Adventure Documentation:

Step 0:

This step was quite straightforward as the instructions for the class Room() and its methods were quite clear, especially as it was already specified what and how the connections to different rooms should be represented in the form of dictionaries.

Step 1:

The syntax for reading a file including the use of the .strip() method were already explored in previous assignments. While the instructions suggested a for-loop format to iterate over the lines, the length of each data file is different so I would have to iterate over each line first to determine how many lines there were before iterating over each line again to read the file. It seemed silly to me to iterate over the file twice when you could just read the file during the first iteration, which is why I just used a while-loop with a break at the newline.

A picture containing text, screenshot, font, number

Description automatically generated

Because readline() “remembers” its location in a file, each section of the data file could be read by reusing the same while-loop as above. Otherwise, this step mostly involved using the various Room() methods and following the instructions.

Step 2:

Since we are already supplied with .current\_room in adventure.py, I understood that the goal of the move method was to assign a new room to .current\_room. Since the get\_connection method returns a room given a direction, I used it to assign a new room to .current\_room.

A picture containing text, screenshot, font, line

Description automatically generated

Step 3:

The visited methods were very straightforward. The instructions mentioned adding the set\_visited method to the move method and the is\_visited method to the get\_description method but since we are already checking if a room has been visited in get\_description, why not just set it to visited once we’ve established that a certain room hasn’t been visited yet?

A screen shot of a computer code

Description automatically generated with low confidence

Step 4:

This step was quite simple to implement is it just involved printing things in main if certain commands were entered.

A screen shot of a computer program

Description automatically generated with low confidence

Step 5:

History is basically just a reskinning of the stack class designed for palindrome. Instead of “items” we use “rooms.

A picture containing text, screenshot, font

Description automatically generated

Before each move, the previous current room is pushed to the history stack and 

Using the “BACK” command pops out the previous room and ‘continues’ the main loop.

A screen shot of a computer program

Description automatically generated with low confidence

Step 6:

Everything worked fine here.

Step 7:

I implemented is\_forced by returning whether FORCED was in the connections of the current\_room.



I then added a check to the at the start of the main loop before any commands are taken and fed “FORCED” to the move method if is\_forced is True.

I then modified the move method to skip pushing to history if direction is “FORCED” in order to have “BACK” function correctly.

A picture containing text, screenshot, font

Description automatically generated

Step 8:

Because most of logic and manipulation behind the Item class is handled in other classes, my representation of an item is just a name and a description.

A picture containing text, screenshot, font, software

Description automatically generated

I added an item list to the Room class and an add\_item method that appends an item to the list. I used the same loop I used in step 1 to load the items to a particular room.

A black background with white text

Description automatically generated with low confidence

A picture containing text, font, screenshot

Description automatically generated

My inventory is just a list in the Adventure class. The take and drop methods that I implemented work in basically the same way. They search through either the current\_room.items list or inventory for the item supplied as an argument and first appends the item to the other list before popping the item out.

A picture containing text, screenshot, font

Description automatically generated

Step 9:

To handle conditional movements, I first had to figure out a way to parse the datafile to separate conditional rooms. I managed to do this by using try and except. I realized that if I tried to typecast a conditional room (such as 12/LAMP) to an int, it would raise a ValueError. So, by trying to typecast each room to an int as it is being read from the datafile, I could separate conditional rooms to a separate dictionary in the Room class. I split the conditional room further into the room value and item using the split method. Because the conditional rooms had 3 parameters, I had to create a new method called add\_cond\_connection to attach the connection. I initially accomplished this by using the following form:



Room 13 gave me a lot of trouble here because it had multiple conditional connections with the same item so the key: value pair would be overwritten each time a new connection was loaded. I solved this by making the value of each item key a list of direction: room pairs and appending to the list instead.

A picture containing text, screenshot, font

Description automatically generated

The last thing that I did was to modify the move method in Adventure to go through the inventory each time to check if any of the items show up in the conditional connections in the current room and attempt to use the direction as a key. If a key error is raised, that means that the direction supplied not a conditional connection and regular movement as in step 2 is used.

A picture containing text, screenshot, font

Description automatically generated

Step 10:

Synonym was not very difficult to implement. By using a dictionary to hold the synonyms as the keys and full commands as keys.

A black background with white text

Description automatically generated with low confidence

Improvements:

While I have started implementing type hints to my code, they are not complete, so I finished type hinting the rest of my code.

Since the main function of adventure.py was not wrapped in a main function, I made sure to do that so that the Adventure class can be used elsewhere if desired. I did this by separating the code in the main section into an init\_game function and a play\_game function containing the initialization code and the game logic respectively.

A screen shot of a computer

Description automatically generated with low confidence  
A screenshot of a computer

Description automatically generated with medium confidence

Reading through my code, I noticed that some of the variable names are a little cryptic. For example:   
A picture containing text, font, screenshot, number

Description automatically generated  
Line is obviously a line from the file but what is line\_list? I therefore replaced line\_list with room\_data, connection\_data, item\_data, and synonym\_data to represent the information on each line being split.



This chunk of code to check for items and find conditional connections is also hard to decipher:

A picture containing text, screenshot, font

Description automatically generated

Iterating through the items feels a lot more pythonic and easy to understand what is going on.

A picture containing screenshot, font, text

Description automatically generated

The same change was made for displaying the inventory in play\_game and the take() and drop() methods:

A screen shot of a computer code

Description automatically generated with low confidenceA screen shot of a computer code

Description automatically generated with low confidence

I realized that the whole move method could be split into a conditional movement method and a regular movement method in order to make the code more modular.

A picture containing text, screenshot, font

Description automatically generatedA picture containing text, screenshot, font

Description automatically generated

Though I should probably have done this before all the other changes to check if I broke anything; I finished by creating pytests for each of the classes.