AVATAR
TOP TRUMPS

Project Brief:

In this project you'll create a small game where players compare stats, similar to the Top Trumps card game. The basic flow of the games is:

1. You are given a random card with different stat
2. You select one of the card's stats
3. Another random card is selected for your opponent (the computer
4. The stats of the two cards are compared
5. The player with the stat higher than their opponent wins

Game Design:

I based the game design on my son's favourite TV series *Avatar: The Last Airbender* as I wanted to create a game that he could play and be involved in the design. The challenge with this however, was that there isn't any existing API for this series. So I set myself the challenge of setting up a simple API with card data that my son helped me create, in addition to making a 'Top Trumps' game as outlined in the project brief. I also wanted to make sure the game I created functioned just like the actual Top Trumps game which my son also likes to play a lot.

Project Requirements:

The project requirements I set myself were as follows:

Create card data for characters in the Avatar: The Last Airbender series: Top Trumps cards usually have 6 stats that you can compare between 15 cards each, so I would need to create 30 cards with stats in 6 categories

Create an API to store the card data to meet the requirements of the project: an existing API does not exist so I would need to create my own API

Create a function to call to the API and to retrieve the card data: this data would then be stored in a Python Dictionary as cards which can then be used in the game play

Create a function that 'shuffles' the cards: the 30 cards are shuffled like real Top Trumps cards then split between the users and the computer opponent dealing them 15 cards each

Ensure that my game follows the existing Top Trumps rules and game play: the user chooses stats which are then compared to the computer opponent's stats. If the player wins, they then gain that card from the computer which increases the total number of cards they hold. If there is a draw, the cards are put into a pile in the middle until someone wins, then they win the draw pile as well. If the computer wins, the computer gains the player's card and chooses a stat to play. Game play continues until either the computer as 0 cards left or the player has 0 cards left.

Create a function to check the player's existing score if they have one: and keeps a scorecard of data for all of the users. I also wanted to see if I could work out how to rank the players in the scorecard.

Other Requirements I noted at the brain storming stage:

- I wanted to look at formatting for the game and potentially using a front end such as HTML or Jinja2
- I wanted to see whether I could add in other gameplay dynamics such as giving a 'stat boost' when a 'Water' type card was played over a 'Fire' type card using a 'rock, paper, scissors' type logic
- I also thought about giving the user a % chance opportunity every time a stat is chosen to use a special move that would further boost their chance at winning against the computer and vice versa

The Final Game:

I created the file atla.py which meets all of my main priority project requirements identified in the brainstorming stage, as well as worked to include some of the feedback I received in the early stages of coding this project.

Back End Features:

- 30 character cards with 6 custom designed stats in a CSV based on the TV series *Avatar: The Last Airbender*

Avatar The Last Airbender Top Trumps

name	bending	power	defence	agility	wisdom	humour	honour	special	special_advantage
Avatar Aang	Air	10	10	10	10	6	10	Avatar State	5
Katara	Water	9	8	6	9	2	5	Water Tendrils	3
Sokka	None	5	2	4	7	10	7	Boomerang Throw	2
Toph	Earth	10	10	7	1	9	1	Metalbending	5
Zuko	Fire	9	3	9	8	2	9	Lightning Redirect	4
Iroh	Fire	10	9	5	10	8	5	Fire Breath	5

- API generated using retool.com/utilities to store the card information in the CSV as JSON, then uploaded to a custom API this can be accessed here:
    - https://api-generator.retool.com/dqbG1V/airbender

atla.py Features:

- Import Modules to help me create the code including: Pandas, Inflect, Requests, Time and Random
- 'get_cards' function requests full card data from API created using Retool
- User validation checks to see if they are an existing user
- Picking a computer user with a name from the Avatar TV series
- F-Strings and Rich Console to help with formatting
- Several functions created in the code to create the gameplay including:
    - 'shuffled' randomises the card order, then splits the cards between the user and computer opponent
    - 'opponent_pick' using random.choice to choose a stat from the dictionary keys
    - 'get_rank' which checks the score in the scorecard and works out a ranking position and if the player is 'tied' with someone else
    - 'play_game' which runs through the Top Trumps style game play
    - 'game_outcome' which runs whilst the user wants to play and if the length of cards is more than 0

PEP8 Adherence

I wanted to try to adhere to PEP8 rules where possible and so used the module 'pylint' to help me with this. The code in *atla.py* currently has a rating of 9.78/10 after making changes to some of the variable names, line lengths and making sure I had doc strings for the functions.

To run pylint, you write this statement on the command line in the terminal: pylint atla.py

Learnings & Conclusions:

I really enjoyed this project and put a lot of time into learning the code so I could understand what it was doing. I feel as though I have learned a lot of Python code in a really short space of time which has been really valuable. I did spend too much time trying to set up an API though. I tried using API gateway in AWS, I also tried using Github with Heroku and even installed Docker. I was not successful in my attempts to learn how to set up my own API, but I do think I learned a lot and would like to keep trying to learn this on my own.

In terms of the game itself, I do think I met the objective of creating a Top Trumps style game, however I do find the game would benefit from having some extra functionality. I had originally thought of adding random events to the game - chances to win more than one card for example, using 'special moves' for the characters in order to stop a card from being taken from you by the computer. If I do decide to carry on working on this game, I think I would like to add in these extra functions as well as creating a front end. I did start to look at potentially doing this using Jinja2 and Flask, but did not have time to implement it for the project. Going forward, I would like to learn how to do this though. Overall I thoroughly enjoyed this project and am looking forward to learning more Python in future.