be doing.

Several elements are required for a successful project:

1. About 3-4 slides, describing the goals of the project, what tools and techniques were used to accomplish it, what was learned, possible future extensions. and citation of work you relied upon. Slides can be created in any commonly used format (PowerPoint, Keynote, Google Preso, or even MS Word, Google Docs, or plain text).

2. Code modules/package implementing a pilot or prototype solution to the stated problem. Python is ***strongly*** recommended for the bulk of coding work, although not absolutely required. (Please obtain instructor permission in advance before writing any extensive non-Python code.) Learning about and importing relevant libraries or frameworks with Python APIs (application programmer interfaces) is often helpful and worthwhile. Attention to style and standards matters; write the code with a view that you might want to share it publicly on GitHub when seeking an internship later.

3. A live, informal **presentation/demo/code review** (from 3 to 5 minutes) to be conducted during Finals week. An audience of at least 2 people (the main instructor, a TA, and ideally 1-2 other people) should be present to observe, question, and critique. Friends, family, and other supporters are welcome and encouraged to join for moral support, but are not required.

   Your presentation should:
   - Use your slides to **summarize** project **goals**, **methods**, **findings.** and **sources**
   - **Demonstrate your program** in operation
   - **Explain** a few key aspects your code
   - **Highlight** how your approach incorporates or illustrates a few **big ideas** and/or **key programming techniques** from the course: decomposing problems into subproblems, computational thinking, recursion, dictionaries, objects, abstract data types, exception handling, etc.)
   - Highlight your ability to **answer questions** about your code

- Include **scholarly references** for work of other people that has been relied upon or incorporated, including libraries imported, snippets of code from tutorials, etc. Every line of code should either be your original work or supported by a citation to an English language cite, book, article, or similar. A substantial amount of your own, original code, is expected.

4. The Scoring Rubric for projects is based on multiple criteria, weighted about equally:
   - relevance to the ideas and skills taught in CS5001-5003
   - authenticity (solving a problem of genuine interest to either your pre-Align field, your CS professional goals, or your outside interests)
   - creativity and originality of approach (It is OK to create your own solution to an already-solved problem, but ***please*** do not merely follow a "YouTube cookbook")
   - appropriate level of ambition (challenging yet doable in available time)
   - completion of working code for a reasonable subset of your project goals
   - inclusion of a suitable quantity and quality of code commentary
   - coding style, including modularity, naming, comment format and placement, and PEP8 compliance
   - quality and clarity of slides or other written materials
   - scholarly citation of information sources (preferably ACM, APA, IEEE, or MLA; but at least – minimally -- URLs to English language web sites)
   - ability to answer detailed questions about both the ideas and your program's operation, during your presentation.

Projects ***must*** be your own work; any other materials and any resources relied upon must be cited and readily available in English. Group projects are not allowed for this assignment.

Choosing a topic is flexible and details may evolve as your work progresses; however, instructor approval of topics, scope, and changes is key, to ensure that your project has the potential to achieve an acceptable score. Learning to define a topic so as to balance challenge and interest versus realism (with respect to existing knowledge and skill as well as calendar constraints), is part of the joy and benefit of graduate school.

Below are some suggestions to help with brainstorming. Promising topic ideas are often mentioned in class and on Piazza, as well. Although projects must be carried out individually, brainstorming topic ideas with other students is encouraged, including seeking information about third party libraries that might be helpful during implementation. Other ideas welcome!

- A short essay, with code examples, comparing parameter passing mechanisms (call-byname, call-by-value, call-by-reference, call-by-object-reference) in 2-3 languages (including Python)
- A simple but *original* chatbot, implemented in Python, such as for first tier technical support or health assessment
- Comparison of how different implementations of Python handle interning of strings, with specific code examples in test files runnable on different versions/platforms to illustrate the differences
- Program for a game. Although Python 3.x is strongly recommended, a case can be made for PyGame, if you can safely find the extra time required to learn additional syntax. Your project could be a game for 2 humans, with the computer's job being to provide suitable graphics and record-keeping, or a 2 player game where the computer serves as a simple "AI" player
- An authentic pilot or prototype "app" related to either your undergraduate major field or an area of personal or professional interest, such as music, sports, art, film, epidemiology, economics ...
- A survey of fractal patterns in nature, illustrated in Python code, using the Turtle library
- An example of image processing in Python, using Pillow, PIL, or a similar, existing library
- A demonstration of the use of Jupyter Notebook to accomplish a fun, authentic task in Python, with discussion of its pros and cons versus a traditional IDE
- A re-implementation of your solution to a previous assignment, but running on a web site, including a description of what was required to make that happen.
- Text Adventure Game
- Program to check file of Python code for poor choices of variable names
- Password checker (see https://learningtech.org/passwordchecker/)
- Tic Tac Toe (heuristically, not ML-based)
- Compare Python to another programming language using various criteria (with code examples), such as:
    - compiled versus interpreted
    - early binding versus late binding
    - call by reference, call by name, call by value, …
    - storage management (stack/heap, reference count/garbage collector)
    - performance versus ease of use
    - expressivity
    - available libraries
    - user community size
- Software project estimation tool
- Program to generate poetry or music
- Explore one of the image processing libraries for Python
- Use publicly available ML libraries to show visual or speech recognition
- Demonstrate the performance of hash tables versus other forms of lookup
- Graphics-oriented game (see *Creative Coding in Python* book)

- Design an algorithm and implement in Python to solve a nontrivial problem
- Evaluation/comparison of 2-3 available graphics packages for Python, such as Zelle: https://www.cs.swarthmore.edu/~adanner/cs21/s15/Labs/graphics.php
- An app to help with care of certain types of pets
- An app to aid in file management on iOS or Android devices
- An LED-lighting control system for a fish tank
- An LED-light show for a band, responsive to sound input
- A Raspberry Pi project, e.g., using micro-Python on the new Pico board
- Connect Python to Discord or Slack APIs
- Web scraper for tickets/stock prices/movies/whatever
- More game ideas: Pong game, snake game
- Exploration of Ren'Py https://www.renpy.org/, to add visual interest to a text-based game