Lab 6

Instructions:

Provide your solutions in a file named **lab6.py**. Make sure you run the doctest before submitting. Note that you should also test the Scrambler game thoroughly.

Problems:

1. Write a class `ScoreList` that can be used to record a number of course scores (from 0 through 100, inclusive of both) and report information about them. :

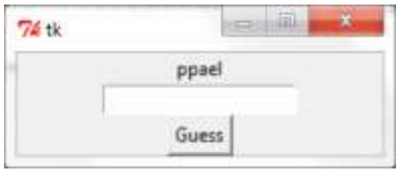
```
    >>> s = ScoreList()
    >>> s.add(57)
    >>> s.passing()
    False
    >>> s.add(83)
    >>> s.add(80.2)
    >>> s.add(104)  #doctest: +IGNORE_EXCEPTION_DETAIL
    Traceback (most recent call last):
    ...
    ScoreError: score 104 not between 0 and 100
    >>> s
    ScoreList([57, 83, 80.2])
    >>> s.avg()
    73.39999999999999
    >>> s.passing()
    True
    >>> max( s )
    83
    >>> min( s )
    57
    >>> len( s )
    3
    >>> s[1]
    83
    >>> s.sort()
    >>> s
    ScoreList([57, 80.2, 83])
    >>>

    Parameterized constructor

    >>> s = ScoreList( [80,30,100] )  # this is ok
    >>> s == eval(repr(s))  # test, init, repr, ==
    True
    >>> s = ScoreList( [80,-10,100])
    Traceback (most recent call last):
    ...
    ScoreError: score -10 not between 0 and 100
```

```
>>> s = ScoreList( [80,10,110])
Traceback (most recent call last):
...
ScoreError: score 110 not between 0 and 100
>>>
```

Implementation details:

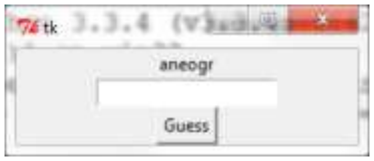
- `ScoreList` should inherit from `list` (note that this automatically provides some of the above functionality)
- `constructor` – constructs either an empty list, or initializes with a list of values (note that this may cause a ScoreError if some score is out of range).
- `__repr__` - to produce the output above
- `add()`
    - accepts one number
    - if the provided number is at least 0 and at most 100, it is added to the ScoreList
    - otherwise, an Exception is raised (see output above for message)
- `avg()`
    - takes no arguments
    - returns the average of the scores on the list
- `passing()`
    - takes no arguments
    - returns `True` if the average is at least 60, `False` otherwise

1. Write a `tkinter` game `Scrambler`, that allows the user to guess a scrambled word. The game is started by giving a word and packing. The parent can be specified or omitted, e.g, either:



```
>>> Scrambler("apple").pack()
>>>
```

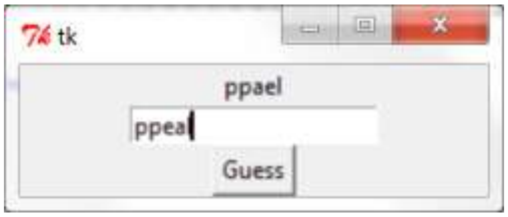
Or



```
>>> root = Tk()
>>> Scrambler("orange",root).pack()
>>>
```

The game will then scramble the word and display it in the gui, see images to the right. The user will then have 3 guesses to guess the original word. (To scramble a word, make sure you add the import "`from random import *`", then use the following code.

```
scramble = list( word )
shuffle(scramble)
scramble = ''.join( scramble)  # this is now the scrambled word
```



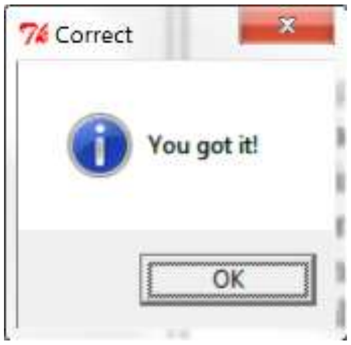
tk

ppael

ppeal

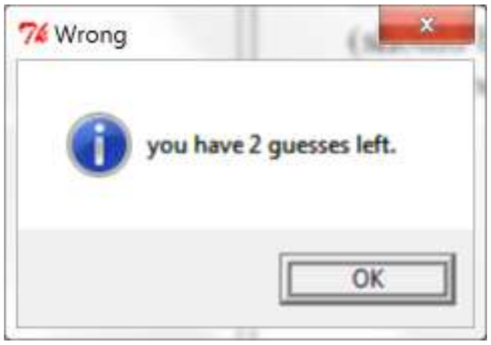
To guess a word, the user types in the Entry and clicks the button. The game then checks the guess against the original. There are three possibilities:

1) The guess is correct, then the user is told "You got it" in a showinfo.
2) The guess is incorrect, and there is at least 1 guess left. Then the user is told the number of guesses left.
3) The guess is incorrect, and no guesses are left. The user is told "You lose"



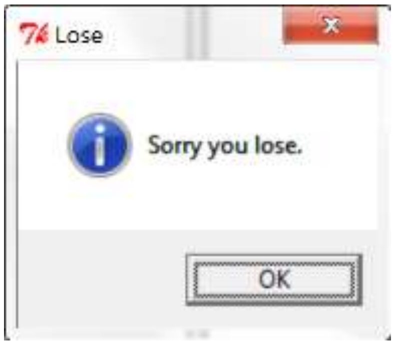
Correct

You got it!



Wrong

you have 2 guesses left.



Lose

Sorry you lose.