CS 106AX Programming Methodologies in JavaScript and Python Autumn 2026

🎮 IG Problem 6: Grand Theft Adventure VI 🎮

Written by Ben Yan

The game you didn't think you needed in your life, and, it turns out, the game you certainly don't need.

Your **Assignment 6** was the most anticipated game release of this year—until GTA VI comes along, that is—and you want to build on its (non-commercial) success by introducing new features 🔥. In fact, if you do have something fun or exciting in mind, you should submit to the **CS106AX Adventure Contest**! ☺

For this IG problem, you'll bring **NPCs**—short for **non-playable character**—into Adventure. While the world you've built so far is full of dazzling rooms and landscapes, from chambers of green gemstone to shimmering crystal bridges, it's sorely missing in human company. Where is the thunder of laughter, the rumbling of footsteps, the mountain ranges that are the pressure of life crushing people closer together?

In fact, the 1977 Adventure game featured a rogues' gallery of hostile entities, e.g., dwarves that threw axes at you, a pirate that stole your treasures and hid them, dragons blocking passages, etc. Good times.

To that end, you're provided with a file, e.g., **`AdventureCharacters.txt`**, that's structured like this.

```
Pirate
PirateLair
GOLDCHEST,PARROT,GUNPOWDER,LONGSWORD
Arrr arrr are you ready kids?
I can't hear you!!!
Who lives in a pineapple under the sea??

Zephyr the Romantic Vampire
OutsideBuilding
VIALOFBLOOD,ELIXIR
Instead of this long, arduous Adventure,
why don't you run off with me instead? <3

Mountain King
HallOfTheMountainKing2
GOLDCROWN
It's not always about the money and gold lad,
it's about Ha Ha Ha Haaland! Haaland! Heia Norway!

Polyphemus the Giant Cyclops
GiantRoom
SHEEPFLOCK,CHEESE,FIREWOOD
Father Poisedon, I beg you, curse this Odysseus and his
men to years of misery, oblivion, and JavaScript code!
```









The file is organized as a sequence of blocks, where each block supplies information on a single non-player character in the game, and obeys this line-by-line structure.

- The first line of this block contains the **character's name**, e.g., **`"Pirate"`**, **`"Mountain King"`**.
- The second line is **the room the character is initially located** in, e.g., **`"PirateLair"`**.
- The third line is a **comma-separated list of the character's inventory or items** at the beginning, e.g., the ***Pirate*** has a ***gold chest, parrot, gunpowder***, and a **longsword**. You may assume each item is incorporated as a valid **`AdvObject`** in game, and don't need to create them yourself.
- The non-empty lines that follow lists the **sentences of dialogue strings this character speaks** when you meet them in the game, i.e. when the player enters the room this character is in.
- A **blank line**, which marks the end of this block (including the last block).

Part I: Defining the AdvCharacter Class

Your first task is to implement most of the **`AdvCharacter`** class, which supports the following methods.

- **`__init__(self, name, room, inventory, dialogue)`**, the constructor, which should initialize internal variables / data structures to support the following getter and setter methods.
- **`getName(self)`**, which returns the character's name as a string.
- **`getRoom(self)`**, which returns the name of the room the character is located in.
- **`getInventory(self)`**, which returns a list of names of the objects the character possesses.
- **`introduce(self)`**, which should print out a string of form **`"Greetings! I am the [playerName]."`**, then print out each string in the player's **`dialogue`** list on a separate line.
- **`grandTheft(self, playerItems)`**, a morally dubious (though realistic to modern gaming) method that adds each item from this character to the passed-in **`playerItems`** list, and clears this character's inventory to be empty. In effect, the Adventure player is doing "grand theft auto."

Note there is no file reading in this part, just the Python class definition syntax you wrote for much of **`AdvRoom`** and **`AdvObject`**. A sample run, with an impression of our dear Professor, is shown below ☺.

```
>>> poohbear = AdvCharacter("Jerry Cain", "CoDa E118",
                ["Messenger Bag", "Fleet Street CD", "Treats for Doris"],
                ["Does that make sense to everyone?", "~does T-pose~"])

>>> poohbear.getName()
"Jerry Cain"

>>> poohbear.getInventory()
["Messenger Bag", "Fleet Street CD", "Treats for Doris"]

>>> poohbear.introduce()
Greeings! I am the Jerry Cain.
Does that make sense to everyone?
~does T-pose~
```



Jerry is certainly a character. Write your implementation of the **`AdvCharacter`** class on the next page.

```
class AdvCharacter():
      """
      Stores and supplies access to information on a non-player character.
      """

      def __init__(self, name, room, objects, dialogue):
            """
            Initializes a new Character with the specified attributes.
            """









      def getName(self):
            """
            Returns the name (string) of the character.
            """


      def getRoom(self):
            """
            Returns the room (string) the character is currently in.
            """


      def getInventory(self):
            """
            Returns a list of objects / items (string) the character owns.
            """


      def introduce(self):
            """
            Prints out a greeting with the character's name and dialogue.
            """









      def grandTheft(self, playerItems):
            """
            Like in GTA, surrenders all this character's items to player.
            """
```

Part II: Creating a Character from Data File

Next, you'll write a method **`readCharacter`**, which takes as an input a **`file`** object, and reads one block of the file to create and return a **`Character`** object initialized with that block's contents. This is similar to the methods **`readRoom`** and **`readObject`**, in that it empowers to your program to be ***data-driven***.

Note that the input **`file`** is not a string filename, but rather the object you get when you write, for instance, **`with open("characters.txt", "r") as file`**. You can imagine **`file`** as a pointer to the line in the file we're currently reading, and take for example the file block below, featuring a Byronic lover ***Zephyr***, who was one of many literary creations of a former CS106AX student who won our Adventure Contest! 👏👏

Your method should read lines of the current block, extracting the **`name`**, **`room`**, **`inventory`** as a list of strings, and **`dialogue`** as a list of strings. Then, use those parameters to create and return a fresh object of the **`AdvCharacter`** class you defined previously.

`file` *Before function call*

`file` *After function call*

```
Zephyr the Romantic Vampire
OutsideBuilding
VIALOFBLOOD,ELIXIR
Instead of this long, arduous Adventure,
why don't you run off with me instead? <3

... next file block starts here ...
```

If the block's first line is empty, i.e., end of file reached, you should just return **`None`**. Otherwise, at the end of the function, **`file`** should point to the starting line of the next block / character in the file.

Present your implementation of **`readCharacter`** below. This method is static in that you don't need an instance of the class to call it, e.g., **`zephyr = AdvCharacter.readCharacter(file)`**.

```
class AdvCharacter():
      @staticmethod
      def readCharacter(file):
            """
            Reads each line from a file block to create an AdvCharacter.
            """
```

Part III: Adding Characters to the Adventure Game

Lastly, you'll be Adventure Prometheus, and use the fire and clay of your imagination to bring each of the characters in the file to life, i.e. by adding them to the **`AdvGame`** class. This will be in the same vein as what you did for the Adventure rooms and objects, that is, reading data files into internal dictionaries.

Your task is to implement **`readCharacters`**, whose input is a filename, e.g., **`characters.txt`**. This method should open that file, and read it one block at a time, instantiating a new **`AdvCharacter`** object for each block. You'll certainly want to use your **`readCharacter`** helper method, and recall that it returns **`None`** when the **`EOF`** (end of file) is reached. A **`while True`** loop is your friend here.

Ultimately, you should populate a dictionary **`self._characters`**, whose keys are the **character names**, and the corresponding value is an **`AdvCharacter` object / instance for that character**. For instance, had we read in the sample file displayed on Page 1, then this dictionary would look like:

```
{
      "Pirate": <AdvCharacter object for the Pirate>,
      "Zephyr the Romantic Vampire": <AdvCharacter object for Zephyr>,
      "Mountain King": <AdvCharacter object for the Mountain King>,
      "Polyphemus the Giant Cyclops": <Adventure object for the Cyclops>,
}
```

Complete the implementation below. Note this belongs to the **`AdvGame`** class, not **`AdvCharacter`**!

```
class AdvGame():
      """ Records the information necessary to play the Adventure game. """

      def readCharacters(self, filename):
            """
            Reads the characters from the data file and stores them here.
            """
            self._characters = {}
```