CodeCamp Programming Project

This is a description of the requirements for the CodeCamp project and the evaluation rubric that will be used for giving feedback.

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 your team shall deliver[1]

Before March 12 at 23:59, on main branch of your team repo:

- An error-free main.py script that calculates a plot (see next point) for TI = 0.1.
  - *Extra credit*: make your main.py such that it can process/visualize results from all 3 TIs.
- The plot shall visualize, in some way, the MEAN and STANDARD DEVIATION of the blade and tower deflections VERSUS MEAN WIND SPEED.
  - In other words, for each wind speed, you should simulate the blade/tower response, calculate $\mu_{xb,wsp\ i}$, $\sigma_{xb,wsp\ i}$, etc., and eventually plot them versus mean wind speed.
  - The design/format of the plot is up to you. Multiple figures/subplots are allowed.
- The design of the main script is up to you. You can add more functions to __init__.py if you want.
- An updated README with a (1) quick-start guide and (2) explanation of how the code works.
  - Target audience is a fellow student who has freshly cloned the repo and has the same terminal/Python skills as you but is not familiar with the CodeCamp project.

In-class on Thursday, March 13:

- Evaluation forms for 3 other teams.

[1] Any missing item(s) by the indicated deadline will result in the team receiving a “Fail” grade.

Further constraints you must fulfill[2]

- You may not push response timeseries to your repo.
    - You may push files with intermediate variables (e.g., statistics) if needed.
- 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[3]. 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.

How you shall deliver it

- The code shall be pushed to your team repo and merged into main before the indicated deadline.

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[4] may be included on remote.			A few unnecessary files on the remote.		No unnecessary files[4] 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		Almost all commit messages are specific, clear, and have been made using the terminal (e.g., no commit messages "Adding <filename>" or "Updating <filename>".

[2] If the code you hand in by the deadline does not fulfill these constraints, your team will receive a "Fail" grade.
[3] 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.
[4] E.g., .DS_Store, desktop.ini, pycache folders, .pyc files, etc.

					messages are generally clear.		
	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[5]	Commits are made directly on main by a single team member.			Some commits have been made directly to main.		Commits[6] 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 mistake in the methodology.			Script analyses a single TI and generates a quality plot that is correct.		Script analyses all 3 TIs and generates a plot that is correct and of very good quality.
	Main script runtime[7]	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

[5] Although the GitHub workflow is up to the team, the recommended workflow is through feature branches merged into main via pull requests.
[6] Commits by github-classroom[bot] may be ignored.
[7] 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", as discussed in an earlier footnote.

							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.
	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 lacking some key details (e.g., working directory).		Process to quickly get started with the code is 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 of very good quality. Assumptions of data-file formats are clearly explained.

Peer feedback in Week 6

In class Week 6, we will start with teams meeting in BORs and completing their peer evaluation of the other 3 teams listed below. Then, we will have a Round Robin where you present code to each other.

Team session (50 minutes)

- Each team reviews 3 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.

Students: who you should fill out feedback forms for

Each team fills out feedback forms for 3 other teams:

Your team	**Teams to give feedback to**		
Team Team	BugBusters	Git Happens	brunchy
Git Happens	Team Team	BugBusters	SIF
BugBusters	Git Happens	Team Team	WindyWizards
brunchy	WindyWizards	SIF	FatalError
SIF	brunchy	WindyWizards	A4 Highway
WindyWizards	SIF	brunchy	BugHunters
FatalError	BugHunters	A4 Highway	¿Qué? ¿Como Qué?
A4 Highway	FatalError	BugHunters	Los Programadores
BugHunters	A4 Highway	FatalError	Mouxtarides tou Mahalla
¿Qué? ¿Como Qué?	Mouxtarides tou Mahalla	Los Programadores	CryptoMania
Los Programadores	¿Qué? ¿Como Qué?	Mouxtarides tou Mahalla	Power-Fire
Mouxtarides tou Mahalla	Los Programadores	¿Qué? ¿Como Qué?	BladePYrunners
CryptoMania	BladePYrunners	Power-Fire	CodeGust

Power-Fire	CryptoMania	BladePYrunners	La Bombas
BladePYrunners	Power-Fire	CryptoMania	NetZero
CodeGust	NetZero	La Bombas	Team Team
La Bombas	CodeGust	NetZero	BugBusters
NetZero	La Bombas	CodeGust	Git Happens
BreezeTech	CodeFusion	CodeTeam	Lightning McTeam
Lightning McTeam	BreezeTech	CodeFusion	CodeTeam
CodeTeam	Lightning McTeam	BreezeTech	CodeFusion
CodeFusion	CodeTeam	Lightning McTeam	BreezeTech
PowerPuff Girls	Stop Fucking Spiders	Push & Pray	WindCoders
WindCoders	PowerPuff Girls	Stop Fucking Spiders	Push & Pray
Push & Pray	WindCoders	PowerPuff Girls	Stop Fucking Spiders
Stop Fucking Spiders	Push & Pray	WindCoders	PowerPuff Girls

Instructors/TAs: who you should fill out feedback for

- Reza (6): Team Team, BugBusters, Git Happens, brunchy, SIF, WindyWizards
- Nicolas (6): FatalError, BugHunters, A4 Highway, ¿Qué? ¿Como Qué?, Mouxtarides tou Mahalla, Los Programadores
- Ju Feng (7): CryptoMania, BladePYrunners, Power-Fire, BreezeTech, CodeFusion, CodeTeam, Lightning McTeam
- Jenni (7): CodeGust, NetZero, La Bombas, PowerPuff Girls, Stop Fucking Spiders, Push & Pray, WindCoders

Round robin (60 minutes)

- Teams enter the indicated BOR below.
- Team A gives a **3-minute** explanation of their code. Then, Teams B and C (and maybe D) present their feedback, item by item. Discuss.

Teams in BORs

```
WEEK 6
========================================
  BOR 0: Team Team, BugBusters, Git Happens
  BOR 1: brunchy, SIF, WindyWizards
  BOR 2: FatalError, BugHunters, A4 Highway
  BOR 3: ¿Qué? ¿Como Qué?, Mouxtarides tou Mahalla, Los Programadores
  BOR 4: CryptoMania, BladePYrunners, Power-Fire
  BOR 5: CodeGust, NetZero, La Bombas
  BOR 6: BreezeTech, CodeFusion, CodeTeam, Lightning McTeam
  BOR 7: PowerPuff Girls, Stop Fucking Spiders, Push & Pray, WindCoders
```