Option 1: Review “Data Analysis with Python” Project

Read through the article Data Analysis with Python – How I Analyzed My Empire State Building Run-Up

Performance by Jose Vicente Nunez, detailing his project to analyze the results of a stair-climbing run using Python. His project repo can be found on GitHub as well: EmpireStateBuildingRunUp.

Create a Word doc named EmpireRunUpSummary.doc and save it to your W9\_Exercises folder, and respond to the questions below:

Provide a summary of the project.

1. What data was being analyzed, and how was it obtained? 1.Why did the author take that approach? What were some of the challenges of the project?

Race data from websites using web scraping as data could be exported as a csv file. Analyzing data required purchases, so the author used open source to web scrape.

2. What tools from the article are you familiar with? What additional tools would you need to learn in order to undertake this project?

I am familiar with python but not web scraping, selenium web driver, and textual

Link: <https://www.youtube.com/watch?v=TdwwPP9oW4c>

3. Take a look at the project repository. What kind of summary information is provided to make the purpose of this repository clear? What sort of data is included in the folders?

There are three folders with the empire run up folder named: empirestaterunup, images,test

files named readme.md, summary.md, empirestate.svg, tutorial, pyproject.toml, requirements.txt(includes the required software/installation)

.md == markdown file

.svg == more for graphics (ex logo)

Link: https://www.bing.com/videos/riverview/relatedvideo?q=.svg+stand+for++git+hub+file&mid=6A6C03D44BFB12D9BBF76A6C03D44BFB12D9BBF7&FORM=VIRE

Toml== **Tom's Obvious, Minimal Language** (**TOML**, originally *Tom's Own Markup Language*[[2]](https://en.wikipedia.org/wiki/TOML#cite_note-2)) is a [file format](https://en.wikipedia.org/wiki/File_format) for [configuration files](https://en.wikipedia.org/wiki/Configuration_file).[[3]](https://en.wikipedia.org/wiki/TOML#cite_note-3) It is intended to be easy to read and write due to obvious semantics which aim to be "minimal", and it is designed to map unambiguously to a [dictionary](https://en.wikipedia.org/wiki/Dictionary_(data_structure)). Originally created by [Tom Preston-Werner](https://en.wikipedia.org/wiki/Tom_Preston-Werner), its specification is [open source](https://en.wikipedia.org/wiki/Open_source). TOML is used in a number of software projects[[4]](https://en.wikipedia.org/wiki/TOML#cite_note-rust-4)[[5]](https://en.wikipedia.org/wiki/TOML#cite_note-5)[[6]](https://en.wikipedia.org/wiki/TOML#cite_note-6) and is implemented in many programming languages.

Link: wikipedia

Remember to commit and push changes when you are done!

FREE CODE CAMP- STEPS TO ANALYZE A DATA USING THE ARTICLE EMPIRE STATE BUILDING RUN UP ANALYSIS WITH PYTHON

BREAK THE MAIN PROBLEM INTO SMALLER PIECES AND TACKLE EACH PIECE

Get the data by scraping the website (very few sites allow you to export race results as a CSV).

Clean up the data, normalize it, and make it ready for automatic processing.

Ask questions. Then translate those questions into code and tests, ideally using statistics to get reliable answers.

Present the results. A UI (Text or Graphic) will do wonders due to its low consumption, but charts speak volumes too.

step 1: git cloning or installing upgrade

step2: for web scraping Choose a good framework to automate what you can and prepare to do heavy data cleanup (also known as "give me patience for the stuff I can't control, like poorly done HTML and CSS"). selenium web driver

The data\_normalizer.raw\_read(raw\_file: Path) -> Iterable[Dict[str, Any]] method does the heavy work of fixing the data for inconsistencies before saving into a CSV format.

3. cleaning data

consistency is key

The esru\_csv\_cleaner script is the sum of the first stage cleanup effort, which takes the raw captured data and writes a CSV file with some important corrections:

4. **How to Analyze the Data**

Using pandas

Using z score for outlier

Describe

small, self-contained pieces of code to test your code. These are called unit tests.

*“The unittest .. supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.” (From the*[*Python docs*](https://docs.python.org/3/library/unittest.html)*)*

*5. Visualize data*

*Using textual*

*RunnerDetailScreen (extends ModalScreen*

*Plotting using mat plot lib*

[*Pilot class*](https://textual.textualize.io/api/pilot/)*that you can use to make automated calls to Textual Widgets and wait for events. This means you can simulate user interaction with the application to validate that it behaves as expected.*