```
%matplotlib inline
import matplotlib
import seaborn as sns
sns.set()
matplotlib.rcParams['figure.dpi'] = 144
```

```
%matplotlib inline
import matplotlib
import seaborn as sns
matplotlib.rcParams['savefig.dpi'] = 144
```

```
from static_grader import grader
```

IN Miniproject

The objective of this miniproject to teach you about our grading system. We will do this by solving two easy problems.

Our grader is a custom bit of code that needs to imported as above in order to work. Questions are scored by calling `grader.score.question_name(answer)`. The `question_name` method be different for each question, and will be provided for you. The form of `answer` will vary question to question. Sometimes you will be asked to submit a fixed value; other times you will be asked to submit a function which will compute something based on some (hidden) input.

Your answer will be graded automatically and the score returned to you on a 0 to 1 scale. (Sometimes you can do even better than 1!) We will count your highest grade, so feel free to try out different solutions!

Problem 1

Compute the five smallest positive even numbers. Provide these in a list.

In this case, the answer is a fixed value, so you may compute (or hard-code) the answer and submit it. Note that we have provided a "dummy solution" below. This dummy solution illustrates the correct format of the solution (i.e. a list of 5 numbers). If you are encountering an error when you try to submit a solution to the grader, double check that your answer has the same structure as the dummy solution.

```
even_numbers = [2*i for i in range(0, 5)]
```

```
grader.score.in__problem1(even_numbers)
```

```
==================
Your score:  0.8
==================
```

Problem 2

Define a function that accepts a list (called `numbers` below) as input and return a list where each element is multiplied by 10. The grader will supply the argument `numbers` to the function when you run the `grader.score.in__problem2` method below.

In this case, you need to write a function that will work for arbitrary input. Before submitting your function to the grader, you may want to check that it returns the output that you expect by evaluating code similar to the following:

```
test_numbers = [1, 2, 3]
mult(test_numbers)
```

```
#Sol1 - Traditional
def mult(numbers):
    for i in range(len(numbers)):
        numbers[i] = numbers[i]*10
    return numbers

#Sol2 - Use list comprehension and lambda function
#def mult(numbers):
#    mult_numbers = [x=numbers[i]*10 for i in range(len(numbers))] !
incorrect syntax
#    return mult_numbers
```

```
test_numbers = [1, 2, 3]
mult(test_numbers)
```

```
[10, 20, 30]
```

```
grader.score.in__problem2(mult)
```

```
==================
Your score:  1.0
==================
```