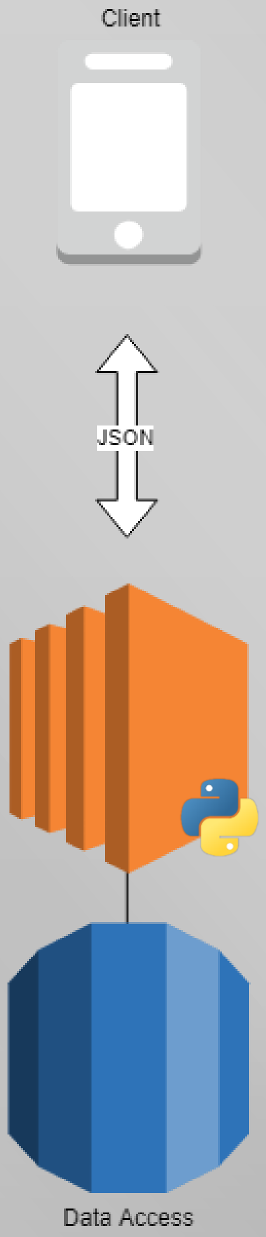
Contract Software Engineer Code Challenge Submission

Hannah Scovill

Submission GitHub Link

Assumptions for Test Environment

- Data flows through the application similarly to the diagram shown, sending & receiving JSON between the client & server. The data is ultimately stored in a relational database.
- Given this assumption, each solution starts with loading the JSON data into a python pandas dataframe. This acts as the result of a SQL query the API would make if the solution was implemented.
- The code used to combine the files containing the event data is in the administrative.py file. Only data collected for the "run" activity type was retained per the prompt's specifications. The resulting JSON file examined for this exercises is called mainJSON.json.



Given the provided data set, calculate the number of times the specified user ran more than 1km in a single run, 3 days in a row. Only count qualifying days once per consecutive streak.

Challenge 1

Challenge 1 Solution:

- The JSON data is loaded into a pandas dataframe. This emulates the result of a SQL query given a user_id as a parameter[1]
- Once the unique dates that the user ran more than 1km was extracted into a list, an algorithm loops through the array and records how long each streak greater than or equal to three is in a separate list.*
- Each item in the list is divided by three and truncated, and finally summed to return an integer representing the number of times the specified user ran more than 1km each day for three consecutive days.

*The prompt says “greater than.” Is the norm greater than or equal to so in the rare case someone runs exactly 1km they would still get credit?
[1]See slide: Future Implementation

Given the provided data set, calculate the number of times the specified user ran more than 10km in a calendar week. The week should start on Monday and end on Sunday.

Challenge 2

Challenge 2 Solution:

- The JSON data is loaded into a pandas dataframe. This emulates the result of a SQL query given a user_id as a parameter.[1] The dataframe also makes it easy to append columns for calculations that return the date greater than or equal to the start of the week the date falls in.
- The number of times the aggregate for the week is greater than or equal to 10k in the pivot_table grouped by the week starting date is returned as an integer.*

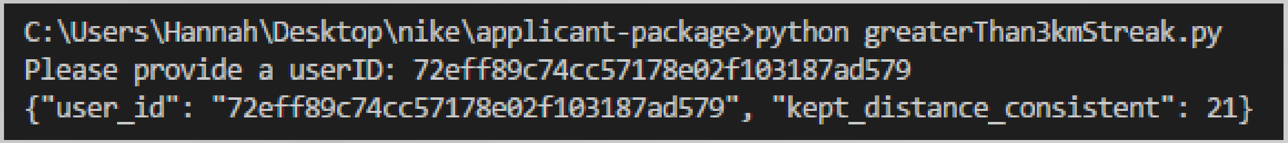
*The prompt says “greater than.” Is the norm greater than or equal to so in the rare case someone runs exactly 1km they would still get credit?
[1]See slide: Future Implementation

Given the provided data set, calculate the number of months the specified user made a personal record concerning pace for runs greater than or equal to 5k in the current year.

Challenge 3: Choose Your Own Adventure

Challenge 3 Solution

- The JSON data is loaded into a pandas dataframe. This emulates the result of a SQL query given a user_id and a minimum of 5k as parameters.
- Find the best pace from these entries before the current year started.[1] This allows the count to start in January for number of PR's, so any personal records the user makes in January is not discounted.
- Return a list of maximum pace for each month, sorted chronologically, taken from entries made in the current year.
- Loop through this array, count how many times the pace personal record was improved.



```
C:\Users\Hannah\Desktop\nike\applicant-package>python greaterThan3kmStreak.py
Please provide a userID: 72eff89c74cc57178e02f103187ad579
{"user_id": "72eff89c74cc57178e02f103187ad579", "kept_distance_consistent": 21}
```

Figure 1

Testing & Execution

- The data has been carefully examined in scratchpad jupyter notebooks in order to verify the answers. With these answers verified, unit tests can be added to the repository.
- To try each solution, from the command line type “python “ & solutionFileName.py. (Figure 1)

- The names of each file containing the solutions are:
  - Challenge 1: greaterThan3kmStreak.py
  - Challenge 2: moreThan10km.py
  - Challenge 3: pr5ks.py
- The unique userID’s* are:
  - 6bd5f3c04e6b5279aca633c2a245dd9c
  - 4e7aaa167b9b5ff7b9b3a22dee8c2085
  - c7e962db02da55209f02fe3d8a86c99d
  - d77908482ed2505ebbf17ef72be2f080
  - 72eff89c74cc57178e02f103187ad579
  - 40d7ae29e393582abdbcb8c726249e22

*Note: the userID’s are case sensitive

Future Implementation

1. Each API would be expanded to take time custom constraints as arguments. For this exercise, the time constraints are already added based on the beginning and end of the data provided in the folder.