CMSC 201 Fall 2017

Project 1 – Music Library

Assignment: Project 1 – Music Library
Due Date:
Design Document: Friday, October 20th, 2017 by 8:59:59 PM
Project: Friday, October 27th, 2017 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 1, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Project 1 is the first assignment where you've had to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberately constructing your program and how it will work, rather than coding as you go along, or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on using functions to break a large task down into smaller parts.

You are ***required*** *to follow the provided design exactly*! You may add additional functions, but you must implement all of the specified functions as described in the design overview.

At the end, your Project 1 file must run without any errors.
It must also be called proj1.py (case sensitive).

Additional Instructions – Creating the proj1 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 1 files. We recommend calling it **`proj1`**, and creating it inside a newly-created directory called **`Projects`** inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

Project 1 is designed to give you lots (and lots) of practice with functions, as well as help you become more familiar and comfortable with 2D lists. You'll need to use **`while`** loops, control statements like **`if/else`**, passing in parameters, returning from functions, concatenation needed for using **`input()`**, and algorithmic thinking.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Task

You'll be using a small selection of data from the Music Library dataset from the CORGIS website (https://think.cs.vt.edu/corgis/csv/music/music.html). We have reduced the amount of information drastically, both in terms of the number of entries and the details for each entry.

The data you'll be dealing with contains information about each song's year, artist, song title, genre, duration (how long it is in seconds), and tempo (how fast it is in beats per minute). We've also written the function that will read in and organize the data for you, storing it in a 2D list of song data, where each interior list is a single song's data.

Your program will allow a user to "load in" a database of their choosing, and to either (1) search the database, using any of the details listed above; or (2) create a playlist of length 10 or less, based on year, artist, or genre. The databases can be downloaded from Dr. Gibson's directory, along with the function that reads them in.

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/proj1.py .
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/songs.csv .
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/short.csv .
```

See the sample output available on Blackboard for detailed examples of how everything works, when run on the **`short.csv`** database.

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

For now, you should pay special attention to the sections about:

- Comments
  - ***Function header comments***
  - Please note that the "Input" and "Output" in the function header comment do NOT mean what is shown on the screen with `print()`, or what is gotten from the user with `input()`. They refer to the **parameters** taken in, and the **return value**. (Both "Input" and/or "Output" may be None if appropriate.)
- Constants
  - You must use constants instead of magic numbers or strings!!!
  - We'll allow "magic strings" in the `colToString()` function
- Make sure to read the last page of the Coding Standards document, which prohibits the use of certain tools and Python keywords

Additional Specifications

For this assignment, **you must follow the design overview** in this document.

For this assignment, you do need to worry about "input validation." You may assume that the user will enter an integer, but it may be negative or outside of the allowable range.
If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

Project

The project is worth a total of 80 points. Of those points 10 will be based on your design document, 10 will be based on following the coding standards, and the other 60 will be based on the functionality and completeness of your project.

Design Document

The design document will make sure that you begin thinking about your project in a serious way early. This will not only give you important experience doing design work, but will help you gauge the number of hours you'll need to set aside to be able to complete the project. **Your design document must be called design1.txt.**

For Project 1, the design overview is included in this document, and you are **required to follow it**. For future projects, you will be creating the design entirely on your own, and may choose to design it however you like.

Your design document must have four separate parts:

1. A file header, similar to those for your assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for
3. Function headers
    a. A complete function header comment for each function
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under the relevant comment (if applicable)

Although you will be presented with a design overview, you must still create the function headers, and the pseudocode for **`main()`** on your own.

A start for your design is provided on Blackboard under "Assignments". Follow the layout and format of that document. You can also copy it using:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/design1.txt .
```

NOTE: The sample design provided is not complete, and is missing many constants and function header comments. You need to add them!

Your **`design1.txt`** file will be compared to the **`proj1.py`** file that you submit. Minor changes to the design are allowed. A minor change might be the addition of another function, or a small change to **`main()`**.

Major changes between the design and your project will lose you points. This would indicate that you didn't give sufficient thought to your design.
(If your submitted design doesn't work, it is generally better to lose the points on the design, and to have a functional program, rather than turning in a broken program that follows the design. The ultimate decision is up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample1.txt". Look at the sample output before reading the notes below.
(Yours does not have to match the sample output exactly, but it should be similar, and the columns should all line up nicely.)

Other Hints

- Writing up and testing each of your functions individually will make your life much, much easier.
- Commenting your code as you write it or even before you write it (think pseudocode) will make the process much simpler, and will also allow the TAs to help you more effectively.

- When comparing data, remember that the year is stored as an integer, and the tempo and duration are floats. The number 10 is not equivalent to the string "10" after all.

Design Information

You are required to implement and use at least the following ten functions for Project 1, in addition to **`main()`**. You may implement more functions if you think them necessary, but the ten below must be implemented **and used**. The information for each function is below.

Printing Functions

As the name suggests, these functions print information to the user. **None** of them return values – they only print information, which does NOT count as output. Some of them take in parameters, while others do not.

- **`def displayMainMenu()`**
  - Prints out the main menu of the program

```
What would you like to do next?
0) Perform a search
1) Create a playlist
```

  - Takes in nothing (the menu is always the same)
  - Returns nothing, since it is a print function

- **`def displayOptions()`**
  - Prints out a list of the six different attributes shown for each song

```
0 – Year
1 – Artist
2 – Title
3 – Genre
4 – Duration
5 – Tempo
```

  - Takes in nothing (the menu is always the same)
  - Returns nothing, since it is a print function

- **`def displayPLOptions()`**
  - Prints out the three different options for creating a playlist

```
1 – Year
2 – Artist
3 – Genre
```

  - Takes in nothing (the menu is always the same)
  - Returns nothing, since it is a print function

- **`def printSongs(songs)`**
  - Prints the details of every song in the given 2D list
    `Year – Genre – Artist – Song Title`
  - Takes in a 2D list of songs
    - Each song is a separate interior list (*i.e.*, a row)
  - Returns nothing, since it is a print function

Helper Functions

As the name suggests, these functions "help" other functions, by performing small tasks that will be needed by many pieces of the program. They are often called from functions other than **`main()`**.

- **`def getValidInput(prompt, minimum, maximum)`**
  - Gets a valid integer from the user that falls within a certain range (between **`minimum`** and **`maximum`**, inclusive), using the provided string of **`prompt`** when asking the user for input.
  - Takes in a prompt and two integers
    - The prompt is used by the function when asking for input
    - The two integers are the min and max values (inclusive)
  - Returns an integer
  - ❖ **HINT**: Look at Lecture 13 for an example of a similar function, **`getValidInt()`**, which does not contain a prompt parameter

- **`def colToString(column)`**
  - Converts a number to the corresponding column heading
    `0` becomes **`Year`**
    `1` becomes **`Artist`**
    `2` becomes **`Title`**
    `3` becomes **`Genre`**
    `4` becomes **`Duration`**
    `5` becomes **`Tempo`**
  - Takes in an integer, the index of a column
  - Returns a string containing the column heading

- **`def songToString(song)`**
  - Converts a song's information into a string of its details
    `Year – Genre – Artist – Song Title`
  - Takes in a 1D list containing the details of a single song
  - Return value is the final string with the details
    - For example:
      `"1994 - rock - Tom Petty - A Higher Place"`

General Functions

These are the "heavy lifters" of the program, and do the majority of the work, and are almost always called from **`main()`**. They often call other functions, often more than one.

- **`def search(songs, col, value)`**
  - Create a 2D list of all the songs that match the value being searched for in the selected column
  - Takes in the list of songs, a column, and a value to search for
    - **`col`** is an integer between 0 and 5 (representing year, artist, title, genre, duration, or tempo, respectively)
    - **`value`** is the value being searched for
  - Returns a 2D list of all the songs that match the value in the selected column
    - If the column selected is tempo or duration, **`search()`** should return all songs where that column is **greater than or equal to** the value being searched for
    - For all other columns, the match must be exact (including case sensitivity)

- **`def createPlaylist(songs, choice, length)`**
  - Creates a 2D list of all songs within the type selected by the user; limits the list to a length of **`length`**
  - Takes in the list of songs, a choice, and a length for the playlist
    - **`choice`** is an integer 1 and 3 (representing the playlist options of year, artist, or genre, respectively)
    - **`length`** is a positive integer between 0 and 10
      - The maximum length of a playlist is 10 songs.
  - Within the function, the user should be prompted to enter either the year, artist, or genre a playlist should be comprised of
  - Returns a 2D list of the first **`length`** songs that match the value for the selected choice
    - For example, if the user selected year and 3 songs, then chose 1999 as the year, the first three songs from 1999 should be returned
  - ❖ **HINT**: Could you make use of **`search()`** to do most of the work here? How would you control the length of the returned list?

- **`def make2DList(filename)`**
  - Takes in the filename as a string
  - Reads in the contents of the song library file
  - Stores the results in a 2D list called **`resultList`**
  - ❖ **This function is given to you!!!**
    - You can find it in Dr. Gibson's public directory
    - Make sure to include the provided function header.
      - You can delete the table showing the data organization (after the "Output" line) if you like.
    - Do not change the code – it works perfectly as is!

Helpful Checklist

We understand that the first big project can be overwhelming, which is why we've provided you with a detailed description of all the functions, along with sample output, a starter design, and more. Below, you'll find a summary of the key actions and where important documents can be found.

Design:

- ☐ Download the starter design from Dr. Gibson's public directory:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/design1.txt .
```

- ☐ Read through the function descriptions provided in this document, and create function header comments for each one
- ☐ Look at the sample output on Blackboard, and begin sketching out an idea for how your **`main()`** will work
  - ☐ Look at the function headers you wrote to gain more insight

Coding:

- ☐ Download the given **`make2DList`** function, along with a starter set of constants for project 1, from Dr. Gibson's public directory:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/proj1.py .
```

- ☐ Download the two given song libraries for testing with:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/songs.csv .
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/short.csv .
```

  - ☐ If you mess these up, you can always re-download them!
- ☐ Going off of your design, start coding up your project piece by piece
  - ☐ Use top-down or bottom-up implementation – your choice!
  - ☐ Don't code up everything at once, and test as you go
- ☐ If you make changes to your design while coding, and it's before the design due date, update your design and resubmit it!
- ☐ Come to office hours when you need help or get stuck

General:

- ☐ Submit the design to PROJ1_DESIGN, **not** PROJ1
- ☐ Start working on the project itself before October 20th if possible, since actually coding it up will improve your design as you go

Submitting

Once your **`proj1.py`** or **`design1.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the project in multiple times, as you reach new milestones. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, October 20th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

To submit your project file (which is due Friday, October 27th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1 proj1.py
Submitting proj1.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**