CSE 101: Introduction to Computational and Algorithmic Thinking

Homework #3-1

Fall 2018

Assignment Due: Monday, December 3, 2018, by 11:59 pm

Getting Started

This assignment contains a "bare bones" file with a name like `playlist.py`. This file contains one or more function and/or class *stubs* and a few tests you can try out to see if your code seems to be correct. You must fill in the missing code as directed in order to complete the assignment. **Do not, under any circumstances, change the names of any classes or functions (including those functions' parameter lists) that we provide in the starter code.** The automated grading system will be looking for exactly those class and function names (you are more than free to add your own helper functions and classes if you desire). **Please note that the test cases and data files that we give you are not necessarily the same ones we will use during grading!**

Directions

This assignment is worth a total of 20 points. The automated grading system will execute each function in your solution several times, using different input values each time. Each test that produces the correct/expected output will earn 1 or more points, adding up to the total point value available for that function. You must earn at least 16 points (80%) in order to pass (receive credit for) this assignment.

This assignment counts towards Course Outcome 3: Using Algorithms to Develop and Express Solutions to Computational Problems.

- At the top of the `playlist.py` file, include the following information **in full-line comments**, with each item on a separate line:
  - your full name ***AS IT APPEARS IN BLACKBOARD***
  - your Stony Brook ID #
  - your Stony Brook NetID (your Blackboard username)
  - the course number (CSE 101) and semester (Fall 2018)
  - the assignment name and number (Assignment #3-1)

⚠ Any functions that we tell you to complete (i.e., that we provide stubs for in the starter code) **MUST** use the names and parameter lists indicated in the starter code file. Submissions with different function names (or whose functions contain different parameter lists) can't be graded by our automated grading system.

⚠ Be sure to submit your final work as directed by the indicated due date and time. Late work will not be accepted for grading. Work is late if it is submitted after the due date and time.

⚠ Programs that crash will likely receive a grade of zero, so make sure you thoroughly test your work before submitting it.

Creating Custom Playlists

Suppose that you have a side business working as a DJ for parties and other events. Each event runs for a different length of time and has a different clientele, so you need a way to quickly generate a customized playlist of songs from your music library. In this assignment, you will write a program that will handle this task for you (or let you know that the task is impossible using the current music library and/or with the specified set of constraints).

1. **Constructing a Library (8 points)**

   First, we need to assemble our library of songs. Each song has several attributes associated with it:

   - the song title
   - the artist who performed the song
   - the song's genre (e.g., "rock")
   - the song's length (measured in seconds)

   We will collect this information from a (plain text) data file and store it in a dictionary. Each key in the dictionary corresponds to a unique genre; its value will be a list of songs (from the data file) that belong to that genre. Each song is represented by a list (or tuple) with exactly three elements, in the following order: title (a string), artist (a string), and length in seconds (an integer). For example:

   ```
   "rock" : [ ("We're Not Gonna Take It", "Twisted Sister", 217),
              ("Born To be Wild", "Steppenwolf", 211),
              ("Pinball Wizard", "The Who", 182)
            ]
   ```

   denotes that our library contains three songs that belong to the "rock" genre.

   Start by completing the `buildLibrary()` function. This function takes a single string argument, representing the name of a data file. The function returns a dictionary in the format above that has been created using the contents of the data file.

   Each line of the data file contains information about a single song, in the following order (fields are separated by commas; assume that there are no leading/trailing spaces between fields):

   song genre, song title, artist, song length (in MM:SS format)

   For example, "jazz,Sputnik,Conrad Boqvist,5:05" describes a jazz song ("Sputnik", by Conrad Boqvist, whose length is 305 seconds (5 minutes, 5 seconds). **Hint:** To convert the time string into seconds, split it based on the colon (":") character. Convert each piece to an integer, and perform the appropriate arithmetic.

   Your general solution strategy should be as follows:

   ```
   Set library to an empty dictionary

   For each line in the data file:
       Split the line based on commas
       Convert the "time" element into an integer number of seconds
       Create a new list or tuple with the song title, artist, and length
       Add the new list to your dictionary, using the genre as the key

   Return the dictionary
   ```

2. **Building a Basic Playlist (12 points)**

Complete the `buildPlayList()` function, which takes three arguments, in the following order:

(a) a dictionary of songs grouped by musical genre (like the one produced by the `buildLibrary()` function from the previous step)

(b) a list of strings representing one or more musical genres

(c) an integer *n* representing the total number of songs to select for the resulting playlist

This function returns a list of exactly *n* 2-element tuples (each tuple consists of a song title, followed by the artist name), randomly drawn from the genres specified above ***(and ONLY those genres)***. For example, if the second argument was the three-element list ["hip-hop", "classical", "metal"], the new list should only contain songs from those three genres (i.e., the result should not contain any songs that belong to a different genre like "country" or "reggae").

- If one or more of the requested genres do not exist in the library, your function should ignore that genre (for example, if the library had no "classical" songs in the example above, then the function should only consider songs from the "hip-hop" and "metal" genres).
- If the library does not contain **ANY** of the requested genres, then the function should return an empty list.
- If *n* is greater than the total number of songs that are available in all of the requested (and available) genres combined, the function should return a list with all of the songs in those genres combined, in any order. For example, if the user requests a 20-song playlist drawn from four genres that have 2, 5, 4, and 6 songs respectively, the function should simply return a list with all 17 available songs.

Start by constructing a list of potential songs from each of the specified genres. Then use a function like `random.sample()` (from the `random` module — see the slides for an example of how to use this function) to construct your playlist.

Examples:

Here are examples of what each of your functions should produce as output using the sample data files that were provided with this assignment (the formatting may vary somewhat). Bear in mind that the `buildPlayList()` function selects songs randomly from its library, so your answers may vary slightly from the ones below. Remember that we will use different data files when we actually test and grade your code.

```
Testing buildLibrary() with file 'library1.txt':
Library contents are:
{'childrens': [('Angela Lansbury', 'Be Our Guest', 224),
               ('Lullabye', 'Billy Joel', 213)
              ],
 'dance': [('Happy Now', 'Kygo', 211),
           ('Grapevine', 'Tiesto', 150),
           ('Headspace', 'Dee Montero', 271)
          ],
 'blues': [('Dream of Nothing', 'Bob Margolin', 208),
           ('Rock and Stick', ' Boz Scaggs', 290),
           ('At Last', 'Etta James', 181),
           ("You're Driving Me Crazy", 'Van Morrison', 286)
          ],
 'kpop': [('Not That Type', 'gugudan', 191),
          ('IDOL', 'BTS', 222),
          ('Believe Me', 'Seo In Young', 191),
          ('Baam', 'MOMOLAND', 208),
          ('Hide Out', 'Sultan of the Disco', 257)
         ]
}

Testing buildLibrary() with file 'library2.txt':
Library contents are:
{'jazz': [('Frankenstein', 'Christian Sands', 523),
          ('Lebroba', 'Andrew Cyrille', 344)
         ],
 'classical': [('Perfect', 'The Piano Guys', 310),
               ('To Where You Are', 'Katherine Jenkins', 226)
              ],
 'country': [('Chrome', 'Trace Adkins', 203),
             ('Nothing I Can Do About It Now', 'Willie Nelson', 198),
             ("Should've Been a Cowboy", 'Toby Keith', 302),
             ('Better Than You', 'Terri Clark', 242)
            ]
}
```

```
Testing buildLibrary() with file 'library3.txt':
Library contents are:
{'metal': [('Call The Man', 'Pentagram', 229),
           ('Brimstone', 'Whitechapel', 205),
           ('Lightning Strike', 'Judas Priest', 209),
           ('Enter Sandman', 'Metallica', 331)
          ],
 'reggae': [('Sidung', 'Kranium', 200),
            ('Naked Truth', 'Sean Paul', 215),
            ('Red Red Wine', 'UB40', 320)
           ],
 'hiphop': [('Spaceship Odyssey 2000', 'Malik Yusef', 276),
            ("It's Tricky", 'Run-DMC', 183),
            ('Cuttin Rhythms', 'Tone Loc', 311)
           ]
}

Testing buildLibrary() with file 'library4.txt':
Library contents are:
{'electronic': [('Fast Life', 'Jackson and His Computer Band', 308),
                ('Stairway to Heaven', 'Jana', 245),
                ('Silikon', 'Modeselektor', 226)
               ],
 'rock': [("School's Out", 'Alice Cooper', 210),
          ('My Reply', 'The Ataris', 254),
          ('Burning Bridges', 'Bon Jovi', 164),
          ("Escher's Staircase", 'Lana Lane', 366)
         ],
 'alternative': [('Psycho Killer', 'Talking Heads', 260),
                 ('So Far So Good', 'Thornley', 202),
                 ('Nothing At All', 'Wired All Wrong', 198),
                 ('Chocolate', 'The 1975', 227)
                ]
}


Testing buildPlayList() on the contents of library1.txt:
6 songs drawn from the blues, childrens, and dance genres:

[('Grapevine', 'Tiesto'),
 ('Dream of Nothing', 'Bob Margolin'),
 ('Angela Lansbury', 'Be Our Guest'),
 ('Rock and Stick', ' Boz Scaggs'),
 ("You're Driving Me Crazy", 'Van Morrison'),
 ('At Last', 'Etta James')
]
```

```
Testing buildPlayList() on the contents of library1.txt:
12 songs drawn from the blues and kpop genres:

[('Dream of Nothing', 'Bob Margolin'),
 ('Rock and Stick', ' Boz Scaggs'),
 ('At Last', 'Etta James'),
 ("You're Driving Me Crazy", 'Van Morrison'),
 ('Not That Type', 'gugudan'),
 ('IDOL', 'BTS'),
 ('Believe Me', 'Seo In Young'),
 ('Baam', 'MOMOLAND'),
 ('Hide Out', 'Sultan of the Disco')
]

Testing buildPlayList() on the contents of library2.txt:
4 songs drawn from the classical, country, and opera genres:

[('Better Than You', 'Terri Clark'),
 ('To Where You Are', 'Katherine Jenkins'),
 ('Perfect', 'The Piano Guys'),
 ('Nothing I Can Do About It Now', 'Willie Nelson')
]

Testing buildPlayList() on the contents of library3.txt:
8 songs drawn from the classical, childrens, and dance genres:
[ ]
```