Yahtzee Final Project

Stéphanie Larocque

July 18, 2018

See here for rules : https://en.wikipedia.org/wiki/Yahtzee#Rules
Play online (and see rules also) here : https://cardgames.io/yahtzee/

1 Global architecture

The architecture of the program will be separated in 3 classes:
1- `dice` class
2- `turn` class
3- `game_yahtzee` class

1.1 The `dice` class

This is the simplest class. An object `dice` will have only 2 instance attributes (`value`and `keep`), one function `roll`and two representant/print function to make the **print** easier (those 2 functions are already written). Each `dice` must then know its value (1, 2, 3, 4, 5 or 6) and whether or not the user wants to keep it or reroll it. The `roll` function will "roll" the dice if the user didn't want to keep it.

1.2 The `turn` Class

We define a `turn` as the (up to) 3 rolls of dices. Recall that we have 5 dices. A `turn` object will take care of all the dice rolls and will ask the user for the dices to keep in a method `play_turn`. It will have a few attributes : `round_done` (a boolean to know where the round is done) and `dices` (a list of 5 objects`dice`). It will also have a function `roll_dices` that will roll the 5 dices (using the `roll`function defined in `dice`).

1.3 The `game_yahtzee` Class

A `game_yahtzee` will have a few attributes to store the different categories (1, 2, 3 of a Kind, Full House...) and their corresponding scores. It will have helpers such as a method`valid_yahtzee` (returns whether or not this set of dices is a valid yahtzee), a method `total_score` (will compute the total score), a method `print_scoreboard` (will print the scoreboard), a method `compute_score` (will compute the according score knowing the final dices values and the category the user picked) and finally a method `play` that will take care of every step of the game.

2 Steps (more detailed)

2.1 Lookout

1. Look at the code globally and understand the global architecture
2. Do each step in order. Test your code between each step by using `print`. Otherwise the bug might be very difficul to find !!!
3. The functions definitions (name, parameters) are already there, as well as the class names. You will need to complete them.
4. Ask questions if you don't understand, especially *what* to do.

2.2 The `dice` class

1. Complete the constructor `__init__()`. It has no parameters (except `self`) but will instantiate 2 objects attributes:
   - `value` attribute : an integer variable representing the value of the `dice` (between 1 and 6). It should be instatiated with 0 at the beginning.
   - `keep` attribute : a boolean variable that represents whether or not (`True` or `False`) the user wants to keep this dice in the next roll. It should be instantiated at `False` since all dices are rolled in the first roll.
2. Complete the method `roll()`. It has no parameters. It rolls the dice only if the user don't want to `keep` it. Otherwise, it does nothing to that particular dice.
3. The `__repr__` and `__str__` methods are useful only to `print` the values of the dice. No modification is needed there.
4. Do some tests to make sure that you implemented the right things. You can use the `print` function to see what the values of the dices are.

2.3 The `turn` class

1. Complete the constructor `__init__()`. It has 2 object attributes:
   - `dices` : a list of 5 `dice()`
   - `round_done` : a boolean that represents whether or not the round must finish now (it can happen if the user keeps all the 5 dices before roll #3). Instantiate it to `False`. We will take care of it later.
2. Complete the `roll_dices()` method. This method simply roll the 5 dices using the `roll()` method defined in `dice` class.
3. Start coding the `play()` method. We will start by a simplified version and we will handle the exceptions and extra features after. These are the steps the `play()` method from `turnclass` must accomplish
   (a) Roll the dices using `roll_dices()` method and show the values to the user.

(b) Keep (and update) a variable `roll_counter` (instantiated to 1 after the first roll) that will keep track of how many rolls we have done yet. It should not go over 3 (since you have a maximum number of 3 rolls)

(c) Do a `while` loop (while the `roll_counter` is below something) inside which the user will be prompted to choose which dices to keep.

(d) If the user writes 'q', `return None` right away. That means the user want to end the game.

(e) Otherwise, extract the integers the user wrote in order to update their corresponding `keep` attribute before rolling the dices again.

This `play()` method should return the final dices at the end of the turn.

3 The `game_yahtzee` class

3.1 Constructor `__init__`

Complete the `__init__()` constructor. A `game_yahtzee` object will have a few attributes:

- `categories_dict` : A dictionary containing the different categories (stored as strings) as keys and description (strings) as values. This will be useful to print the scoreboard using the values. Here are the different keys you must use.
  - Key = '1', Value = 'Ones'
  - Key = '2', Value = 'Twos'
  - Key = '3', Value = 'Threes'
  - Key = '4', Value = 'Fours'
  - Key = '5', Value = 'Fives'
  - Key = '6', Value = 'Sixes'
  - Key = '3K', Value = '3 Of A Kind'
  - Key = '4K', Value = '4 Of A Kind'
  - Key = 'FH', Value = 'Full House'
  - Key = 'SS', Value = 'Small Straight'
  - Key = 'LS', Value = 'Long Straight'
  - Key = 'C', Value = 'Chance'
  - Key = 'Y', Value = 'Yahtzee'
  - Key = 'B', Value = 'Bonuses' (useful for bonus points)
- `categories` : A list containing the 14 above categories, stored as strings.
- `available_categories` - A list containing the 13 available categories at the beginning of the game. It must contain all the different categories, except 'B'. It represents the categories the user can pick. We will need to keep that list updated by removing categories the user picks, because a category can only be used once (will be dealt with later).
- `scoreboard` : A dictionary containing the above categories (including 'B') as keys and their corresponding score as values (values should be instantiated to 0).

Make sure you can define a `game_yahtzee` object now.

3.2 The `play` method

This is where most of the game happens. For now, it must include a `for` loop that will loop 13 times (for the 13 available categories). In each of these 13 times, a `turn` object must be initialized. Its `play_turn` method will prompt the user for the dices he/she wants to keep. Store the output of `play_turn` in a variable named `dices`. Those are the dices at the end of the turn.

The next step is to prompt the user to obtain the category he/she wants the dices in. Take for granted (for now) that the user doesn't mispell the category, and we will handle input exceptions later, if time permits it. Once you know the choosen category, you will compute and keep the correponding score in a variable `score`, using the `compute_score` method that we will complete later. Update the `scoreboard` attribute. You can use the `print` function to make sure everything is working well.

3.3 The `compute_score` method

For every category, fill the details to correctly compute the score corresponding to these dices and category. You should start with easier ones and verify your code by playing the game. The `compute_score` take 2 arguments (apart from `self` argument) : the `dices` and the `category` (a string) the user picked. It should return the corresponding score.

3.4 The `print_scoreboard` method.

It takes no argument, and prints the scoreboard. On each line, you can print the name of a category and its corresponding score. You can iterate through the `categories` attributes and access the score using the `scoreboard` attribute.

3.5 The `total_score` method.

It takes no argument, and returns the total score. Use the `scoreboard` attribute to access the scores.

4 Extras

Here are some extras you can do.

1. You should handle wrong inputs from user in the category choice. A category can only be choosen once.
2. Do a next round (a next dice roll) only if the user doesn't want to keep all dices
3. In a `turn`, if the user writes '0', re-roll all the dices.
4. Compute the "upper" bonusat the end of the game. (See Wikipedia page). Always show the upper sum value in the scoreboard.
5. Compute the 2nd, 3rd (and so on) Yahtzee scores. (See Wikipedia page for Free Choice Joker Rule)
6. You should handle wrong inputs from user in the dices choice. If the user enters a wrong combination of dices (letters, characters, invalid number of dices), then you should keep asking for a valid combination, unless the user write 'q' (in that case, the `play` method should return None).
7. You should handle wrong inputs from user in the category choice. You should prompt the user until he gives you a valid category.
8. Make it a 2-players game.