After reading the instructions for the project I had some initial thoughts:

1. What type of service do we want to supply. These could be things such as an application with GUI, a CLI Application, a full stack web application or just an API that will serve the JSON.

I chose an API using either FASTAPI or Flask framework.

I did have a dillemma wether to insert the information into a database in our own API using a script that would run on a time interval of X. This would be a decision based on the number of users we have and the frequency of use for those users along with possible call limiting to the NFL APIs. With some more information we may be able to run a script as little as once and day and supply the information to our customers through a database. I believe for the simplicity of the project I will simply ETL(extract,transform,load) the data to the customer on GET requests.

2. How will we plan our development. What steps first - last.

I will start by designing a function which makes both get requests and transforms the data into the properly formatted JSON response.

After this I will work on the API.

A new focus here for me will be taking parameters in this form: (/api/endpoint?datefrom=<DATE>&dateto=<DATE>)

Which we will need to pass on to the scoreboard endpoint.

Once the API is set up and the function building the response is complete I will write error handling for possible bad dateto and datefrom parameters being passed.

I will than test the code by passing several dateto and datefrom ranges including misconfigured or bad dates.

Lastly I will review the code checking for effeciency, errors and that it is properly commented and readable.

3. Now that we have a plan we will start on the ETL function by writing some some pseudocode.

```
intake standings data and return json
intake scoreboard data and return json(pass datefrom and dateto)
** figure out default values for the date times if necessary
dateto = current date
datefrom=7days,14days,1month?
transform the data into a new json object using the retrieved json data
**iterate over the scoredboard eventids and fill additional information with standings data
return the new json object to be served by the API
```

4. Time to code!

While working on the functions for returning the new JSON object I ran into the problem of days with no events returning a list type instead of dict type and had to do some research to avoid having an error on those days. There is no data in thos days so I skip them if the type of days['day'] is a list type

After completing the function working with the events API and building the response object I filled the values needed from the rankings API with placeholder data. Than I began working on functions for the rankings API.

This API was much easier to parse for the required data. I built the functions to return a dict type of the team based on a team id passed. I wrote test cases to confirm it was returning the correct teams.

After this I began refactoring the functions into a class to be used in our API.

The class will be initialized with the following:

self.rankings_api_key from secrets.py

self.scoreboard_api_key from secrets.py

self.datefrom from API params

self.dateto from API params

Hit a roadblock refactoring. One of my functions inside the class is only recognizing 1 of two variables. Started investigating. Answer found I was not passing self,arg1,arg2 inside the class I was only passing arg1,arg2.

After finishing the class and checking the returned object I moved the API keys into a config file so we can use .gitignore

Now I will start to work on the fastapi application and integrate the new class for building the desired response.

Started by reading the fastapi documentation about passing parameters.

FASTAPI does the error handling for missing parameters. Based on the test cases we ran as long as we have valid parameters passed we will be ok.

We have to handle 2 error cases as determined by testing:

1. incorrect date formatting by validating the string

2. if the datefrom value passed is ahead of the dateto value.

    example. datefrom = 2021-10-12

    dateto = 2021-10-10

Read stackoverflow dealing with date validation to achieve the error handling.

Completed validation of both error cases and set the return JSON for both with information regarding why the error occured and how to fix it.

The API is now functioning however it will return empty if you query an amount > 7 days. This can be fixed with async await.

Reading documentation/stackoverflow on async in python and fast api

After implementing asyn I came to the realization that it did not fix the issue so I stepped back and went to my tests on the functions used in my class. When I did this and tried to pass more than 7 days I had similar results so I dug deeper and found out that the scoreboard API only allows you to pull a range totalling 7 days. This was extremely relieving.

With this new knowledge I have to write one more error case for the API parameters only allowing a mximum date range of 7 days.

Wrote the function for checking the difference in the dates. Tested the function against multiple test cases to confirm it is working. Implemented the error handling into the API.

Everything seems to be working as intended.

Now I will go through and clean up code and check for comments.

Lastly I will write instructions on how to test the API in the README.md file.

I hope you enjoyed following along in my thought proccess and planning. I tried to stop and write notes as much as I could. Sorry for any spelling errors as I was in a notepad file :)

Thank you again for the opportunity and I look forward to hearing ANY and ALL feedback that you may have so I can continue to grow as a developer.

-