UDACITY

Project Evaluation

STUDENT

Justin Locke

COURSE

Introduction to Computer Science

Dear Justin,

Thanks for your resubmission. I'm happy to tell you that your final project meets specifications! You've done a great job making the changes noted in the previous evaluation—namely, passing the autograder, providing readable comments in your code, and using the simpler Pythonic control constructs.

Here's something to keep in mind that can simplify your code. Remember that a tuple data structure in Python is immutable, while lists are mutable. That means that you can simplify the add_connections procedure, where you append the new connection to a friend_list *and* update the network item.

```
#then add the friend to the friend_list
friend_list.append(user_B)
# and update the network and return the result
network[user_A] = friend_list, games_liked
```

Because user_B has been appended to friend_list, mutating the list, updating the network is unnecessary. To get a better understanding of this behavior, try running this code and see what gets printed:

```
collection = ([],[])  # tuple with two lists
first_list = collection[0]
print collection  # prints ([],[])

first_list.append("Hello")  # add to the first list
print collection  # prints (["Hello"], [])
```

Keep in mind the properties of the data structures a program uses is always a good thing to note when programming.

Congratulations on completing the final project!

Code Functionality — Exceeds Specifications

- Code completes the desired tasks and uses a data structure that allows for efficient execution of the code even with large inputs.
  Comment: You have chosen a good data structure for the network. Another possibility would be to use a dictionary with dictionaries: {'John': {'games' : ['The Movie: The Game', 'The Legend of Corgi', 'Dinosaur Diner'], 'connections': ['Bryant', 'Debra', 'Walter']}, ...}. One advantage of doing this is that you will be able scale the network with new categories (sports or movies, for example) without needing to allocate new tuple collections each time something changes.
- All procedures function as specified in the comments preceding each one.

Use of Control Statements — Meets Specifications

- Selection of control statements is always appropriate.
- Looping control statements are appropriately used to avoid repetition in the code.
- Branching control statements are used correctly, in a way that is not difficult to understand.
  - Branching control statements can be simplified in the code. For example, in the following snippet:

    ```
    # ...
    if condition:
        return value
    else:
        return None
    ```

    The `else` clause can be dropped and be simplified to

    ```
    if condition:
       return value
    return None
    ```
- Appropriate Python syntax is used with control statements, making them more intuitive.

Use of Procedures — Exceeds Specifications

- Helper procedures are defined effectively and contribute to code functionality and clarity.
  Comment: Excellent job with creating and utilizing helper procedures! This was an awesome way to break up the final project into simpler components.
  The functionality that the helper procedure `comma_parse` provides can be replaced by a simple built-in Python procedure that you have learned in the course. See the Building the Web Index video from Lesson 4 for more information.
- Procedures that are already defined and can be used in other parts of the project are appropriately used.
- Recursion is used correctly and in an intuitive manner when needed.
  Comment: Nice job with the recursion problem and with storing the recursive call in a variable.
- Choice of arguments and/or return values is aligned with the procedure's purpose.

Make-Your-Own-Procedure (MYOP) **Meets Specifications**

- MYOP is well planned and designed, and produces a useful result.
- Comments clearly explain MYOP usage and functionality.
  Comment: Remember that the comments that describe our MYOP should be consistent with your actual procedure definition. For instance, the comments name the procedure `games_my_friends_and_i_like_the_most`, but the procedure is really called game_popularity_tally.

Code Readability **Exceeds Specifications**

- All variables and helper procedures are given useful and meaningful names.
- Appropriate variables are used instead of hard-coded values, clarifying code logic.
- Comments are concise and effectively explain code procedures that are difficult to understand.
  - There are sections of the code where the comments do not reflect the actual executable code. For example, the comments for `path_to_friend` mention that the procedures `recursive_defs` and `find_previous_node` are used within `path_to_friend`, but they are not used nor defined in the code.
- Code is ready for personal review and neatly formatted (indentation, removal of commented code and unnecessary extra spaces/ blank lines).

PROJECT EVALUATION

Project Meets Specifications

Click here to tell us whether this feedback was helpful.