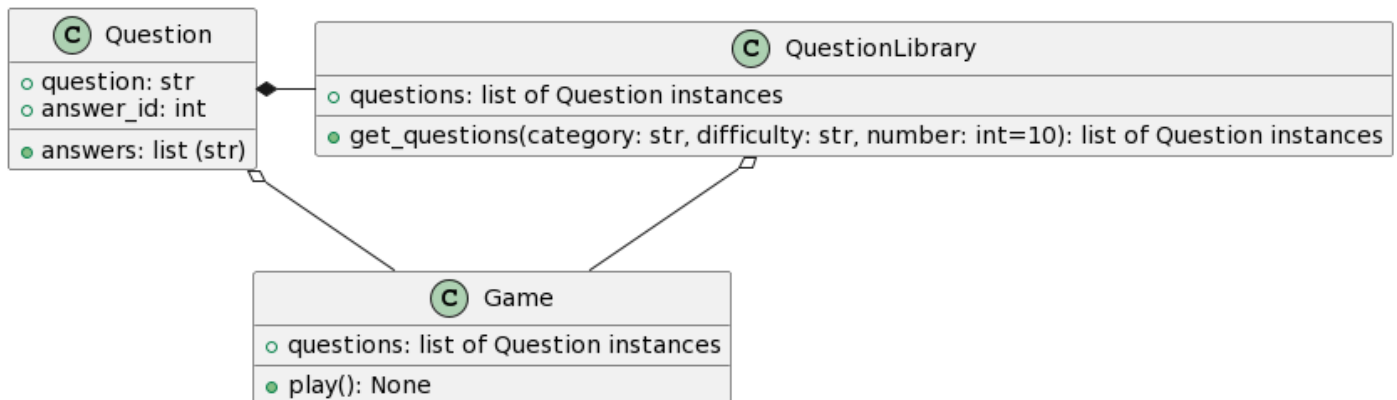


# Lab: trivia game

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



**YOU MUST USE THE CONSTRUCTOR DEFINITION BELOW.**

- Write the constructor: `def __init__(self, question, correct_answer, incorrect_answers, category, difficulty)`
  - `question` is a string: the question. Example: *What is the name of the instructor for the Python course in term 2 ?*
  - `correct_answer` is a string: the correct answer to the question. Example: *Tim*
  - `incorrect_answers` is a list of strings: incorrect answers to the question. Example: *[Alice, Bob, John]*
  - `category` is a string: the category to which the question belong
  - `difficulty` is a string: the difficulty of the question. 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`. See the example below.

## Example

```
>>> q = Question("What is the name of the instructor?", correct_answer="Tim",
incorrect_answers=["Alice", "Bob", "John"])
>>> q.answers # contains a randomly ordered list of answers - example below
['John', 'Bob', 'Alice', 'Tim']
>>> str(q)
'What is the name of the instructor?\n1 John\n2 Bob\n3 Alice\n4 Tim'
>>> print(q)
What is the name of the instructor?
1 John
2 Bob
3 Alice
4 Tim
>>> q.answer_id
4
```

## 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 ONE ARGUMENT: THE FILENAME. NOTHING ELSE!**

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. Can be "easy", "medium", or "hard". If it is something else, ignore the argument
- a number (int): the number of filtered questions to return. If not provided, return all questions available.

### 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

- Takes the following arguments:
  - the filename to pull the questions from (default value = "trivia.json")
  - a category (if no argument is provided, then use any category)
  - a difficulty (must be "easy", "medium" or "hard" - if anything else or no argument provided, use any difficulty)
  - the number of questions to answer (if not provided, use all questions)

The constructor then initializes a game containing the correct number of questions, with the correct category and difficulty if specified:

- Create a `QuestionLibrary` instance
- 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 all your Python files to D2L.