Lab: trivia game (continued)

This lab builds on last week's lab. We are going to introduce a timed mode, where the computer computes the time taken to answer a question, and use it to calculate your score.

1. Improve the `Question` class

This class must be in the `question.py` file. You can use the tests in the `test_question.py` test file.

- Make sure the constructor raises an exception when the difficulty provided is not one of "easy", "medium", or "hard".
- The class must implement the dunder method `__str__`. You can reuse the code from `as_string`!
- The `Question` class has a new method: `get_score`.
  - This method takes an argument: the time taken to answer the question (in seconds).
  - It returns an integer (the score for this time)
  - If the player spent `elapsed` seconds to answer the question:
    - if `elapsed` is more than 5 seconds, the score is `10 * difficulty`
    - if `elapsed` is less than 5 seconds, the score is `difficulty * (225/elapsed - 7 * elapsed)`
    - difficulty = 1 for easy, 2 for medium, 3 for hard
  - Read the tests!

2. Improve the `QuestionLibrary`

- The class must implement the dunder method `__len__`. It should return the number of questions in the library (no filters).
- The class also has a new method: `get_categories()`. It returns an iterable of **unique** categories in the library.
- Make sure that the `get_questions` work as expected: if a difficulty other than "easy", "medium", or "hard" is used, ignore the filter.

Example

```
l = QuestionLibrary()
l.get_questions(category="Geography", difficulty="easy", number=2)  # returns
2 easy geography questions
l.get_questions(category="Geography", number=2)  # returns 2 geography
questions, any difficulty
l.get_questions(category="Geography", difficulty="whatever", number=2)  #
returns 2 geography questions, any difficulty
l.get_questions(difficulty="hard", number=2)  # returns 2 hard questions, any
category
l.get_questions(number=10)  # returns 10 questions, any category, any
difficulty
l.get_questions(difficulty="whatever", number=10)  # returns 10 questions,
any category, any difficulty
```

Improve the `Game` class

The `Game` class will use the `get_categories` method to display a list of available categories to the player.

The constructor

- Does not take any arguments
- Create a `QuestionLibrary` instance
- Displays the list of categories, with a number
- If the player types enter, use any category
- Ask the player for a difficulty (empty string, or anything else than "easy", "medium" or "hard" = any difficulty)
- Get 10 filtered questions from the library using `get_questions`
- Store the questions in an instance variable

The `play` method

This method loops through all the selected questions, and, for each question:

- display the question text
- display the answer options
- records the current time using `start = time.time()` (make sure you `import time` first)
- ask the player for input (a number, the correct answer)
- ask again if the value provided is not 1, 2, 3 or 4
- records the current time again using `end = time.time()`
- the time spent to answer the question is `elapsed = end - start` (in seconds)
- calculate the score, and adds it to the total score of the player
- the method prints *AND* returns the **total** score of the user.

Submission

Submit your files: `question.py`, `question_library.py`, `game.py`.

Grading

Item	Marks
The tests pass (1 mark per test)	10
The program works according to the instructions, and one can play a game. 3 possible grades: not working, somewhat working, working fine	0/4/10