

**Students:**

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

# 11.1 CSW 8 (S22): Final Project Guidelines

## CSW 8 (S22): Final Project Guidelines

Read these guidelines in their entirety before reading the project description.

In this project, you are asked to create a Task Management system, which can help with managing a To-do list.

For this project you will need to submit

- final project code via Gradescope
- final anonymous reflection via the Qualtrics survey

The due dates and intermediate checkpoints are listed below.

---

## Table of Contents

1. [Instructions about Academic Honesty](#)
  2. [Submission to Gradescope](#)
    1. [Required files](#)
    2. [Uploading / backing up to Gradescope](#)
  3. [Due Dates](#)
    1. [Checkpoint 1](#)
    2. [Checkpoint 2](#)
    3. [Final Submission](#)
    4. [Final Reflection](#)
  4. [Late Submissions](#)
  5. [Project requirements](#)
  6. [Grading](#)
  7. [Workflow](#)
  8. [Submission checklist](#)
  9. [Asking for help](#)
  10. [Updates](#)
-

Read these guidelines in their entirety before reading the project description.

## Instructions about Academic Honesty

- This project is supposed to be **your own individual work**. This project is open book, open notes, however, **you are not allowed to ask others for help** (in-person or virtually) or look things up online -- doing so constitutes plagiarism or cheating. A violation of academic integrity on the final project **can result in getting an F in the entire course**, since it puts into question your work throughout the quarter.

We will run an automatic similarity check, which is not easily fooled by the variable / whitespace changes or other modifications.

- Using the material that we haven't covered in this course, which usually indicates that someone looked things up online, is also considered to be a violation.
- Note that if you use online resources, someone else can be copying/modifying the resource that you decide to use, resulting in similar-looking code, which can be flagged for violation.
- **All** students involved in an incident, *regardless if they are copying or sharing their work*, are going to be reported to The Office of Student Conduct.

Please complete your own work (without using resources other than your own knowledge) and keep it to yourself. You should not share your code or answers directly with other students. Reminder that we have a [Regret clause listed in the syllabus](#).

“ By submitting your final project work, **you are asserting that all work on this project is yours alone**, and that you will not provide any information to anyone else doing the project, nor will you solicit help from anyone in the class or on the internet. In addition, you are agreeing that you will not discuss any part of the solutions for this project with anyone. This includes discussing the solutions with others after you finish this course or posting any information about this project on any site on the internet. Doing so constitutes a violation of the academic integrity agreement for this course, which can lead to getting an F in the course. **We reserve the right to change the grade retroactively**, even after the final grades have been posted.

[Back to top](#)

---

## Submission to Gradescope

### Required files

You will not need to submit anything on zyBooks. You will need to submit **three** Python files to Gradescope.

Common Gradescope Questions: <https://ucsb-csw8.github.io/s22/faq/#gradescope>

**Important:** The files need to be in **the same** folder on your computer. The `import` statements in your code will not work correctly, if the files are in different locations on your hard drive.

## File 1 - all your functions

Create a file called **task\_functions.py** in which you will assemble **all** the relevant functions.

You do not need to use `if __name__ == 'main':` in any of your files, because we can import your functions as shown below.

At the start of the **other two files**, you will need to import the functions as follows (`task_functions` is the name of our Python file/module; the asterisk `*` means "all/everything"):

```
from task_functions import *
```

This project's functions will need to be implemented and added to this file.

## File 2 - the main program

Create a new file **task\_system.py**, which will contain your main program. As you will place all function definitions in the **task\_functions.py**, you can import them here, similar to the above import statement.

## File 3 - test code

Create a separate file to test your work. Call it **task\_tests.py** - this is where the `assert` statements will be collected to help you verify that your implementation of each function is correct.

- Important: you need to *run this file separately*: running the main program will *not* execute the `assert` statements.

Each function that returns values will need to have the corresponding `assert` statements to check its correctness.

- If the `assert` statements are failing, then the autograder on Gradescope will fail as well.
- If you are getting an `AssertionError`, then fix the corresponding function - do not modify the `assert` test to pass, fix the code instead.
- Write your own `assert` statement to make sure to **test all edge cases** specified in the instructions: the autograder will be verifying that those work correctly.

You will need to submit **all** files at *the same* time to Gradescope (not in zyBooks).

## Uploading / backing up to Gradescope

- There are required **Checkpoint** submissions, which are described below.
- Do not submit your Final Project via zyBooks.
- The project will be automatically graded for some functions, and manually graded for functionality and style in Gradescope.

Upload the files to Gradescope **all at once** to the **correct assignment**.

- If you haven't submitted anything for this assignment yet, Gradescope will prompt you to upload your files. You can either navigate to your file or "drag-and-drop" them into the "Submit Programming Assignment" window. (You can use Command+click or Ctrl+click, depending on your OS, to select multiple files at once.)
- If you already submitted something on Gradescope, it will take you to their "Autograder Results" page. There is a "Resubmit" button on the bottom right that will allow you to update the files for your submission.

You can re-submit your work as many times as you would like until the deadline.

- **Submit the files and the function stubs as soon as you read the assignment.** See the Checkpoints below.
- Make sure to **periodically backup your work** by submitting it to Gradescope, so that in an unexpected event, we can use your backup, instead of giving you a 0 for a missing submission.
- We will only grade the *latest* Gradescope submission.

[Back to top](#)

---

## Due Dates

The following components will count in your final project grade as follows:

- 1% - Checkpoint 1 grade
- 20% - Checkpoint 2 grade
- 75% - Final submission grade
- 4% - final anonymous reflection via the Qualtrics survey

### Checkpoint 1: due Friday, June 3, before 10PM.

During the labs on Wednesday, create the required files listed above and verify that you can submit them **all at once** to Gradescope. If you run into any issues, let the mentors know during the lab.

### Checkpoint 2: due Sunday, June 5, before 10PM.

Checkpoint 2 assignment will open up on Gradescope as soon as Checkpoint 1 closes.

Upload the requested files shown above with the listed functions added to them.

- Add the functions from previous labs (remember to include their proper documentation and the corresponding code).

- Add the code and the documentation for the `save_to_csv()` function that creates the correct file with the saved tasks.
- Add the function stubs and the documentation for the new functions - full functionality is not required for this checkpoint but you are welcome to include it as well.

## Final Submission: due Tuesday, June 7 before 10PM.

The Final Project assignment will open up on Gradescope as soon as Checkpoint 2 closes.

There will be a **separate Gradescope submission** for the extra credit implementation that is due at the same time as the final project. You will need to add your optional, extra credit functions to the file `extra_credit.py` and submit it together with your final project files.

Remember that you will need to `import extra_credit` in order for those functions to work correctly in the main program.

## Final Reflection: due Wednesday, June 8 before 10PM.

- submit the final reflection via the following link: XXX
  - The survey is anonymous.
  - You should take this survey only once. At the end of the survey, **you will be automatically redirected** to the link where you can enter your 7-digit PERM and email, so that we can give you credit for taking the survey.
  - **Do NOT close the survey window until you submit your PERM and email** -- since the survey is anonymous, we won't be able to give you credit without your data from the redirected link (you can access it **only** by submitting the survey answers).
  - *We will not be looking at the survey responses until we release the final grades.*

[Back to top](#)

---

## Late Submissions

We will not accept any late submissions. This is a hard rule to ensure that we can grade the submissions in a timely manner and submit the final grades before the university deadline.


- Please plan accordingly to make sure that travel, Internet or electricity outages or any other unexpected circumstances or events will not have a severe effect on your submission.
- *start early and backup your work* on Gradescope periodically.

[Back to top](#)

---

## Project requirements


- Programs that have syntax errors in the file *will not receive any credit*.

-  **Test your code for syntax errors before making a final submission.**
- Run it in an IDE *each time you make even a smallest change*: remember that even an extra space can cause an error in Python.
- It is better for you to submit a partially working program than one that has syntax errors. We will not be modifying/correcting your code.
- You are **not** allowed to use external libraries for this assignment except where specified in the instructions. Your project might fail because of an unrecognized `import` (this includes libraries that we have not used in this class). Do **not** `import` anything *unless the project specifically requests it*.
- Do **not** create new functions or change the requested functions' names and/or the parameters, since you may lose points from the rubric set in Gradescope.
- You must **correctly and appropriately comment your code** to clearly explain your solution.
  - **Every function** should include an appropriate, informative docstring *that is included immediately underneath the function signature*. You will be graded on the presence and quality of your documentation.
  - Include more extensive comments for the parts that you found harder to code.
  - There shouldn't be any `pass` or `TODO` or `FIXME` comments left in your code.

[Back to top](#)

---

## Grading

- Your project will be **automatically graded** on Gradescope for the presence and naming of required files, functions, inclusion of docstrings, correct return values, etc. The autograder score will be the majority of the project grade.
- If there are no syntax errors, we will manually grade the final project checking the correct code style (we will be checking the [PEP 8](#) guidelines).
- If your project fails due to syntax errors, we will not manually grade it, automatically assigning it a 0 for the manual score.  **Test your code for syntax errors before making a final submission.**

[Back to top](#)

---

## Workflow

As you are reading project instructions, each time you come across an edge case or a specific value that you need to check/test, immediately add an `assert` statement to your test file. Add a comment above it to remind you what part of the instructions you are testing.

We highly recommend developing your code incrementally: as soon as you add an option or a new condition, immediately run your code to verify that it works as intended.

Use the `assert` statements to check that your code returns the expected value and is of a correct type. Use it especially to test the edge cases (e.g., when to return -1 or 0, what to do when the input is empty).

We highly encourage you to add your own tests to check each function specification and to verify the correctness of your implementation.

If you see an `AssertionError`, check the line number that the error message pointed to. In IDLE, you can select the "Show Line Numbers" option from the top "Options" menu.

[Back to top](#)

---

## Submission checklist

Long before each checkpoint deadline, walk through these steps:

- run your files to ensure that they are free from syntax errors.
- verify that the function names are correct - check them again.
- check that each function has a proper docstring directly underneath *each* function signature
- make sure that the main program only has the necessary `import` and the `if __name__ == "__main__":` block
- check all default values! We will test that you are using them.
- be sure that you test for the invalid 'arguments'/'input values' in each function (use the `assert` statements).
- submit *all* **properly-named** Python files to Gradescope
- submit the final reflection via the [provided link](#)

[Back to top](#)

---

## Asking for help

This assignment is supposed to be like an exam in that it is supposed to test your knowledge and understanding of the course material, so it is supposed to be your own work. Make sure that you read and follow the instructions carefully.

TBA

## Updates

If there are any updates to either the project description or these guidelines, they will be posted below (remember to refresh zyBook pages to see the updates):

- Initial draft of the instructions released on Monday, May 30.

398092.2571084.qx3zqy7

LAB  
ACTIVITY

11.1.1: CSW 8 (S22): Final Project Guidelines

0 / 1



main.py

1

**Develop mode****Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

**Run program**

Input (from above)

**main.py**  
(Your program)

Program output displayed here

Coding trail of your work [What is this?](#)

History of your effort will appear here once you begin working on this zyLab.

[Trouble with lab?](#)