As you set forth on the Adventure project there are several early stumbling points that can arise. The qusetions on this week's section or geared to help smooth over and aleviate those potential problems!

1. The Adventure project is structured around 3 main classes, and thus having a good familiarity with defining and utilizing classes is important. This first problem seeks to refresh your memory on how classes work and practice some of these basics.

   Imagine that you are a shop owner who is interested in writing code to help you better organize, run, and manage your shop. One thing that you are continually needing to manage is tracking the current price and stock of all the items that you are currently selling. As such, you decide to write some code to assist you with this, and what makes the most sense initially is constructing a class to represent a single "Item".

   To begin this problem, your task is to define a class and class constructor which takes in three arguments when the class in instantiated and saves them as internal attributes. These arguments should be:

   - the name of the item as a string
   - the current stock of the item as an integer
   - the current price of the item as a float

   Ensure that upon finishing your class definition you can create objects of the `Item` class type without issue.

   ```
   >>> apple = Item("Apple", 8, 0.75)
   >>> pumpkin = Item("Pumpkin", 13, 3.67)
   ```

   We can do more than just store values in a class though: we can also define methods that allow us to easily work with our class objects. Here you want to add two methods to your `Item` class, though you could certainly easily imagine more that might be useful:

   - A `restock` method. This method should take a single integer as an argument and increase the stock of that item by the provided amount. For instance, if the original stock of `apple` item above was 8, then calling `apple.restock(2)` would result in a new stock of 10. No value should be returned from this method.
   - A `purchase` method. This method should also take a single integer as an argument, where this integer represents the *desired* amount of that item to purchase. If the desired amount is less than or equal to the current stock, then that amount should be subtracted from the stock and the method should return the total amount to be charged to the user. For instance, if there are 8 apples in stock and `apple.purchase(2)` is called, then two apples should be deducted from the stock, and the returned amount to be charged would be 1.50. But if the desired amount to purchase is *greater* than the current stock, then the entirety of the current stock should be sold, but a warning should also be printed stating that they are not getting all of the item they desired. And the returned value should only charge

them for the amount they could purchase. So for instance if there were 8 apples in stock and `apple.purchase(10)` is called, then a warning should be displayed stating that they are only getting 8 apples, and the returned value should be 6.00.

Upon completion, your defined `Item` class should be able to complete the following run:

```
>>> ball = Item("Red ball", 10, 2.50)
>>> print(f"Trying to purchase 6 balls. Charged: {ball.purchase(6)}")
Trying to purchase 6 balls. Charged: 15.00
>>> print(f"Trying to purchase 6 balls. Charged: {ball.purchase(6)}")
Warning: Ran out of Red balls! You only got 4!
Trying to purchase 6 balls. Charged: 10.00
>>> ball.restock(8)
>>> print(f"Trying to purchase 6 balls. Charged: {ball.purchase(6)}")
Trying to purchase 6 balls. Charged: 15.00
>>> print(f"Trying to purchase 6 balls. Charged: {ball.purchase(6)}")
Warning: Ran out of Red balls! You only got 2!
Trying to purchase 6 balls. Charged: 5.00
```

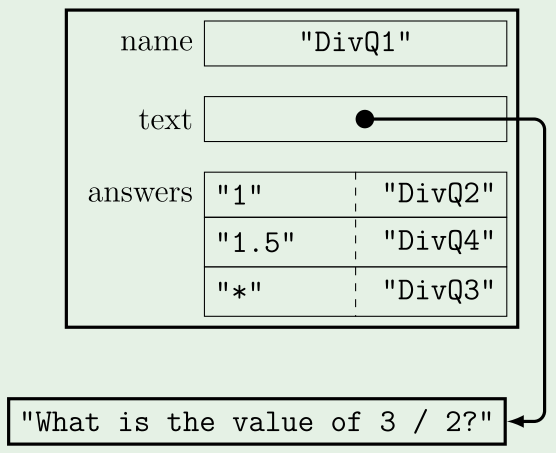
2. **Diagramming the internal data structure**

Before you write the code for the Adventure game, it is important to understand the teaching machine program presented in Chapter 12, which has a nearly identical structure. This problem reinforces your knowledge of the Teaching Machine and begins the process of converting from the Teaching Machine to the Adventure model.

In class and in the text you saw how the internal data structure of the teaching machine program was constructed. Shown below is the human-readable text and corresponding data structure for the first question to serve as a reminder.

Question 1 Representations

```
DivQ1
What is the value of 3 / 2?
-----
1: DivQ2
1.5: DivQ4
*: DivQ3
```



Your goal in this problem is to sketch out the corresponding data structure for the `AdvRoom` class populated with the contents of the first room, shown below:

```
OutsideBuilding
Outside Building
You are standing at the end of a road before a small brick
building.  A small stream flows out of the building and
down a gully to the south.  A road runs up a small hill
to the west.
-----
WEST: EndOfRoad
UP: EndOfRoad
NORTH: InsideBuilding
IN: InsideBuilding
SOUTH: Valley
DOWN: Valley
```

Pay particular attention to what data types are being used to store different items, and come up with reasonable names for the different attributes. In particular, there is an extra piece of information in the `AdvRoom` file than in the Teaching Machine that you need to decide where and how it will be stored.