Unit 3 | Assignment - Py Me Up, Charlie

Background

Well... you've made it!

It's time to put away the Excel sheet and join the big leagues. Welcome to the world of programming with Python. In this homework assignment, you'll be using the concepts you've learned to complete **2 of 4** Python Challenges. Which ones you choose are completely your choice. In fact, feel encouraged to complete them all if you can muster the time.

Each of these challenges encompasses a real-world situation where your newfound Python scripting skills can come in handy. These challenges are far from easy so expect some hard work ahead!

Before You Begin

1. Create a new GitHub repo called `python-challenge`. Then, clone it to your computer.
2. Inside your local git repository, create a directory for **2** of the **4** Python Challenges. Use folder names corresponding to the 2 challenges that you have chosen to complete: **PyBank**, **PyPoll**, **PyBoss**, or **PyParagraph**.
3. Inside of each folder that you just created, add a new file called `main.py`. This will be the main script to run for each analysis.
4. Push the above changes to GitHub.

Option 1: PyBank

In this challenge, you are tasked with creating a Python script for analyzing the financial records of your company. You will be given two sets of revenue data (`budget_data_1.csv` and `budget_data_2.csv`). Each dataset is composed of two columns: `Date` and `Revenue`. (Thankfully, your company has rather lax standards for accounting so the records are simple.)

Your task is to create a Python script that analyzes the records to calculate each of the following:

- The total number of months included in the dataset
- The total amount of revenue gained over the entire period
- The average change in revenue between months over the entire period
- The greatest increase in revenue (date and amount) over the entire period
- The greatest decrease in revenue (date and amount) over the entire period

As an example, your analysis should look similar to the one below:

```
Financial Analysis
----------------------------
Total Months: 25
Total Revenue: $1241412
Average Revenue Change: $216825
Greatest Increase in Revenue: Sep-16 ($815531)
Greatest Decrease in Revenue: Aug-12 ($-652794)
```

Your final script must be able to handle any such similarly structured dataset in the future (your boss is going to give you more of these -- so your script has to work for the ones to come). In addition, your final script should both print the analysis to the terminal and export a text file with the results.

Option 2: PyPoll

In this challenge, you are tasked with helping a small, rural town modernize its vote-counting process. (Up until now, Uncle Cleetus had been trustfully tallying them one-by-one, but unfortunately, his concentration isn't what it used to be.)

You will be given two sets of poll data (`election_data_1.csv` and `election_data_2.csv`). Each dataset is composed of three columns: `Voter ID` , `County` , and `Candidate` . Your task is to create a Python script that analyzes the votes and calculates each of the following:

- The total number of votes cast
- A complete list of candidates who received votes
- The percentage of votes each candidate won

- The total number of votes each candidate won
- The winner of the election based on popular vote.

As an example, your analysis should look similar to the one below:

```
Election Results
-------------------------
Total Votes: 620100
-------------------------
Rogers: 36.0% (223236)
Gomez: 54.0% (334854)
Brentwood: 4.0% (24804)
Higgins: 6.0% (37206)
-------------------------
Winner: Gomez
-------------------------
```

Your final script must be able to handle any such similarly-structured dataset in the future (i.e you have zero intentions of living in this hillbilly town -- so your script needs to work without massive re-writes). In addition, your final script should both print the analysis to the terminal and export a text file with the results.

Option 3: PyBoss

In this challenge, you get to be the *boss*. You oversee hundreds of employees across the country developing Tuna 2.0, a world-changing snack food based on canned tuna fish. Alas, being the boss isn't all fun, games, and self-adulation. The company recently decided to purchase a new HR system, and unfortunately for you, the new system requires employee records be stored completely differently.

Your task is to help bridge the gap by creating a Python script able to convert your employee records to the required format. Your script will need to do the following:

- Import the `employee_data1.csv` and `employee_data2.csv` files, which currently holds employee records like the below:

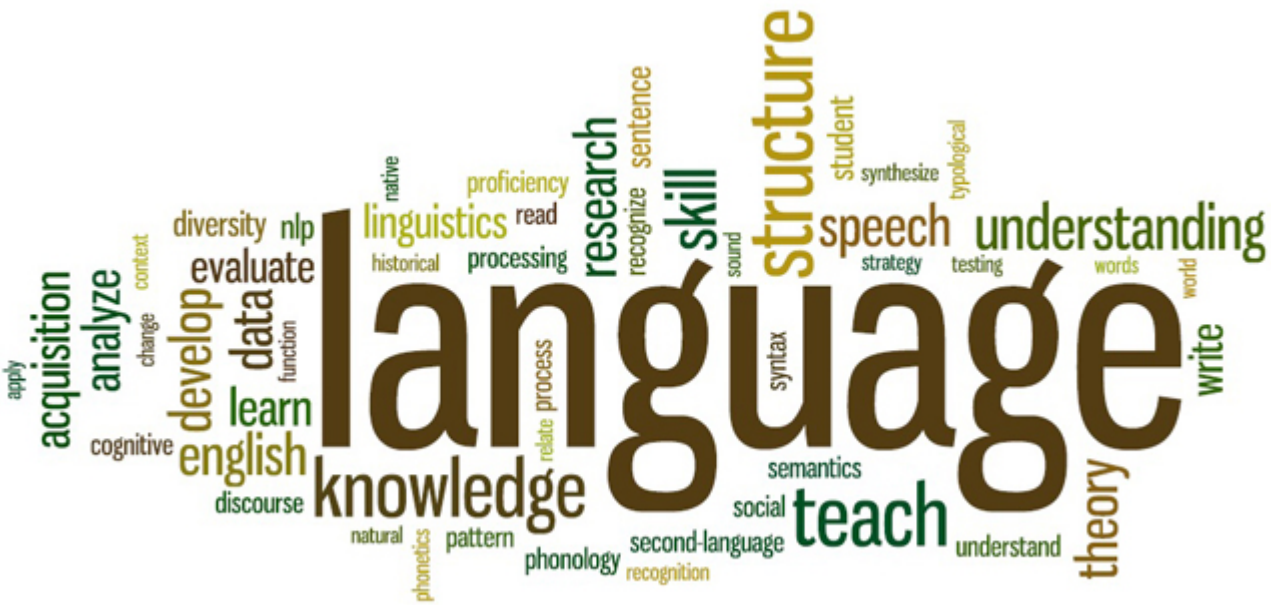
```
Emp ID,Name,DOB,SSN,State
214,Sarah Simpson,1985-12-04,282-01-8166,Florida
15,Samantha Lara,1993-09-08,848-80-7526,Colorado
411,Stacy Charles,1957-12-20,658-75-8526,Pennsylvania
```

- Then convert and export the data to use the following format instead:

```
Emp ID,First Name,Last Name,DOB,SSN,State
214,Sarah,Simpson,12/04/1985,***-**-8166,FL
15,Samantha,Lara,09/08/1993,***-**-7526,CO
411,Stacy,Charles,12/20/1957,***-**-8526,PA
```

- In summary, the required conversions are as follows:
  - The `Name` column should be split into separate `First Name` and `Last Name` columns.
  - The `DOB` data should be re-written into `DD/MM/YYYY` format.
  - The `SSN` data should be re-written such that the first five numbers are hidden from view.
  - The `State` data should be re-written as simple two-letter abbreviations.
- Special Hint: You may find this link to be helpful—Python Dictionary for State Abbreviations.

Option 4: PyParagraph



In this challenge, you get to play the role of chief linguist at a local learning academy. As chief linguist, you are responsible for assessing the complexity of various passages of writing, ranging from the sophomoric Twilight novel to the nauseatingly high-minded research article. Having read so many passages, you've since come up with a fairly simple set of metrics for assessing complexity.

Your task is to create a Python script to automate the analysis of any such passage using these metrics. Your script will need to do the following:

- Import a text file filled with a paragraph of your choosing.

- Assess the passage for each of the following:
  - Approximate word count
  - Approximate sentence count
  - Approximate letter count (per word)
  - Average sentence length (in words)
- As an example, this passage:

> "Adam Wayne, the conqueror, with his face flung back and his mane like a lion's, stood with his great sword point upwards, the red raiment of his office flapping around him like the red wings of an archangel. And the King saw, he knew not how, something new and overwhelming. The great green trees and the great red robes swung together in the wind. The preposterous masquerade, born of his own mockery, towered over him and embraced the world. This was the normal, this was sanity, this was nature, and he himself, with his rationality, and his detachment and his black frock-coat, he was the exception and the accident - a blot of black upon a world of crimson and gold."

...would yield these results:

```
Paragraph Analysis
-----------------
Approximate Word Count: 122
Approximate Sentence Count: 5
Average Letter Count: 4.56557377049
Average Sentence Length: 24.4
```

- **Special Hint:** You may find this code snippet helpful when determining sentence length (look into regular expressions if interested in learning more):

```
import re
re.split("(?&lt;=[.!?]) +", paragraph)
```

Hints and Considerations

- Consider what we've learned so far. To date, we've learned how to import modules like `csv` ; to read and write files in various formats; to store contents in variables, lists, and dictionaries; to iterate through basic data structures; and to debug along the way. Using what we've learned, try to break down you tasks into discrete mini-objectives. This will be a *much* better course of action than attempting to Google Search for a miracle.
- As you will discover, for some of these challenges, the datasets are quite large. This was done purposefully, as it showcases one of the limits of Excel-based analysis. While our first instinct, as data analysts, is often to head straight into Excel, creating scripts in Python can provide us with more robust options for handling "big data".
- Your scripts should work for each dataset provided. Run your script for each dataset separately to make sure that the code works for different data.
- Feel encouraged to work in groups, but don't shortchange yourself by copying someone else's work. You get what you put in, and the art of programming is extremely unforgiving to smoochers. Dig your heels in, burn the nigh oil,

and learn this while you can! These are skills that will pay dividends in your future career.

- Start early, and reach out for help often! Challenge yourself to identify ***specific***questions for your instructors and TAs. Don't resign yourself to simply saying, "I'm totally lost." Come prepared to show your effort and thought patterns, we'll be happy to help along the way.
- Always commit your work and back it up with GitHub pushes. You don't want to lose hours of your work because you didn't push it to GitHub every half hour or so.
    - **Commit often**.
- Don't spend too much time trying to choose the "right" challenge. Each of these is fairly comparable in difficulty level, and you'll need to use most of the same techniques in each. Waffling won't get you anywhere. Pick your battle, and stick with it! Good luck!

Copyright