Planet finder project

GitHub repo

Defining the project

1. This program enables the user to navigate through a Json file containing data describing the characteristics of planets and star systems in a video game.
   Using user input and prompting loops, the user can select the parameters they are looking for within the data, and at the end the IDs for those planets are printed.

2. The Json data contains a hierarchical dictionary describing star systems within the game and their characteristics. The main part of that dictionary I will be using is the "planets" list of dictionaries which each describe a planet and its characteristics.

   Json provided by a friend who scraped the data from the game which will be included with the program files.

3. The program is built around 2 main classes
   - Node, holds user choices.
   - LinkedList, arranges the user choices and enables an undoing/redoing functionality.

   - Beyond the classes, the structure of the program is built around 2 main prompting loops. An outer loop for choosing an action [find(), load_star_data(), show_data(), show_config(), end_program()]. While the 2nd loop within the find() function iterates until the user is done choosing planet parameters.

4. I used a couple of functions and variables from one of my own libraries and another project I worked on.
   Functions pulled from other sources:

   [planetfinder_visuals.py] library

   - color_input(), pulled from custom visual library which just colors the default input() function
   - Clear_console(), pulled from custom visual library which resets cursor to top left
   - Various text variables, (red, green, blue, yellow, violet, white/reset, dim(text), bold(text))

   [planetfinder_config.py] pulled from another project and slightly altered to fit the data

   - Config class, handles the json data (updates, loading, ensuring it exists)
   - find_config_path(), finds the running program path and returns where config.json should be

   [other libraries]

   - os, for interacting with the operating system (file handling)
   - json, for handling Json files
   - Msvcrt, for windows, for reading individual characters in the console (used in color_input()
   - Tty, termios, for Linux, for reading individual characters in the console (used in color_input()

5. Data structures:
   - Dictionaries, used extensively for managing function calls and the Json data
   - Lists, used for storing output information such as the ID list printed after searching
   - Tuples, used for storing information which requires interpreting part of the tuple. For example, the parameter entries (“category”, value).

Or for the Boolean tuple used within the find() function which determine when to end the search with or without printing the IDs

- Linked lists and nodes, used for the undo/redo functionality of the program

6. [Test case 1]
   User input:
   1
   air
   jovian_a
   done
   Results:
   [Search Results]
   ==========
   37-6 : Anossa
   655-4 : Areistia
   965-4 : Dhrsvan
   355-6 : Dioneura
   174-5 : Libus
   634-5 : Persaena
   563-5 : Shu
   671-5 : Tvagala
   773-8 : Vasutra

[Test case 2]
User input:
1
continental
no_life
undo
life
methane
done
Results:
[Search Results]
==========
347-4 : Budsvan
791-3 : Tveni
884-5 : Viva

[Test case 3]
User input:
1
martian
dunes
undo
redo
life
done

Results:
[Search Results]
==========
120-2 : Orno
574-3 : Tartaesius

7. I think the program runs well, but it isn't perfectly unbreakable. Specifically, the function that allows the user to add data paths to the config. It's possible for the user to enter a valid path but that path not lead to a Json file with the correct format. Other than that, I made sure to add checks to all other user inputs. So overall, the program is very flexible.