AE1205 Assignment 3: Poker Dice

In this assignment we will make a program to play the game of Poker Dice against the computer. In this game, the goal is to get the highest-scoring combination of five dice, in three rolls per player. The scoring, from highest to lowest is:

1. Five of a kind
2. Four of a kind
3. Full house (3 of a kind + 2 of a kind)
4. Straight (1, 2, 3, 4, 5 or 2, 3, 4, 5, 6)
5. Three of a kind
6. Two pairs
7. One pair
8. Bust (no equal dice and no straight)

An example of how this program would run is given below (user input in red):

```
Welcome to Dice Poker! Beat the computer by getting the highest score.
Round 1:
The computer rolled: 1, 3, 4, 5, 6
and kept: 3, 4, 5, 6
You rolled: 1, 2, 3, 5, 6
Which dice do you want to keep? (choose from dice 1, 2, 3, 4, 5)
Your answer: 2345
Round 2:
The computer rolled: 3, 4, 5, 5, 6
and kept: 5, 5
You rolled: 2, 3, 5, 6, 6
Which dice do you want to keep? (choose from dice 1, 2, 3, 4, 5)
Your answer: 45
The computer rolled: 2, 3, 4, 5, 5
You rolled: 1, 2, 2, 6, 6
The computer has a pair
You have a two pair
Congratulations, you won!
```

Elements of the program

When complete, your program needs to contain the following items:

1. A function to randomly roll between zero and 5 dice, where the number of dice to roll depends on how many dice the player has kept from the previous round.

2. A function to calculate the score corresponding to a combination of five dice. Here, a hand can be one of the eight types described above.
3. The computer 'AI': a function that determines which dice the computer wants to keep with each roll.
4. Code (a loop) to perform the three turns for the two players.
5. Code to compare the scores, and print the outcome.

Below are some hints for each of these elements:

Item 1: rolling the dice

This step involves creating a function that determines how many dice to roll, and then randomly rolls those dice. A possible way to structure this function is to make it accept the dice held from the previous roll as an argument, and return the complete new hand of dice as a result. Randomly rolling dice can be done with the `randint` function from the `random` module. The final function would look something like this:

```python
from random import randint

def roll(hand):
    ndice = ... # 5 minus the number of dice in 'hand'

    # Use the randint function to roll the remaining dice
    # and add them to 'newhand', together with the dice from 'hand'
    ...
    # Return a sorted hand
    return ...
```

Item 2: Calculating the score of a hand

The score of a hand is determined by the best combination of dice you can make with the given five dice. The different scores (five of a kind, ...) were given at the beginning of this assignment. Make a function that determines this score for a given hand, and which also returns the relevant dice values in a tuple. So for example, a five of a kind with all fives returns a `score` of 1, and a tuple of dice values containing only: `values = (5,)`. Another example: four ones and a six, would return a score of `score = 2`, and a tuple of values of values = (1, 6) (i.e., first the value of the 'four of a kind', then the value of the remaining dice).

An easy way to count how many double values there are in a hand is to use python's built-in type `set()`. A set is an unordered list, with duplicate values removed. The final function will look something like this:

```python
def calc_score(hand):
    # Create a sorted list of all unique values in 'hand'
    # If it's sorted it will be easier to generate the 'values' tuple
    unique = sorted(list(set(hand)))

    score = 0
    values = unique

    # Determine the highest-scoring combination in 'hand'.
    # Hint: use the length of 'unique'.
    # In addition, fill 'values' ordered from the largest set contributing
    # to the score, to the lowest.
    # Example: for this hand: 5, 1, 1, 1, 1, 'values' becomes:
    # [1, 5] (i.e., first the value of the four-of-a-kind, then the remaining
     ↪ dice)

    return score, values
```

Item 3: The computer 'AI'

In this game you want to be able to play against the computer. This means that the computer needs to be able to decide which dice to keep after every roll of the dice. You could make this quite complicated, but in this game we will keep it simple: We'll create a computer AI '`select()`' function that keeps all dice that are part of a pair or better. When there is no pair at all it will try to go for a straight. The resulting function will look something like this:

```
def select(hand):
    unique = set(hand)

    if len(unique) == 5:
        # There are no pairs or better, optimise for a straight
        newhand = ...
    else:
        # pick all the pairs or better from hand, and discard the single numbers
        newhand = ...
    return newhand
```

Item 4: The game loop

The game loop should allow each player to roll the dice three times. Recall from step 1 that for each round you can use the roll function to roll the remaining dice by passing it the dice kept from the previous round:

```
hand = roll(previous_hand)
```

After doing this for both players, the computer AI decides what to do with the computer hand, and the program has to ask the human player what to do with his/her hand. You can do this by asking the player to select the dice to keep by indicating their position in the printed sequence (see also the example at the beginning of this assignment):

```
print('Which dice do you want to keep? (choose from dice 1, 2, 3, 4, 5)')
answer = input('Your answer: ')
```

Based on this you can generate the hand of dice to take to the next round. In each game this is done three times, where in the last round no selection needs to be made anymore.

Step 5: Compare scores and print the outcome

In the final step of the game, the scores from the human and computer player should be compared, to be able to print the outcome. In terms of the score there are three possible outcomes:

- Computer score > human score: This one is easy: the human player has the lowest score and therefore wins.
- Computer score < human score: This is also easy: the computer wins in this case.
- Computer score == player score: In this case the game is undecided based on the score alone. This is where the returned values come in. Each of the returned values should be compared in order, and the first non-equal comparison decides who wins.