

Project 8

[Start Assignment](#)

Due	Sunday by 11:59pm	Points	50	Submitting	a file upload	File Types	zip
------------	-------------------	---------------	----	-------------------	---------------	-------------------	-----

Goals

At the completion of this assignment, you should:

- have gained experience working with dictionaries
- have gained additional experience working with lists
- have gained additional experience working with input, strip() function and while loops
- have gained additional experience working with conditionals, logical and Boolean operators, and break
- have gained additional experience using functions
- have gained additional experience creating more complex applications
- have gained additional experience working with more complex data structures using lists and dictionaries

Expected Output and Grading Rubric

You can view the expected output and grading rubric for this lab at the end of the assignment.

Overview

For this assignment you will leverage your experience with the last assignment creating a list manager to create a group manager.

Your group manager allows you to create a group with properties that you define when you create a group. Your group manager will support the following use cases:

- Create a new group by group name and defining properties for the group
- Add data to a group
- List groups
- List data for a group
- Display help screen with commands

Storing Group Information

To store your group information, you will use lists and dictionaries. You will use a list whenever you simply need to store a list of data, and dictionaries whenever you need to store data with an associated name.

Below is the data structure you will use for this assignment, where `[]` represents a list, and `{ }` represents a dictionary.

```
{
  'group_name': {
    '_fields_': [ ],
    '_data_': [ { }, { }... ],
  },
  ...
}
```

For example, suppose you wanted to create a group called Songs to store a song title, and the song artist.

- **Group:** Songs
- **Properties:** Title, Artist

If you wanted to store *Believer* and *Natural* by Imagine dragons, your data structure would look like the following:

```
{
  'Songs': {
    '_fields_': ['Title', 'Artist'],
    '_data_': [
      {'Title': 'Believer', 'Artist': 'Imagine Dragons'},
      {'Title': 'Natural', 'Artist': 'Imagine Dragons'}
    ],
  },
}
```


If you created another group, such as Albums, and wanted to store *Evolve* and *Origins* by Imagine Dragons, your data structure would look like the following:

```
{
  'Songs': {
    '_fields_': ['Title', 'Artist'],
    '_data_': [
      {'Title': 'Believer', 'Artist': 'Imagine Dragons'},
      {'Title': 'Natural', 'Artist': 'Imagine Dragons'}
    ],
  },
  'Albums': {
    '_fields_': ['Album', 'Artist'],
    '_data_': [
      {'Album': 'Evolve', 'Artist': 'Imagine Dragons'},
      {'Album': 'Origins', 'Artist': 'Imagine Dragons'}
    ],
  },
}
```

The data structure may look complicated, but focus on identifying lists and dictionaries. When using a dictionary, you simply add an item to a list. When using a dictionary, you need a dictionary key. A list item can be a dictionary.

From the above data structure, the `'_fields_'` dictionary item is a list of fields for that group. The `'_data_'` dictionary item will hold the data as a list of dictionary items.

Visualization Tip

You can use `print()` statements to display your dictionary to see the current state of your dictionary structure; however, the output in the IDLE Shell will not be formatted for ease of visualization. You can use an online [Python Formatter](https://codebeautify.org/python-formatter-beautifier)  (<https://codebeautify.org/python-formatter-beautifier>) to reformat the dictionary contents to resemble the output shown above with the sample dictionary.

Programming Tips

Remember that after running your Python code, any variables defined outside of functions is still available within the IDLE Shell. As you run and test your program, you can then work with the variable. This tip will help you in determining the proper syntax for working with the variable storing the group data of dictionary and list data.

You can also use a single line of code to access the different data elements in the group dictionary variable. The following examples illustrate the correct syntax to access data from the above songs and albums data, assuming `d` is the group dictionary variable:

`d['Albums']` : Access the Albums dictionary/group

`d['Albums']['_fields_']` : Access the fields stored with the Albums dictionary/group

`d['Albums']['_data_']` : Access the data stored with the Albums dictionary/group

`d['Albums']['_data_'][0]` : Access the first list item of data stored with the Albums dictionary/group

`d['Albums']['_data_'][0]['Artist']`: Access the Artist dictionary entry of the first list item of data stored with the Albums dictionary/group

Assignment

Part I: Required Functions

Create a file called **p8-group-manager.py** for your group manager program.

Below is a list of **required** functions, where `d` is a dictionary parameter. You should create all of these functions in your code file with a single statement `pass`. Remember to replace the `pass` statement with your actual code as you implement each function.

1. **create_group(d)**: Creates a new group and group properties
2. **list_groups(d)**: Lists group names and number of properties
3. **add_group_data(d)**: Adds data to a group for each group property
4. **list_group_data(d)**: List all data for a group

Part II: Group Variable

You will require a dictionary variable to hold your group. The group variable should be declared outside of any function, and initially be empty.

Part III: Primary Input Loop

You will require a primary input loop to prompt for group commands. The loop must only handle the commands, and then call the appropriate function to handle the command. You must use the *strip()* function to remove leading and trailing white space, and the *upper()* function to ensure the command matches. If the Enter key is pressed with no input, exit the program.

The prompt must be: **"Command (empty or X to quit, ? for help): "**.

```
>> Welcome to Group Manager <<
This program creates groups with dynamic properties

Command (empty or X to quit, ? for help):
```

Part IV: Help Command

Your list manager must display and process the following help:

- ?: list commands
- C: Create a new group
- G: List groups
- A: Add data to a group
- L: List data for a group
- X: Exit

```
Command (empty or X to quit, ? for help): ?
?: list commands
C: Create a new group
A: Add data to a group
G: List groups
L: List data for a group
X: Exit
```

```
Command (empty or X to quit, ? for help):
```

Part V: Guardian Code

For this assignment, you do not have to use any guardian code other than the code necessary to make the program work, such as checking for length of input to know when to exit, or checking if a dictionary

item exists.

Part VI: Create New Group

The Create New Group (C) command will:

- Prompt the user for the name of a new group.
 - If nothing is entered, exit the Create New Group.
 - If a group name is entered, validate the group doesn't already exist.
 - If the group does exist, print a warning and re-prompt
 - If the group doesn't exist, prompt for property names
 - When nothing is entered, save the group and the properties for that group

```
Command (empty or X to quit, ? for help): c  
** Create new group **
```

```
Enter group name (empty to cancel): Songs  
Enter field name (empty to stop): Title  
Enter field name (empty to stop): Artist  
Enter field name (empty to stop):
```

```
Enter group name (empty to cancel):
```

```
Command (empty or X to quit, ? for help):
```

Part VII: List Groups

The List Groups (G) command will:

- Display a sorted list of groups
 - Display the group name
 - Display the number of properties
 - Display a comma-delimited list of property names

```
Command (empty or X to quit, ? for help): g  
** List of groups **  
Songs : 2 properties (Title, Artist)
```

```
Command (empty or X to quit, ? for help):
```

Part VIII: Add Group Data

The Add Group Data (A) command will:

- Display sorted list of groups (**tip**: call the `list_groups()` function)
- Prompt for a group name
 - When nothing is entered, leave the Add Group Data command
 - If a group name is entered, validate the group exists
 - If the group doesn't exist, display an error and re-prompt
 - If the group does exist, prompt for input for each of the group properties
 - Save the properties for the group

```
Command (empty or X to quit, ? for help): a
** Add group data **
** List of groups **
Songs : 2 properties (Title, Artist)
```

```
Enter group (empty to cancel): Songs
Enter Title: Believer
Enter Artist: Imagine Dragons
```

```
Enter group (empty to cancel): Songs
Enter Title: Natural
Enter Artist: Imagine Dragons
```

```
Enter group (empty to cancel):
```

```
Command (empty or X to quit, ? for help):
```

Part IX: List Data for Group

The List Data for Group (L) command will:

- Display sorted list of groups (**tip**: call the `list_groups()` function)
- Prompt for a group name
 - When nothing is entered, leave the List Data for Group command
 - If a group name is entered, validate the group exists
 - If the group doesn't exist, display an error and re-prompt

- If the group does exist, display a line for each stored group properties displaying the property and the property value

```
Command (empty or X to quit, ? for help): 1
** List group data **
** List of groups **
Songs : 2 properties (Title, Artist)

Enter group name (empty to cancel): Songs
0 Title = Believer, Artist = Imagine Dragons
1 Title = Natural, Artist = Imagine Dragons

Enter group name (empty to cancel):
```

```
Command (empty or X to quit, ? for help):
```

Submissions

For this assignment you may include the optional labs, Lab 9 and Lab 10, for extra credit.

*Remember that all submitted **Python** files must include the header docstring at the top of each file.*

Create a single compressed file that contains the Python files. To create the compressed file, select the deliverable Python files, right-mouse click on the selected files, and select Compress (Mac) or Compressed (Windows). A new file will be created that contains all of the deliverable Python files.

Rename the compressed file **p8.zip**. Use the Project 8 Canvas site to submit/upload the compressed/ZIP file.

The compressed file will contain the following deliverables (point values in parentheses).

- (50) p8-group-manager.py
- (2) Extra credit: lab9-dictionary.py
- (6) Extra credit: lab9-student.py
- (8) Extra credit: lab10-tuples.py
- (2) Extra credit: lab10-data.txt

Once uploaded, revisit the Project 8 page in Canvas and confirm the file was uploaded.

Remember, you are responsible to submit your assignments before the deadline. Late submissions will not be accepted.

Grading Rubrics

p8-group-manager.py

- Verify create_group() implemented and called
- Verify list_groups() implemented and called
- Verify add_group_data() implemented and called
- Verify list_group_data() implemented and called
- Verify help prompt
- Verify strip() used
- Verify upper() used
- Verify validated input using while loops (minimum 5 loops)
- Verify group listing is sorted
- Validate help (?) command works
- Validate group command works
- Validate group requires field prompt
- Validate list groups works
- Validate add group data works
- Validate add group data works
- Validate list group data works
- Validate list group display field data works