Final Project Description - Programming Foundations with Python

Final Project Description

Step 1: Problem/Idea Definition
Step 2: Solution Design
Step 3: Read the Project Rubric
Step 4: Write Code
Step 5: Write a README file
Step 6: Submit Deliverables
Important Resources

The final project is a place for students who have a paid subscription to this class to demonstrate the ideas learned in the course. Note: This document describes the project for those taking the stand-alone version of the class only. If you are in a Nanodegree, you do not need to complete this project.

If you are in a paid subscription, a coach will be available to help guide you through the project. You will also receive a verified certificate after successfully finishing the final project. If you are enrolled in the Free Course Materials, we would love to see the output of your final project on the discussion forum. Either way, read the instructions below to find out more about the project.

Step 1: Problem/Idea Definition

Begin by coming up with an idea for the project you would like to make. Here are some choices that may help in picking the idea/problem:

1. **The project can be based on an original idea that attempts to solve a problem you care about.**
2. **You can also decide to enhance any one of the projects we have made in the course. The enhancement needs to add significant functionality that goes beyond the scope of the project as it was developed in the course; minor tweaks or repackaging of components will not be sufficient.**
3. **You may also choose to work with a module from the Python Standard Library that we haven't experimented with in the course thus far. Here are a few modules worth considering, complete with links to Python's documentation:**
    a. **csv (reading and manipulating comma-separated values from a file)**
    b. **datetime (what it sounds like, pretty much)**
    c. **smtplib (sending email)**
    d. **re (regular expressions)**
    e. **threading (doing multiple things at the same time)**
4. **Alternatively, you can choose to download and experiment with an external Python package you find interesting. Here are a few external packages to consider:**
    a. **matplotlib (plotting)**
    b. **plotly (analytics and data visualization)**
    c. **requests (http/web activities)**
    d. **numpy (scientific computing)**
    e. **pandas (manipulating lots of data quickly)**

This list is not exclusive. You are free to choose any library you like, even one that is not listed. But you must use at least one standard or external library in your project.

Deliverable

Write a 1-page document (500 words or fewer) that lists out the problem or idea for the project. Note: as a sanity check, you might want to run the problem statement by a coach before doing anything else, to ensure that your project is within the expected scope.

Step 2: Solution Design

In this step, you should define the steps you will need to take to design a solution for your problem statement. No code is necessary at this point. We want you to use simple English to highlight the steps you would need to take to build your project.

Since this is a course about object-oriented programming, your programming solution must demonstrate familiarity with these techniques in the following two ways:

- By instantiating classes from the Python Standard Library.
- By designing, creating, and instantiating new classes customized for your solution. These could extend classes from existing libraries via inheritance, or be entirely new.

In that spirit, we want you to draw class diagrams (much like we did with class Movie, class TVShow and class Video in Lesson 3b) to explain your solution design. You can use Google Presentation to draw this diagram.

Deliverable

Write a 1-page document (500 words or fewer) that lists the steps you would need to take to build your project. Please include class diagrams that highlight the classes and the objects you intend to create in your program. Please ensure that the document is understandable by someone with no programming background.

Note: As you work on your code, you may find that your original solution design needs to be revised. That is perfectly fine. Redesigning your solution as you run into roadblocks or find better approaches is part of the process. In this case, please update your solution design document to reflect any changes. The solution design you submit should be the one implemented in your code.

Step 3: Read the Project Rubric

Go through the project rubric to make sure you understand the expectations for the project. These guidelines will be followed by the Udacity coach evaluating your project.

Step 4: Write Code

Write code based on your solution design (Step 2) to finish your project. Feel free to ask us questions via chat or email.

Deliverable

All your source code, i.e., the Python programming files.

Step 5: Write a README file

Provide all necessary instructions that will enable an end-user to install the required files and run your project.

- The course is taught using Python 2.7. What version of Python is your project written in? If your program needs to be run in Python 3, specify this in the README. If you have used Python 2.7, specify this in your README.
- The course is taught in a platform-agnostic way. If your code will run only on Windows, only on Mac, etc., the README should specify this. For safety's sake, even if you think the code is platform-agnostic, include details regarding the operating system and version you used to develop your project.
- If you used any external libraries, provide detailed instructions on how to download and install the files:
  - If you have used a package manager such as pip, please provide the exact command to be typed for downloading and installing the library.
  - If you downloaded the file manually, provide the URL to the download site.
  - If there are several versions of the external library, be sure to specify which version is needed.
  - If there are any known “gotchas” that the user should keep in mind while installing or running the external library, be sure to include instructions or URLs regarding those.
- Explain how your program is to be run. Ideally, a user should be able to launch it from the command-line interpreter. If your code needs a particular environment, e.g. IDLE or Anaconda, please specify this.

The README file need not be very long, but it should provide all the necessary details that will enable an end-user to set up and run your project as you have designed it. If you do not include such details (e.g., if your project needs Python 3 but this is not specified), then your project may not work as expected for the evaluator. You should also provide the

README in a file format that is commonly available for evaluators to read. For example, you can use plain text or text with markdown syntax formatting, or convert richer text formats into .pdf files.

Step 6: Submit Deliverables

Once you have completed the steps above, follow the instructions below to submit your project.

- **If you are in a paid subscription, zip the following files. Here is some information on how to zip files on your computer. The files to submit are:**
  - **Your problem definition document**
  - **Your solution design document**
  - **All program files**
  - **A README with instructions on how to set up and run your project**
- **When you're ready to submit your project, go back to your Udacity Home, click on the button to submit your project, and we'll walk you through the rest of the submission process.**
- **It can take us up to 7 business days to grade the project.**
- **If you are enrolled in the free course materials, please submit screenshots of the output of your program to the following discussion forum.**

If you are having any problems submitting your project or wish to check on the status of your submission, please email us at oop-project@udacity.com.

Important Resources

Project Rubric
Submission Email: oop-project@udacity.com
Discussion Forum