3. **[11 marks] Data classes.**

In this question, we will define and use a data class that represents a person's record when playing Wordle. This data class stores the person's *name* and the *number of guesses made* for each Wordle game the person has played.

Note: the number of guesses includes the "correct guess" at the end of the game if the person successfully guessed the word. So each number of guesses is ≥ 1.

Here is the start of the data class:

```
@dataclass
class WordlePlayer:
    """A record of Wordle games for a player.

    Instance Attributes:
        - name: the name of this player
        - guesses: the number of guesses made by this player for each game that
        they have played, in the order that the games were played.
    """
```

(a) **[2 marks]** Write the appropriate *instance attributes and type annotations* for the data class.

name : str

guesses : list [int]

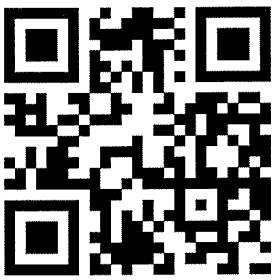
(b) **[2 marks]** Translate each of the following representation invariants into valid Python expressions. Remember to use "`self.`" to refer to instance attributes.

(i) The player's name must be non-empty.

self.name != ''

(ii) The player's numbers of guesses are *all* between 1 and 6, inclusive. (Remember that this must be translated into a single Python *expression*!)

all([1 <= n <= 6 for n in self.guesses])



(c) **[1 mark]** A person named Julia has played Wordle twice, making 5 guesses in the first game and 4 guesses in the second game. Complete the assignment statement below to create a new `WordlePlayer` object that records this data for Julia and assigns the object to variable `julia`.

```
>>> julia = WordlePlayer('Julia', [5,4])
```

(d) **[1 mark]** Julia then plays a third game of Wordle and guesses the correct word on her **first guess**! Write one line of code that *mutates* the object referred to by `julia` to record the result of her third game.

```
>>> julia.guesses.append(1)
```

(e) **[5 marks]** Implement the following function according to its specification. You may **not** define any additional functions in your solution. You may **not** use comprehensions or the built-in `sum` and `len` functions.

```
def average_guesses(players: list[WordlePlayer]) -> float:
    """Return the average number of guesses made by all given players combined.

    Preconditions:
        - players != []
        - any({player.guesses != [] for player in players})

    >>> example_players = [WordlePlayer('david', [2, 3, 4]), WordlePlayer('tom', [5])]
    >>> average_guesses(example_players)  # (2 + 3 + 4 + 5) / 4
    3.5
    """
    sum_so_far = 0

    len_so_far = 0

    for p in players:

        for n in p.guesses:

            sum_so_far += n

            len_so_far += 1

    return sum_so_far / len_so_far
```