16.1 Portfolio Project 3

Portfolio projects are designed to showcase your problem-solving and coding abilities to future employers, mentors, supervisors, etc. You will translate the given specifications into a coded solution.

Due: Dec. 1, 2023 11:59 pm

Late submissions: -10% if submitted by December 2nd 11:59 pm, -20% if submitted by December 3rd 11:59 pm

No submissions will be accepted after 11:59 pm on December 3rd. No extensions or replacement assessments will be given.

Learning Outcomes:

- Accept and validate user input through varied menu options
- Create and manipulate arrays using the numpy module
- Process data according to specifications
- Manipulate and execute numpy array computations
- Plot data using the matplotlib module
- Create an object instance using a given class (bonus)

Specifications:

Python is well-known as a language that excels at data manipulation and visualization. There are many powerful tools built for Python which allow up to process large datasets to find, isolate, and present patterns within the data. We call the process of taking raw data and transforming it into a format we can work with "data wrangling". This is often required when working with graphing libraries, as they may expect data to be provided in a specific format. For your third portfolio project, you are being asked to design a terminal-based application for calculating and visualizing information from a Spotify dataset. The provided data includes top streamed songs in the first half of 2023 and various features about those songs.

Your application must meet the following design specifications:

- Prompt the user to enter a song feature based on the provided menu (already coded in the given template).
- Some menu options do not need to be developed. If the user enters 0, 1, 5 or 6, nothing should happen and the user should be prompted for their next entry.
- Menu option 2: If the user enters 2, calculate the total span of release years that are included in the data. Then output the artist, key, and mode of the oldest song using the required format (see screenshots below). This calculation/analysis must be done in a function called age_stats that

accepts the menu input value and prints the required information. It should not return any values.
- Menu options 3, 4, 7, 8, 9, 10, 11, 12, 13: For each of these menu options, calculate the highest value, lowest value, and mean value of the desired song feature. These calculations must be done in a function called feature_stats that accepts the menu input value and prints the required information. The function should return the index location of the highest value, which should then be used to find and print the title of the song with the highest value.
- If the user enters -1, the entry loop should end and move to the plotting stage.
- If the user enters a value outside of the provided range, the user should be warned and prompted to try again: "You must enter a valid menu option."
- To conclude the data analysis, generate a plot of danceability vs. beats per minute of all songs. Your plot formatting should match the given example below.
- Your plot will be graded manually. You should save a .png version of your plot and upload it into your working directory. Your code must output the same plot that was uploaded.
- A screenshot with example terminal and plot behaviour is included below the rubric. Your output should match the given format exactly.
- The function stubs are provided in the given template- you must use the given name and argument/return value structure as written here. Your functions may not access global variables aside from those presented in the data section.
- Your code must follow the conventions discussed so far in the course (names_with_underscores, four spaces for indentations, spaces between variables/operators, comments throughout, etc.). All functions and classes must include docstring documentation. You may only use built-in Python functions that support compound data structures, user entry, or casting (such as len(), input() or int()). You may use any NumPy and Matplotlib functions. You may not import any modules beyond the those specified in the template.

Data notes: All data is imported from the included csv file. You may not modify the content, order, or location of the csv file. Your code may be tested against a different data file. You may not hardcode (manually add into the code) any data values beyond those given in the data section.

Plotting notes: To view your matplotlib output, you will need to click on "Open Desktop" in the top, right-hand corner of the environment window. This will open a simulated desktop where you can view and save your plot directly in the working directory. If you see the following warning in the terminal, you may ignore it: QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-mysql'

Bonus Specifications:

For optional bonus marks that may be applied to your total course grade, the following additional tasks may be completed:

- Create a Song class that constructs a Song object when given a single row of data. Each column should be saved as an attribute. Title, artist, key, and mode should be saved as strings. The remaining ten features should be saved as float values. Another attribute called "percentages" should be defined as a list containing all of the values for danceability through to speechiness.

- After the required plot, the user should be prompted to enter any row number (you may assume that only valid entries will be given). Create a Song object from the data located at the given row. The row value corresponds to the numpy array, not to the original spreadsheet.
- Use the Song object to plot the percentage values (danceability through speechiness) as a bar graph for the chosen song. Your plot formatting should match the given example below.
- Your plot will be graded manually. You should save a .png version of your plot and upload it into your working directory. Your code must output the same plot that was uploaded.
- Remember to document your class!

Assessment Submission:

The final submitted file must be named spotify_program.py and must be contained in the same directory as the provided data. You may choose to code in a different IDE, but make sure you copy the given template code and that your final submission is done through this zyLabs section. You may modify and resubmit your code as many times as you'd like before the deadline.

No automatic tests will be available until November 20th. You should debug/test your code via the terminal and determine the intended execution on your own. After November 20th, you must press **"Submit for grading"** at least once to receive a mark for execution. At that time, you will see some tests performed against your code- this is only part of your grade. The remaining elements will be graded manually by a TA. A detailed rubric is provided at the end of this page.

Frequently asked questions/answers will be provided via the D2L discussion boards. You should double-check your solution against the answers posted there before submitting.

LAB ACTIVITY 16.1.1: Portfolio Project 3 Submission 39 / 32

Submit for grading

Coding trail of your work What is this?

```
11/24 F 0 F 10 ,10 ,1 ,10 ,10 ,10 ,10 ,10 ,10
```

Latest submission - 10:53 PM MST on 12/01/23 Total score: 39 / 32

Only show failing tests

Open submission's code

1: Commenting and Syntax 4 / 5

This is a manual score.

Comment

```
(-1) No docsting comments
```

2: Function Return Value 100 / 100

Checks that the feature_stats function returns a value.

This test case's results were hidden by the instructor

3: Program Repetition 100 / 100

Checks that the program continues to prompt for user input with the required phrase until -1 is entered.

This test case's results were hidden by the instructor

4: Oldest Song Key and Mode 100 / 100

Checks that the correct key and mode was returned for the oldest song in the spreadsheet.

This test case's results were hidden by the instructor

5: Top Song in a Selected Feature 100 / 100

Checks that the program returns the top song associated with the highest value for a selected feature.

This test case's results were hidden by the instructor

6: Lowest Value of a Feature 100 / 100

Checks that the program returns the correct lowest value for a selected feature.

This test case's results were hidden by the instructor

7: Oldest Song Artist 100 / 100

Checks that the correct artist was returned for the oldest song.

This test case's results were hidden by the instructor

8: Span of Years 100 / 100

Checks that the span of years between the oldest and newest song is correct.

This test case's results were hidden by the instructor

9: Invalid Feature Input 100 / 100

Checks that the program recognizes invalid input, prints the required message, and continues to prompt for user input.

This test case's results were hidden by the instructor

10: Mean Value of a Feature 100 / 100

Checks that the program returns the correct mean value for a selected feature.

This test case's results were hidden by the instructor

11: Highest Value of a Feature 100 / 100

Checks that the program returns the correct highest value for a selected feature.

This test case's results were hidden by the instructor

12: Code Structure and Semantics 5 / 5

This is a manual score.

Comment *Not specified*

13: User Interface and Functionality 20 / 12

This is a manual score.

Comment *Not specified*

14: 1 Day Late (-10%) 0 / -3.2

This is a manual deduction.
Your instructor hasn't graded this test yet.

15: 2 Days Late (-20%) 0 / -6.4

This is a manual deduction.
Your instructor hasn't graded this test yet.

5 previous submissions

9:29 PM on 12/1/23	10 / 32	View
9:27 PM on 12/1/23	10 / 32	View
9:15 PM on 12/1/23	10 / 32	View

9:08 PM on 12/1/23	10 / 32	View

Portfolio Project 3 Rubric

32 marks + 8 bonus marks, 3+% of overall grade

Your code must successfully run for the execution to be graded. Partial marks may be given for each criterion listed below.

Commenting and Syntax (5 marks):

(1) Your name must be included in the file header

(2) Comments must be included throughout the code to explain the functionality. Any user-defined functions are fully documented using docstrings (including summary, parameters, and return values)

(1) All variables and functions have clear and useful names that use lowercase words separated by an underscore

(1) Code is clearly indented and spaces are included between variables and operators

One mark will be deducted for each error or missing component, up to a maximum of 5 marks

Code Structure and Semantics (5 marks):

(2) Solution contains at least two user-defined functions that are outlined as above

(2) No values are hardcoded beyond the given data values*

*Programs with significant hardcoding may lose additional marks in other categories

(1) Functions do not access global variables beyond the given data values

One mark will be deducted for each error or missing component, up to a maximum of 5 marks

User Interface and Functionality (12 marks):

(1) Program runs continuously until the user chooses to end

(1) Program allows the user to choose between menu options

(1) Program checks that the menu input is valid

(5) Program outputs the required statistics for menu options 2-4 and 7-13

(1) The program output follows the required format

(2) The required plot shows the correct data values

(1) The required plot is formatted and labelled correctly

One mark will be deducted for each error or missing component, up to a maximum of 12 marks

Execution (10 marks):

Example execution behaviour is provided in the screenshot below for your reference. Your interface must match exactly.

Your program will be run against a variety of test cases, including select menu inputs and repeated entry. Tests will look for correct numerical values as well as correct song titles. Invalid menu input will also be tested.

Bonus Rubric (8 marks):

(1) Song class is full documented and defined according to correct naming conventions

(1) User is prompted to provide a row value

(1) Song object takes in a single numpy array row

(2) All required attributes are correctly initialized

(2) The bonus plot shows the correct data values

(1) The bonus plot is formatted and labelled correctly

Figure 16.1.1: Execution Behaviour (including bonus prompt)

```
ENDG 233 Spotify Statistics

Song analysis options:
0 title
1 artist(s)
2 release
3 num_of_streams
4 bpm
5 key
6 mode
7 danceability
8 valence
9 energy
10 acousticness
11 instrumentalness
12 liveness
13 speechiness
Choose -1 to end the program.

Please enter a song feature to analyze: 0

Please enter a song feature to analyze: 1

Please enter a song feature to analyze: 2
Span of years: 81
Artist of oldest song: Bing Crosby . John Scott Trotter & His Orchestra . Ken Darby Singers
Key and mode of oldest song: A Major

Please enter a song feature to analyze: 3
Highest value: 3703895074
Lowest value: 2762
Mean value: 532959390
Top song in selected feature: Blinding Lights

Please enter a song feature to analyze: 7
Highest value: 96
Lowest value: 23
Mean value: 66
Top song in selected feature: Peru

Please enter a song feature to analyze: 14
You must enter a valid menu option.

Please enter a song feature to analyze: -1
Bonus - Enter any row number: 7
```

Figure 16.1.2: Required Plot

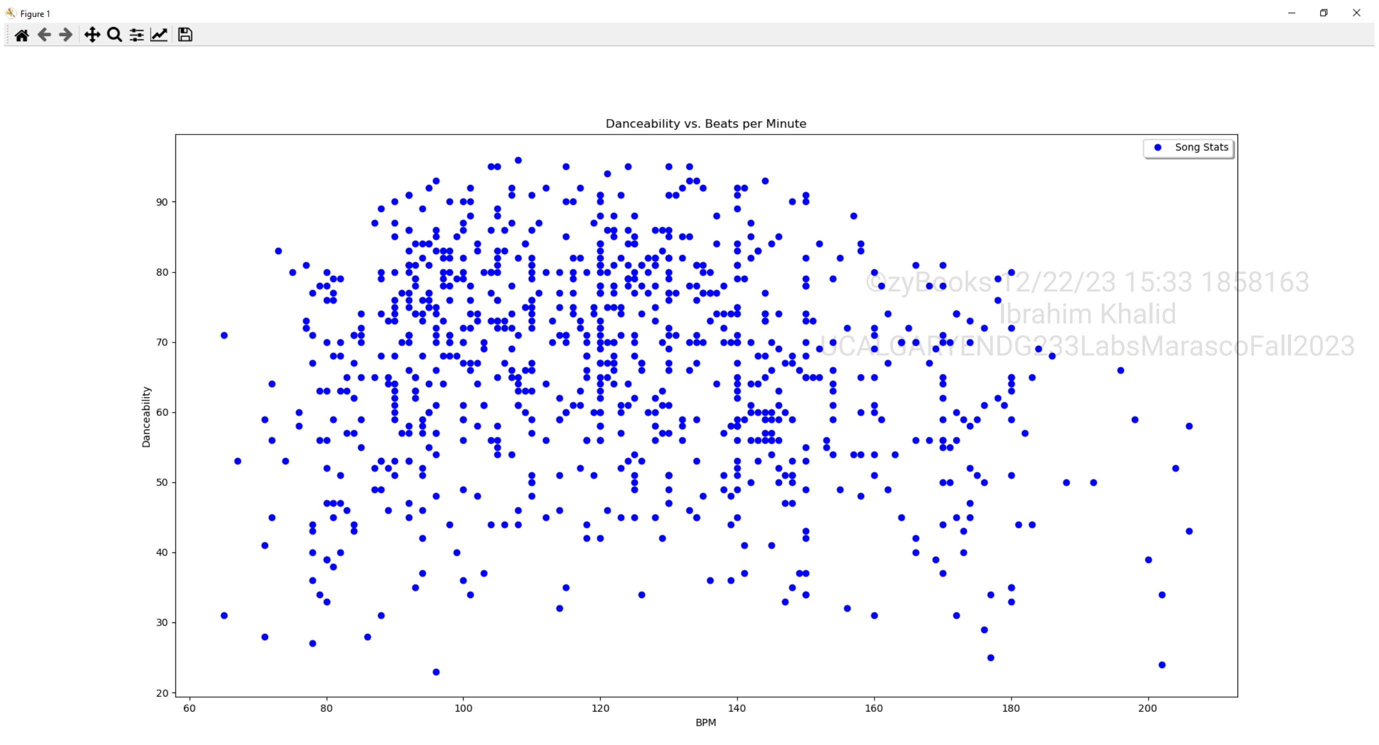
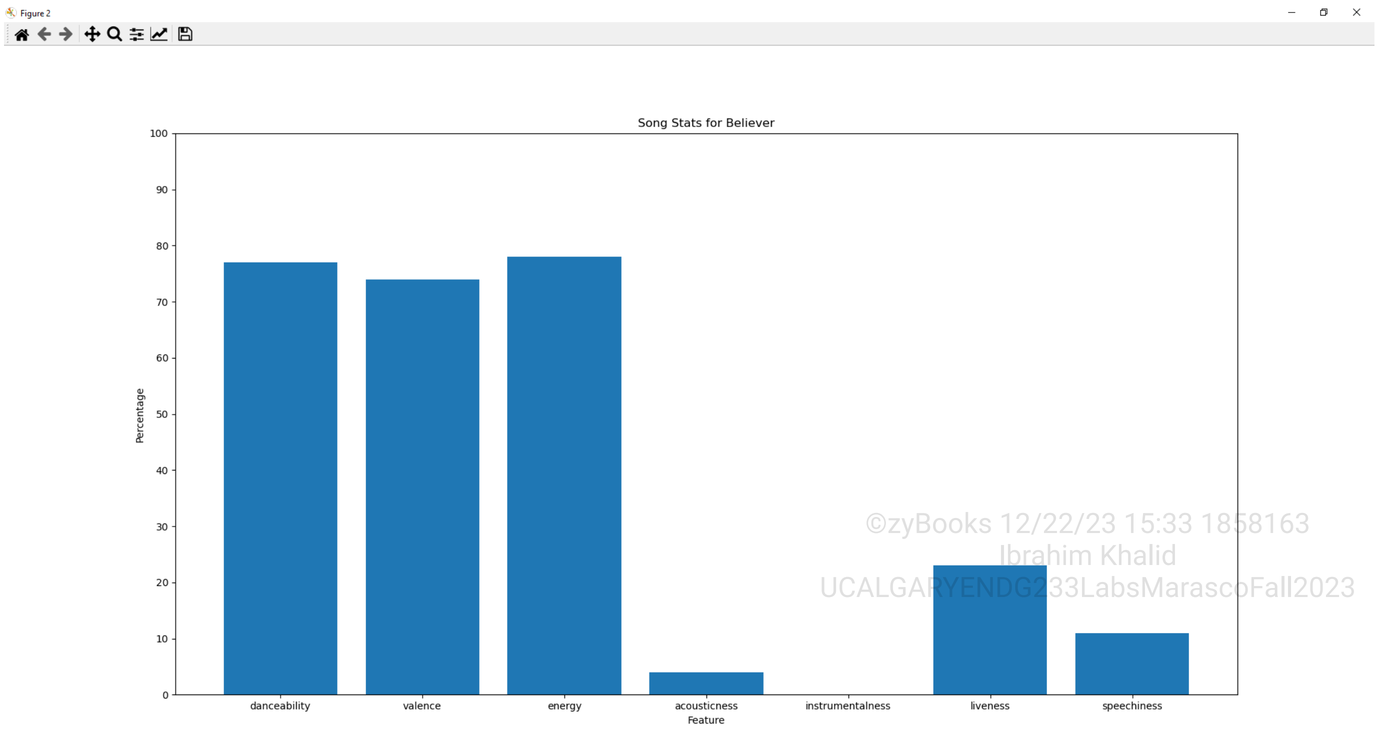


Figure 16.1.3: Bonus Plot



Spotify data sourced and modified from https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023/. License is not specified. Contains data up to September 2023. Most recent song in the dataset is from 2023-07-14.

Column explanations:

title: Name of the song as it appears on Spotify.

artist(s): Names of the artists involved.

release: Year that the song was released.

num_of_streams: Number of times the song has been streamed.

bpm: Beats per minute (BPM), a measurement of musical tempo.

key: Scale of the song.

mode: Modal scale of the song.

danceability: Percentage indicating how suitable the song is for dancing

valence: Percieved positivity of the song's musical content (percentage)

energy: Perceived energy level of the song (percentage)

acousticness: Amount of acoustic sound in the song (percentage)

instrumentalness: Amount of instrumental content in the song (percentage)

liveness: Amount of live performance elements (percentage)

speechiness: Amount of spoken words in the song (percentage)