24 Card Game

February 6, 2018

1 24 Card Game

In this exercise, you will write code to find all possible solutions for a given 4-card combination and print each solution in postfix notation. A combination can be a solution only if it evaluates to 24.

In this game, 4 cards are randomly picked from a standard deck of 52 cards. We will use the values of these 4 cards and a combination of addition, subtraction, multiplication, division, modulus and concatenation to produce the value 24. Note that each of the 4 cards can only be used once.

For the purposes of this assignment, we will consider the following to be the face values of each card in a suit:

$'A'$: 1
$'2'$: 2
$'3'$: 3
$'4'$: 4
$'5'$: 5
$'6'$: 6
$'7'$: 7
$'8'$: 8
$'9'$: 9
$'10'$: 10
$'J'$: 11
$'Q'$: 12
$'K'$: 13

For example, if the 4 cards chosen are "8 8 3 3" we can produce the value 24 by doing the following operation: # $\frac{8}{3-\frac{8}{3}}$

Note that there are many possible solutions for a given 4-card combination. For instance the cards "8 8 3 3" can produce 34 different solutions which produce 24. Read the following link for more information about the game: https://en.wikipedia.org/wiki/24_Game.

2 Postfix Notation

Postfix notation is an unambiguous way of writing an arithmetic expression without parentheses. It is defined so that if "$(exp1)\ op\ (exp2)$" is a normal, fully parenthesized expression whose operation is op, the postfix version of this is "$pexp1\ pexp2\ op$" where pexp1 is the postfix version of exp1 and pexp2 is the postfix version of exp2. The postfix version of a single number or variable is just that number or variable.

For example, the postfix version of "$((6-1) * (7+2)) / 4$" will be equal to "$6\ 1\ -\ 7\ 2\ +\ *\ 4\ /$".

3 Fractions in Python

The very first line in the skeleton code imports the smaller module 'Fraction' from module 'fractions'. A module is simply a file containing Python definitions and statements. We can use the following methods to perform the required arithmetic operation.

```
In [7]: Fraction.__add__
        Fraction.__sub__
        Fraction.__mul__
        Fraction.__truediv__
        Fraction.__mod__
```

4 Evaluating a postfix expression

The evaluate() method will take in a string containing a postfix expression. In this function, as long as the size of the stack is not 0, you will pop the next element from the stack. If the element is an operation (i.e. is either '+', '-', '*', '/', '%' or 'c') you will pop the next two elements from the stack, which will always be card values. Try playing around with different postfix notations to see why this is true. Evaluate the two numbers using the operation. If the operation is division and the denominator is 0, you can simply return None to the function and terminate the evaluation, because this means that it will not evaluate to 24 anyway. You must also return None if the expression isn't a valid postfix expression. If the element is a card value, then just push the value back onto the stack. Once the function finishes running, the final remaining element on the stack will be the result of evaluating the postfix expression.

5 Checking if a postfix expression is valid or not

In the isvalid() method, you must call evaluate method for the postfix expression to check if it returns None or if it returns a value. This method returns a boolean value.

6 To do:

1. Complete the evaluate() method and make sure to account for division by zero and invalid postfix expressions.
2. Complete the isvalid() method to check if a postfix expession is valid or not.