**Instructions**

**Comment Sections simple enough to use**

The steps for this Challenge are divided into the following subsections:

1. ~~Version Control: Create the Project Repo~~
2. ~~Software Requirements: Translate the Business Requirements into Code~~
3. ~~System Design: Organize Your Code~~
4. ~~Usability: Update the CLI~~
5. Documentation: Update All the Docs

**Submission**

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

# Completed

* ~~The primary application file, app.py~~
* ~~A data folder that contains the CSV file your application uses~~
* ~~A qualifier folder that contains all of the functions imported into the main app, organized into two subfolders:~~
  + ~~filters, which includes .py files for all of your filter functions~~
  + ~~utils, which includes your financial calculator module and your fileio module~~
* ~~A README.md file that explains the purpose of the project and how to use the code~~

**Coding Conventions and Formatting (15 points)**

To receive all points, your code must fulfill the following criteria:

* ~~Place imports at the top of the file, just after any module comments and docstrings, and before module globals and constants. (2 points)~~
* ~~Name functions and variables with lowercase characters, with words separated by underscores. (3 points)~~
* ~~Follow DRY (Don't Repeat Yourself) principles, creating maintainable and reusable code. (5 points)~~
* ~~Use concise logic and creative engineering where possible. (5 points)~~
* ~~Software Requirements: Translate the Business Requirements into Code~~
  + ~~Business requirements translated into code. (10 points)~~
* ~~Systems Design: Organize your Code~~
  + ~~Code organized and integrated into the existing project. (10 points)~~

**HINT**

**Systems Design: Organize Your Code**

~~In this section, you'll refine the systems design. You should consider the overall architecture and design of your system. Specifically, think about the purpose of this function, then search for a location that best represents the functionality, organization, and use of the new code.~~

~~To do so, complete the following steps:~~

1. ~~Integrate your new save\_csv function into the existing loan qualifier application.~~

**~~PRO TIP~~**

1. ~~Update the function and module docstrings for the new feature.~~

**~~Version Control: Create the Project Repo~~**

~~In this section, you'll use the provided starter code to create the GitHub repo for your project. To do so, complete the following steps:~~

1. ~~Create a new GitHub repo.~~
2. ~~Add your README.md and .gitignore files.~~

**~~IMPORTANT~~**

~~Remember to update the README.md file as you develop your code.~~

1. ~~Commit your project files.~~

Remember GitHub best practices: frequently commit and push your updates to the repo, and provide a specific commit message each time. When you’re ready to submit your project, you’ll include your command history to demonstrate that you’ve done this.

**Software Requirements: Translate the Business Requirements into Code**

~~In this section, you'll translate the high-level business requirements into functional code. To do so, complete the following step:~~

~~In app.py, write a function named save\_csv that uses the csv library to save the qualifying data as a file.~~

**Initialization and Design (30 points)**

To receive all points, your project must fulfill the following criteria:

* ~~Version Control: Create the Project Repo~~
  + ~~Challenge repository initialized with starter code and a gitignore file. (5 points)~~
  + A README.md file added and updated. (5 points)
* **User Story**
* ~~As a user, I need the ability to save the qualifying loans to a CSV file so that I can share the results as a spreadsheet.~~

**Acceptance Criteria**

* ~~Given that I’m using the loan qualifier CLI, when I run the qualifier, then the tool should prompt the user to save the results as a CSV file.~~
* ~~Given that no qualifying loans exist, when prompting a user to save a file, then the program should notify the user and exit.~~
* ~~Given that I have a list of qualifying loans, when I’m prompted to save the results, then I should be able to opt out of saving the file.~~
* ~~Given that I have a list of qualifying loans, when I choose to save the loans, the tool should prompt for a file path to save the file.~~
* ~~Given that I’m using the loan qualifier CLI, when I choose to save the loans, then the tool should save the results as a CSV file.~~

**Usability: Update the CLI**

In this section, you'll improve the usability for the nontechnical users. (Then, they’ll be able to access and save their qualifying loans to a CSV file.) You’ll do this by adding a save dialog to the existing CLI. To do so, complete the following steps:

1. In app.py, create a new function named save\_qualifying\_loans that accepts the list of qualifying loans.
2. Inside the new save\_qualifying\_loans function, create a user dialog that prompts the user for whether they want to save their qualifying loans. Use Questionary to prompt the user with .confirm.ask.

**HINT**

1. ~~Inside the save\_qualifying\_loans function, create a user dialog for the function that saves a CSV file. Use Questionary to prompt the user, and ask for the output file path.~~
2. ~~Test your new dialogs to make sure everything is working properly.~~

**~~Comments (10 points)~~**

~~To receive all points, your code must:~~

* ~~Be well commented with concise, relevant notes that other developers can understand. (10 points)~~

**~~Documentation: Update All the Docs~~**

~~In this section, you'll update your README.md file so that it documents the new features of your code. To do so, complete the following steps:~~

1. ~~In your README.md file, add a description and usage examples of the new functionality.~~
2. ~~Generate a text file of your command history. To do this in the terminal, navigate to the top level of your repo, type the following command, and then press Enter:~~
3. ~~history 50 > terminal\_history.txt~~

**~~NOTE~~**

~~This saves the last 50 commands that you entered into the terminal. This includes the history of your commits and commit messages.~~

~~You’ll apply software-engineering best practices to add new features and enhancements to the loan qualifier application. Specifically, you'll create a GitHub repository that includes all of the following files:~~

* ~~A text file that shows your command history, demonstrating that you used version control best practices~~

~~As you move through the Challenge, keep in mind the user story and acceptance criteria that apply to each task.~~

**Requirements**

**Development and Deployment (10 points)**

~~To receive all points, your code must fulfill the following criteria:~~

* ~~Usability: Update the CLI~~
  + ~~Save dialog added to existing CLI. (10 points)~~

**Documentation: Update All the Docs (20 points)**

To receive all points, your project must fulfill the following criteria:

* ~~Documentation included that contains all or most sections from the README.md template. (10 points)~~
* ~~Clear and concise documentation updated to reflect the current state of the project and its features. (5 points)~~
* ~~Rich documentation content included, such as example usage screenshots and code blocks that demonstrate the application. (5 points)~~

**~~Deployment and Submission (15 points)~~**

~~To receive all points, your files must fulfill the following criteria:~~

* ~~Exist in a repo that’s cloned to your local machine. (5 points)~~
* ~~Have been added to the repo by using the command line, as demonstrated by the included terminal\_history.txt file. (5 points)~~
* ~~Contain appropriate commit messages. (5 points)~~