COSC 311 - Lab 1

Dr. Joe Anderson

Due: 10 September 2020

1 Objectives

1. Gain familiarity with the Python programming language
2. Gain familiarity with Jupyter notebooks and the Jupyter hub cloud system
3. Practice efficiently manipulating data with Python

2 Tasks

1. Log in to the Jupyter hub environment at `jupyterhub.salisbury.edu`. When prompted, select the "Data Science" environment in order to get access to many common data processing and numerical computing libraries. Note that you can change your environment by stopping your personal server and re-starting it; you will prompted to select an environment again.
2. Create a new notebook to use for the remainder of these exercises.
3. Hard-code a *list of 10 dictionaries*, where each dictionary contains the keys `id` and `name`; each of these entries will represent a musician. For example,

```
cities = [
  { "id": 0, "name": "Susan Tedeschi" },
  { "id": 1, "name": "Bob Dylan" }
  # Add at least eight more here!
]
```

4. Similarly, create a list of dictionaries for song titles:

```
songs = [
  {"id": 0, "name": "Angel from Montgomery"},
  {"id": 1, "name": "Don't Think Twice, It's Alright"
  {"id": 2, "name": "It Hurt So Bad"},
  {"id": 3, "name": "Simple Twist of Fate"},
  # Add more! Make sure every artist has at least one, and that
  # some artists have songs in common.
]
```

Make sure you include some songs that are played/covered by multiple artists!

5. Create a *list of tuples* to denote whether two artists perform the same song (regardless of who wrote the original version). Example:

```
artist_songs = [
  (0, 0), (0, 1), (0, 2),
  (1, 1), (1, 3)
]
```

Note that this way of "linking" artists to is pretty space efficient and logical in the long-term. If you need to edit the artist or song name – or other metadata that we are omitting here – then the id relationships are still in-tact; if you have taken a course in databases, this is usually how many-to-many relationships are captured.

6. Write a subroutine `who_covered` that takes the name of a song as an argument and returns a list of names of artists who have covered that song.

7. Write a subroutine `shared_songs` that takes an artist name and returns a list of songs they share with other artists.

8. Write a subroutine `song_popularity` that prints out – in order from most to least covers – the name and number of times each song has been covered.

9. Next, create a *list of tuples* where each tuple contains a pair of data: the `id` of a musician and a "keyword" that describes their music. For example:

   ```
   keywords = [
     (0, "blues"), (0, "female lead"), (0, "guitar"),
     (1, "folk"), (1,, "guitar"), (1, "male lead")
   ]
   ```

   Be sure to add several for each artist and use some creativity! Importantly, be sure that repeated keywords (e.g. `guitar`) are spelled the same, with the same capitalization.

10. Write a function `similar_artists` that takes an artists name and returns the top three other artists that have the most keywords in common with the one passed in.

11. Now on to some visualization! You're currently set up with a "database" of musical artists, songs, and attributes of those artists. Consider how to use your data to answer the question:

    How can we visualize the "type" of each song? What styles and what types of artists tend to play each song?

    For a specific example, what visualization can answer the question:

    What style of song is "Simple Twist of Fate"? Does it have a style or does it "defy genre"?

    Write down some (any that you can think of) visualization strategy to answer this question. Can you use line/bar charts to illustrate? Can you use positional data (points), and what might the axes look like? Are there other "non-standard" charting methods you can think of?

    (a) Sketch by hand or digitally an example of your idea, using the data you entered above

    (b) Investigate how you might get Python to automate the production of your idea. What libraries can you find that might be useful?

    (c) Finally, write down one strategy to try answering the above questions about song-style *quantitatively*. Why? If you're, say, Spotify, then you would probably like a way to appease a user who says: "I like classic rock that uses lots of keyboard and slow rythm"!

3 Submission

Zip your source files and upload them to the assignment page on MyClasses. Be sure to include all source files, properly documented, a `README` file to describe the program and how it works, along with answers to any above discussion questions.