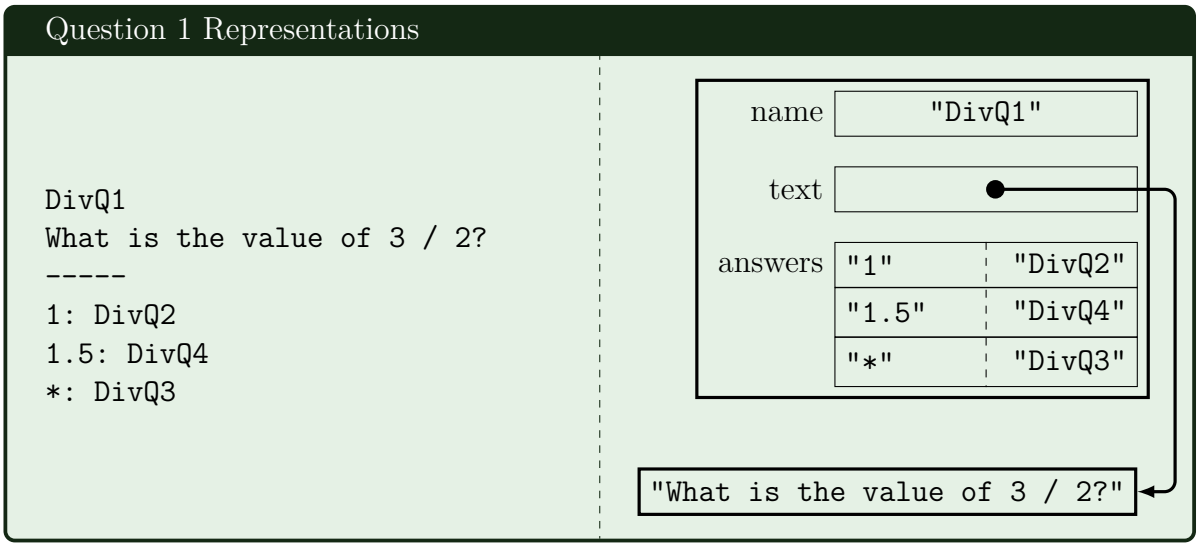
Before you write the code for the Adventure game, it is important to understand the teaching machine program presented in Chapter 12, which has a nearly identical structure. This section includes two problems that reinforce your knowledge of the Teaching Machine and begin the process of converting from the Teaching Machine to the Adventure model.

1. **Diagramming the internal data structure**

   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.



   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.

2. **Adding messages to the teaching machine**

   As written, the `TeachingMachine.py` program provides no feedback when the user gives an incorrect answer. Here you will look into possible ways of implementing such feedback, which will also be directly applicable to the Adventure project.

   (a) Devise a strategy that would let the course designer specify an optional message along with each response. How would you need to adjust your data-file and corresponding internal representation?

   (b) While there are many strategies that you could have come up with, one that is easy to implement and mimics what Will Crowther did with Adventure is to allow questions in the Teaching Machine to have a "FORCED" answer. If a question has such an answer, then the user is **not** prompted for a response, and instead the user is taken immediately on to the new question after the text is displayed. An example of a snippet of the data file might thus look like:

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

DivMsg
The / operator always produces floats.
-----
FORCED: DivQ2

DivQ2
What is the value of 9 / 3?
-----
3: DivMsg
3.0: DivQ4
*: DivQ3
```

   The `FORCED` entry in the question named `DivMsg` tells the Teaching Machine to continue automatically to `DivQ2` without asking the user for an answer, as could be seen in the following run transcript:

```
〉 python TeachingMachine.py
Course name: Python
What is the value of 3 / 2?
> 1
The / operator produces floats.
What is the value of 9 / 3?
> 3
The / operator produces floats.
What is the value of 9 / 3?
>
```

Making the necessary changes in the `TMCourse` class is not difficult, you need only to check if a question has a `FORCED` answer before prompting the user for input, and set the next question accordingly. An example course is provided in the template file for you to test your program against.