Grading system - 8 marks

In this assignment, you will create Python classes and an application that reads and compare text files.

Concept

The application is used to automatically grade quizzes submitted by students.

The 'instructor' quiz file

This file is a CSV file, and contains the quizz questions. Each line has the following format:

```
<question>,<answer 1>,<answer 2>,<answer 3>,<answer 4>,<correct answer number>
```

For example:

```
How much is 2+2?,1,10,4,0,3
```

The correct answer is answer #3 (`2+2 == 4`, which is the 3rd answer in the list `1,10,4,0`). Note that the answer numbers start at 1 and not 0.

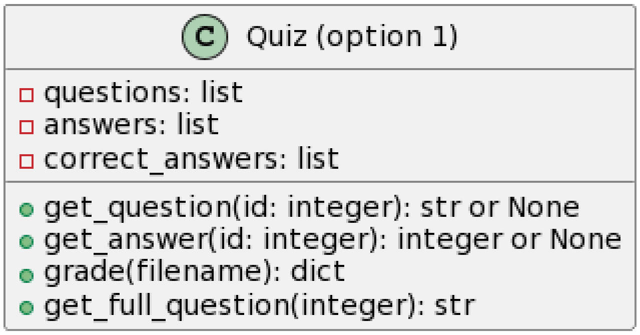
The 'student' quiz file

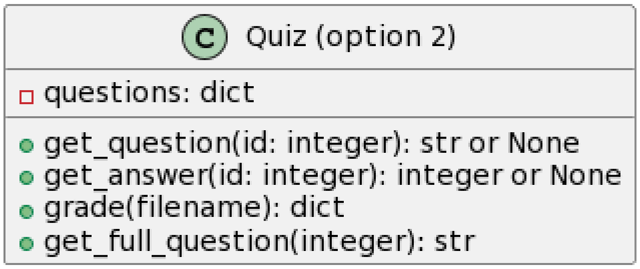
This file is a plain text file, and contains the student answers (one answer per line).

For example, if the student chose option `3` at the first question, and option `4` at the second question, the file will be:

```
3
4
```

Start building the `Quiz` class


C
Quiz (option 1)
questions: list
answers: list
correct_answers: list
get_question(id: integer): str or None
get_answer(id: integer): integer or None
grade(filename): dict
get_full_question(integer): str


C
Quiz (option 2)
questions: dict
get_question(id: integer): str or None
get_answer(id: integer): integer or None
grade(filename): dict
get_full_question(integer): str

> Note: the internal implementation of the `Quiz` class is completely up to you. 2 possible options are offered above, but you may chose the data structures you want, as long as the methods work and the tests pass. The private attributes are not tested.

Constructor

The constructor receives ONE argument: the name of the file containing all questions, answers and correct answers. It opens and reads the file, storing all the information required (see below).

Hint: use the `csv` module

The instructor quiz file is a CSV file. You can read it easily with the CSV module in Python.

```
import csv

# Option 1: read all data into one variable
with open("quiz.txt", "r") as fp:
    reader = csv.reader(fp)
    data = list(reader)

# Option 2: read all data line by line using a for loop
with open("quiz.txt", "r") as fp:
    reader = csv.reader(fp)
    for line in reader:
        # Do something with the line
        print(line)
```

Each "line" of the CSV file is a list. The data is already split based on the `,` separator. For example, in the instructor quiz file, `line[0]` will be the question.

`get_question`

This method takes ONE argument (an integer). It returns a string: the quiz question corresponding to that number. For example, `get_question(1)` returns the **FIRST** question of the quiz.

- if `number` is not an integer, return `None`
- `get_question(number)` returns `None` if `number` is "outside of the list"

- `get_question(0)` returns `None`, `get_question(-100)` returns `None`

`get_answer`

This method takes ONE argument (an integer). It returns an **INTEGER**: the correct answer number corresponding to that quiz question. For example, `get_answer(1)` returns the **CORRECT** answer **NUMBER** to the **FIRST** question of the quiz.

> If `get_answer(1) == 4`, it means the correct answer is the fourth one, not that the correct answer is `4` !

- if `number` is not an integer, return `None`
- `get_answer(number)` returns `None` if `number` is "outside of the list"
- `get_answer(0)` returns `None`, `get_answer(-100)` returns `None`

The `grade` method

This method takes ONE argument: the path of the file containing the student answers. It opens and reads the file with the student answers, and returns a dictionary with the following elements:

```
{
    "score": 2,  # the number of correct answers
    "wrong": [   # a list containing the texts of all questions answered WRONG
        "How much is 2+2?",
        "Which Python keyword do you use to define a function?",
    ]
}
```

> Hint: when you read from a file, all the values are **strings**, but `get_answer` returns an integer! You only need to worry about the answer "numbers", and not the actual values.

`get_full_question` method

This method returns a **STRING**. The string has multiple lines, and contains:

- the question text on the first line
- each answer with their answer number on a separate line (starting at 1)

For example:

```
Which of the following instructors do you like most?
1 Sarah
2 Bob
3 John
4 Tim
```

Or in Python string format: `"Which of the following instructors do you like the most?\n1 Sarah\n2 Bob\n3 John\n4 Tim"`.

Use the `test_quiz.py` file and make sure all your tests pass.