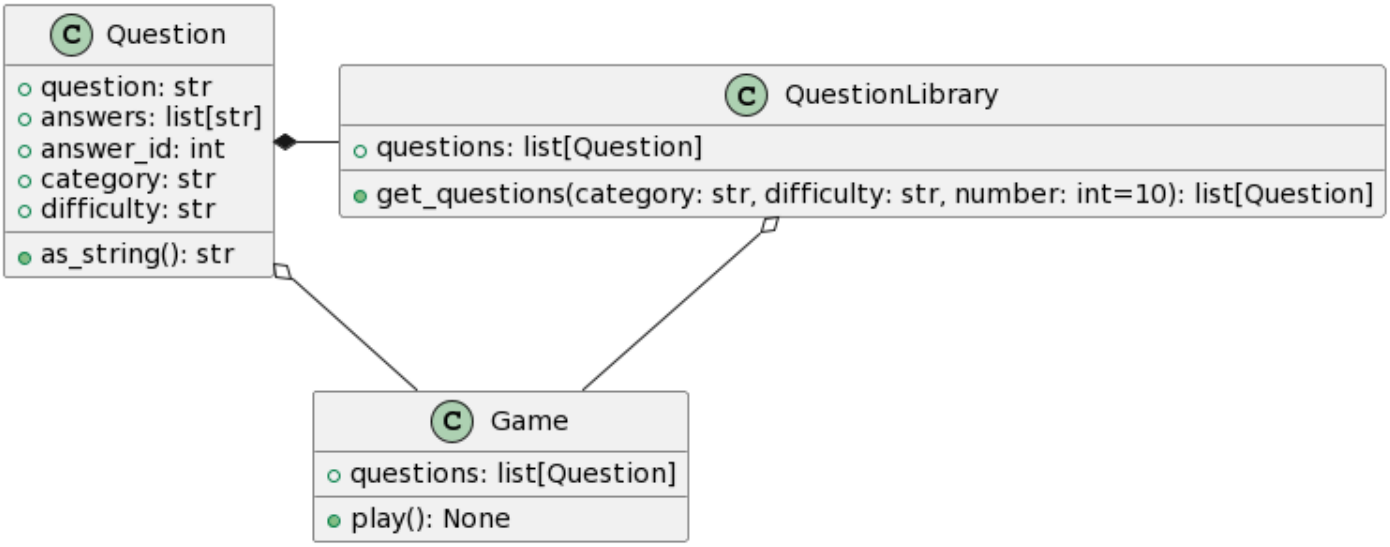
Lab: trivia game

In this assignment, we are going to build a trivia game. The game will display a series of questions, and the player has to choose the right answer among several options.

Class diagram



The questions were pulled from the Open Trivia Database API.

1. Build the `Question` class

You should create this class in the `question.py` file.

The constructor

- Write the constructor: `def __init__(self, question, correct_answer, incorrect_answers, category, difficulty)`
  - `question` is a string
  - `correct_answer` is a string
  - `incorrect_answers` is a list of strings
  - `category` is a string
  - `difficulty` is a string. It must be either "easy", "medium", or "hard".
- In the constructor, make sure you store all the answers (correct + incorrect) in a list `answers`
- Shuffle the list to randomize the order (use `random.shuffle(my_list)`)
- Store the correct answer "index" in the list and make it accessible through the `answer_id` attribute
- ⚠️ Make sure you adjust the indexes. If the correct answer is the first element in the list, `answer_id` is 1 !

Hints

- You can get the index of an element in a list by using `my_list.index(element)`. See above
- `index()` starts at 0!

`__str__` method

This method:

- displays the question
- displays the list of answers, with indexes
- the indexes MUST start at 1 (see above!)
- you can use the `enumerate` function
- this method makes it easy to "print" the question and its answers by returning a string formatted with `\n`

Example

```
>>> q = Question("This is my question?", correct_answer="correct",
incorrect_answers=["wrong", "false", "incorrect"])
>>> q.answers # contains a randomly ordered list of answers - example below
['false', 'wrong', 'correct', 'incorrect']
>>> str(q)
'Question text?\n1 false\n2 wrong\n3 correct\n4 incorrect'
>>> print(q)
Question text?
1 false
2 wrong
3 correct
4 incorrect
>>> q.answer_id
3
```

2. Build the `QuestionLibrary` class

This class loads a JSON file, and holds all available questions. You should create it in the `question_library.py` file.

The constructor

The constructor takes an argument `filename`, whose default value is `trivia.json`. It reads from this file, and stores all questions **as `Question` instances** in a `questions` instance variable.

`get_categories`

This method does not take any arguments, and returns a list of strings: all the categories available across the questions in the library. Each category must only be present once!

`get_questions`

This method takes three arguments:

- a category (str): the name of the category to filter questions (ex: "Geography")
- a difficulty (str): the difficulty level to filter questions. Must be either None, "easy", "medium", or "hard". If it is something else, ignore the argument
- a number (int): the number of filtered questions to return

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=None, 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
```

3. Build the `Game` class

This class manages the full game, with user input and loop control. Create it in the `game.py` file.

The constructor

- Does not take any arguments
- Create a `QuestionLibrary` instance
- Ask the player for a category (empty string = any category)
- Ask the player for a difficulty (empty string = any difficulty)
- Ask the player for a number of questions to answer
    - if the answer is not a strictly positive number, ask again
- Get filtered questions from the library using `get_questions`
- Store the questions in an instance variable
- Initializes an attribute `score` to 0

> Note: you don't have to save the question library in an instance attribute - you only want the filtered questions! Note 2: you don't have to check for correct input for categories and difficulty. It would be nice if you would, though!

The `play` method

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

- display the question text

- display the answer options
- ask the player for input (a number, the correct answer)
- ask again if the value provided is not 1, 2, 3 or 4
- if the answer is correct, add points to the current score (easy = 1, medium = 2, hard = 3)
- you may add informative messages for the player using `print`

Submission

Submit your files to D2L.