Assignment 3: Sept 7 - Sept 13 (Basic Python 2)

Hand-in deadline: Sept 18

Important: The scripts of this assignment should be saved in a subdirectory named `week3` of your INF3331 github repository.

3.1 String class (2 points)

Define a class `SimpleString` which has at least two class methods:

- `getString`: to get a string from console input
- `printString`: to print the string in upper case

Make use of the `raw_input()` function to input a string. Document both functions with docstrings such that `pydoc simplestring` gives a good idea of what `SimpleString` does.

Example usage:

```
ipython>  from simplestring import SimpleString
ipython>  s = SimpleString()
ipython>  s.getString()
Enter String value:  Hallo world!
ipython>  s.printString()
HALLO WORLD!
```

Name of Python file: simplestring.py

3.2 Circle calculation (2 points)

Define a class named `Circle` which is constructed just by a radius.

The `Circle` class has two other class methods:

- `area`: returns computed area of the circle
- `perimeter`: returns computed perimeter of the circle

Example usage:

```
>> c = Circle(radius=10)
>> print "Area = {0}, Perimeter: {1}".format(c.area(), c.perimeter())
```

Name of Python file: circle.py

3.3 Flexible circle calculation (3 points)

Implement a class named `FlexCircle` which is constructed just by a radius.

The `FlexCircle` class should have three attributes: `radius`, `perimeter` and `area`. The user should be able to set or get any of these attributes. If one of the attributes is set, all other attributes must be updated accordingly. If an invalid value is given for an attribute, a useful error message should be printed.

Example usage:

```
>> c = FlexCircle(radius=2)
>> print c.area       # print area of circle with radius 2
>> c.perimeter = 1.5  # set perimeter of circle to 1.5
>> print c.radius     # print radius for circle with a perimeter of 1.5
>> c.area = 0.6       # set area of circle to 0.6
>> print c.perimeter  # print perimeter of circle with 0.6 area
```

Hint: Make use Python properties.

Name of Python file: flexcircle.py

3.4 Make a specialized ranking function (3 points)

For this assignment you need to download the `data.log` file here.

Suppose we have a script that performs numerous efficiency tests. The output file from the script is called `data.log` and contains lots of information, but our purpose now is to extract information about the CPU time (given in seconds) and compute the minimum, average and maximum run times. The time statistics should be computed independently for each test. The `data.log` file takes the following form:

```
[...]
Running test Heart1
Test Finished
Name: Heart1      CPU: 301.607774687

Running test Heart1
Test Finished
Name: Heart1      CPU: 275.909681963

Running test Vessel_top
Test Finished
Name: Vessel_top        CPU: 262.656994654
[...]
```

First we need to extract the lines with the test name and the CPU-time. From these lines, we need to extract the test name as a string and the CPU time as a float. Then we collect all CPU times for each test separately, and compute the maximum, the minimum and the average, and print them out.

Hint: You can use a `if "CPU" in line` test: to check if the CPU keyword is in a line. Then use the `split()` method to convert the line into a list of words. Create a dictionary to store the CPU times for each test. The dictionary has as key the test name, and as value a list with all CPU times of that test.

The output of our ranking program should look like this:

```
>>> python ranking.py

Test name: Heart2
CPU time:  160.3 s (min)
           191.5 s (avg)
           207.7 s (max)

Test name: Vessell_top
CPU time:  2092.2 s (min)
           2159.8 s (avg)
           2215.1 s (max)
[...]
```

Name of scriptfile: ranking.py

Total points: 10