Python Programming

Unit 3 Lab: Object-Oriented Programming

Overview

Welcome to the Unit 3 lab!

Our goal is that at the end of the Unit 5 lab, you'll have an app that prints out the Rotten Tomatoes rating for any movie a user enters. We're getting closer!

Right now, let's use object-oriented programming concepts to improve our code. Specifically, we'll be using dictionaries and classes.

Deliverables

You're going to continue building this locally from where you left off with the last lab. You'll write all of your code in the same `movie_app.py` file.

Run the file from the command line to check your work.

Reminder: On your laptop, you can run the file from your command line with the following:

```
python movie_app.py
```

> **Hint:** Make sure you are printing something out with the `print` statement. Otherwise, you won't see any output from running your program!

Requirements:

1. You have a `Movie` class.
2. Have docstrings on each function.
3. Your `main` function always prints: `The movie Back to the Future has a rating of 4 The movie Blade has a rating of 4 The movie Spirited Away has a rating of 4`

Then: 1. If `search_or_ratings` is `1`, your program prints an indented list of movies: `Back to the Future Blade Spirited Away`

1. If `search_or_ratings` is `2`, your program prints `The rating for Moana is 7`.

Directions

Augment the code you wrote for the Unit 2 lab.

Part 1: Docstrings

1. First, for each of your existing functions, add docstrings to document what each function does. (It doesn't hurt to add them to all functions, including `main()`!).
   1. Remember that a `docstring` is made with `"""`. For example, your `main()` could look like this: `def main(): """ Main is the entry point into the program, and it calls into the search or ratings functions depending on what the user decides to do. """`
   2. As you go through this lab, update your docstrings and create new ones for new functions. It will help you, and others in the future, keep track of what each function does.

Part 2: Adding a Class

Part 2a: The Class

1. OK, let's get going! Let's create a class. We're going to have several movies, each of which will have a title and a rating. We can use a `Movie` class as a scaffold to create many movie objects. Near the top of your file, create a class, `Movie`, that takes an argument of `object`. Your `Movie` class should have three functions:
   1. `__init__`, which will take in `self` and `movie_data` (this will be a dictionary containing the title and rating of each movie). `__init__` will set a member variable, `movie_data`, to equal the `movie_data` dictionary passed in.
   2. `get_movie_title()`, a getter function that returns the value of the `title` key in the `movie_data` dictionary.
   3. `get_movie_rating()`, a getter function that returns the value of the `rating` key in the `movie_data` dictionary.
2. Now that we have a `Movie` class with getter functions, we won't need `print_movie_rating()` or `print_movie_title()`, so delete them.
   1. Then, in your `main()` function, set the existing `elif search_or_ratings == 2:` to `print_single_movie_rating`, and set `else` to print `"Error: Your input must be 1 or 2!"`.
3. Now that we have a class, let's make a function that creates `Movie` objects.
   1. Create a function called `return_single_movie_object()` that takes two arguments, `movie_title` and `movie_rating`.
   2. Have it create and return a `Movie` object with those values.

Part 2b: The Class Objects

Now we can make `Movie` objects. Let's look at all the places we're currently using regular movie titles. Can we replace some of them with `Movie` objects? 1. First, let's look at `print_all_ratings`. Right now, it loops through a list of movie titles and prints out each one. We want it to make `Movie` objects instead, but we don't have ratings. Let's use placeholders and set each rating to `4`. 1. Change `print_all_ratings` to loop through the movie title list and call `return_single_movie_object` on each title, passing in the title and `4` in the list for parameters. Then, print out `"The movie", title, "has a rating of", rating)`. Use the object's getter functions for the `title` and `rating`.

1. Next, let's look at `print_single_movie_rating`. Right now, it prints out the variables for

   `movie_title` and `movie_rating` that we have declared at the top of the file. However, we're going to want it to print out the rating and title of any `Movie` object so that it's useful to the user when they look for a movie.
    1. Change `print_single_movie_rating` to take in a movie title, `movie_query`.
    2. Then, create a `Movie` object from the title that's passed in. In `print_single_movie_rating`, call `return_single_movie_object` on that title, with a rating of `7` (later, the rating will be accurate; for now, let's hard-code it.)
    3. Now, change the `print` statement to use the getter functions on the new `Movie` object.
    4. We're calling `print_single_movie_rating` in `main()`, so we'll need to change that to provide an argument. Pass it a title of a movie you like, such as `"Moana"`.
2. Now that we aren't using `movie_title` and `movie_rating` at the top of the file, you can delete them.

Pro tip: If you get confused as to how all these functions interact, it's helpful to draw it out on a piece of paper. Check back to the Requirements section above for an overview.

Phew. You're done! Awesome job.