CodeCamp Programming Project

Project requirements

Input data

- Turbie parameters and C_T curve.
- Turbulent wind time series for 22 different mean wind speeds (at 3 different turbulence intensities).

What you shall deliver

- An error-free main.py script that calculates and plots the mean and standard deviation of the blade and tower deflections versus wind speed for TI = 0.1.
    - The design/format of the plot is up to you.
    - The design of the main script is also up to you. You can add more functions to __init__.py if you need.
    - *Extra credit*: make your main.py such that it can process/visualize results from all 3 TIs.
- An updated README with a (1) quick-start guide and (2) explanation of how the code works.

Further constraints

- You may not push response timeseries to your repo.
- Your main.py script on the main branch shall run in 10 minutes or less on a standard student computer.
    - If your code is slow, consider designing main.py such that it has a “demo mode” and a “full mode”, where “demo mode” is designed to run faster[1]. Demo mode must still create the requested plot.
- Your team repo is locked on March 12, 23:59.
- The main branch will be used for evaluation/feedback.

---

[1] *Example: perhaps you could save some intermediate variable to file, then add/push just that variable but not the full time series. So in “demo mode”, the script re-uses the intermediate variable, but in “full mode” it re-simulates everything, including the intermediate variable.*

Peer-feedback rubric

Category	Item	Fail/missing	Very Poor	Poor	Okay	Very good	Excellent
Git use and workflow	Files on remote	Response files are pushed to the remote. Other in-progress or irrelevant files[2] may be included on remote.			A few unnecessary files on the remote.		No unnecessary files[2] on remote.
	Commit history	All commits are generic and made through the GitHub website (e.g., "Updating <filename>").			Some commit messages have been made from GitHub website, but most have been made from the terminal. Commit messages are generally clear.		Commit messages are specific, clear, and have been made using the terminal (e.g., no commit messages "Adding <filename>" or "Updating <filename>".
	Pull requests (PRs)	There are no open or closed PRs on the repo.			There are no open PRs and a few closed PRs.		There are several closed PRs with clear descriptions. Each PR has a different author and merger. There are no open PRs.
	Git workflow[3]	Commits are made directly on main by a single team member.			Some commits have been made directly to main.		Commits[4] are made exclusively in feature branches and then merged into main. The commits are evenly distributed amongst the team members.
Code and folder structure	Main script task	Script exits with an error, does not create requested plot OR has serious			Script analyses a single TI and generates a		Script analyses all 3 TIs and generates a plot that

[2] E.g., .DS_Store, desktop.ini, pycache folders, .pyc files, etc.
[3] Although the GitHub workflow is up to the team, a cleaner/more advanced workflow is through feature branches merged into main via pull requests.
[4] Commits by github-classroom[bot] may be ignored.

		mistake in the methodology.			quality plot that is correct.		is correct and of very good quality.
	Main script runtime[5]	Script does not complete after ~10 minutes.			Script completes after 5 minutes.		Script runs in less than 1 minute.
	main.py robustness	Script has many hard-coded or magic numbers related to the input data files.			Script has some magic numbers.		Script has no magic numbers and makes minimal assumptions about input data files.
	main.py "understandability"	Script has almost no comments, is poorly organized, and/or is extremely difficult to understand.			Script organization is okay, and there are some in-line comments, but room for improvement.		Script is logically organized and easy to understand by itself. Input parameters to script are clearly grouped together. There are both in-line comments and module/function docstrings where applicable.
	__init__.py "understandability"	Functions have little to no comments and are difficult to understand.			There are some in-line comments/ docstrings, but room for improvement in some functions.		Code in functions is easy to understand. There are both in-line comments and module/function docstrings where applicable.
	Folder structure	Folder structure is extremely poorly organized. There are missing files.			Folder structure is okay, but some room for improvement.		Folder structure is well-organized and logical. Files are easy to find.
Documentation	Collaboration.md	File is missing, incomplete or almost incomprehensible.			Collaboration methodology is generally explained but lacking detail.		Collaboration methodology is clear and well-explained.
	Quick-start guide (README.md)	No quick-start guide is given in the README.			Quick-start instructions are provided but		Process to quickly get started with the code is

[5] Of course this varies from computer to computer, so consider this a guideline. But you can be clever about how you write the script to make it run fast in a sort of "**demo mode**". (

					lacking some key details (e.g., working directory).		extremely clear, correct and easy to follow.
	Explanation of how the code works (README.md)	No explanation of how the code works is given in the README.			Explanation is given but not very specific/clear.		The explanation of how the code works is clear, creative, and includes at least one diagram. Assumptions of data-file formats are clearly explained.

Peer feedback in Week 6

Team session (50 minutes)

- Each team reviews 2 other teams.
- Every team member clones the repos of both of the other teams.
- Decide in your team how to split up the evaluation.
    - We suggest that you split by rubric item. I.e., Team Member A evaluates both of the other teams for Rubric Items #1, #2, #3, Team Member B evaluates Items #4, #5, #6, for both other teams, etc.
- Start running the main.py script early, in case it's slow. Remember to keep track of runtime.
- By time indicated in class: collate and submit a team-feedback form for each of the other teams. Save a copy of your feedback to present during the Round Robin.

Round robin (60 minutes)

- 3 teams enter each BOR.
- Team A gives a 2-minute demo of their code. Then, Teams B and C show their feedback. Discuss.