Lab 6: Objects Serialization

INFSCI 0201
Intermediate Programming (Spring 2025)

In this week's lab, you will continue your journey as the developer for the unnamed yet successful RPG game. As you learned in class, objects can be serialized (i.e., converted into a platform-independent text format) and de-serialized (i.e., converted from some text format back into Python objects). Now, the playtesting team is asking you to implement a serialization and deserialization interface for the Inventory class you have built. For this lab, you will extend your work from Lab 5 and Lab 4.

Submission Details

You are to write a Python program that meets the requirements outlined in the Lab 6 Tasks section.

- As with all programming assignments in this course, you must submit Lab 6 through GitHub in the git repository the instructor created for you. You must place your Lab 6 eclipse project in the folder **/labs/lab06/**
- If you did this correctly, this project will be in the directory **/labs/lab06/**

Testing

There will be three main tests for this lab, which cover the following operations:

1. An Inventory object correctly serialized as JSON
2. A complete Inventory object can be restored by deserializing a given JSON string.
3. Provide a Docstring or other form of documentation for the structure of your serialized JSON file.

Note: Style guide violations will count as test failures! Be sure you're strictly adhering to the Python coding style guide on Canvas.

Lab 6 Tasks

- Give all *item* type objects a new interface named *to_json()*
    - t*o_json()* method will convert the data in the *item* instance to a JSON encodable object (e.g., dict) and return the JSON encodable object to the caller
    - The JSON returned by this to_json() method should contain all the necessary attributes that can be used to make a copy of the item.

- Modify the *Inventory* class and add a new method called *to_json()*
    - t*o_json()* method will return a JSON-encodable object that contains all the information on the inventory itself and all the items stored inside the inventory and return the JSON-encodable object back to the caller.
    - The JSON returned by this to_json() method should look something like the following:

- Provide a function that can be called in json.dump() function to handle the serialization of the custom class Inventory. Or you can define a JSONEncoder class to achieve the same thing.
    - t*o_json()* interface in the item class and the inventory class should help you in this step.

- For the Inventory class and all the item subclasses (weapon, potion, shield), add a new classmethod named `from_json()` to create new instances using the data from JSON strings. Ideally, the classmethod should only require one parameter, which is the JSON string used to create the instance. But you have the freedom to decide how the signature (number of parameters, parameter types) for the `from_json()` method will be. Make sure to include a docstring to explain the required input and expected output for this classmethod.