Python-exercises
Object-oriented programming

April 11, 2012

1 String

A simple class

1. Write a class with one member variable called `mystring` and initialize it in the constructor.
2. Add a class (static) variable called `counter` and initialize it with 0.
3. Add a method which is given another string `other` as argument. The purpose of this method shall be to test if `other` is contained in `mystring`. *Hint:* Try the `in` keyword.
4. Change the initializer such that it increments counter by one.
5. Create several instances of this String class and check afterwards how many you have.
6. Add a 'destructor' function `__del__` which decreases `counter` by one. Test it!

2 Animal

Inheritance

1. Write an `Animal` base class. Each animal shall have an age and a weight. Once set in the constructor these variables should not be changeable from the outside (you cannot just change the age of an animal as you like, right?). But provide functions that allow to read the values of these variables.
   Note: In Python "private" attributes are not really private. But it is the best protection of member attributes against the outside that we can get — so we take it.

2. Implement a `speak` and a `move` method which print an error message (an abstract animal can neither speak nor move).

3. Write a `Cat` and a `Fish` class which inherit from `Animal`. Override the `speak` and `move` methods such that they print out an appropriate message when called (something like "Meow" for the cat...)

4. Test your implementation in the python interpreter by creating instances of `Cat` and `Fish`.

5. Try to make sense of what's happening here.

```
> c=Cat(2,3.7)
> c.speak()
Meow.
> Animal.speak(c)
I'm an abstract animal and cannot speak.
> num = 3.7
> Animal.speak(num)
[some peculiar Python-error message]
```

3 Neuron

A more complex example: using what we've learned so far
The equation determining the membrane potential of a *leaky integrate and fire neuron* is given by

$$c_{\mathrm{M}}\dot{V} = -g_{\mathrm{L}}\left(V - V_{\mathrm{L}}\right) + i_{\mathrm{ext}}$$

where c_{M} is the membrane potential, g_{L} the leak conductance, V_{L} the corresponding reversal potential and i_{ext} an external current that drives the neuron.

In addition, whenever V becomes larger than a threshold value V_{th} a spike is elicited and V is reset to a value V_{r}.

1. Write a class which is initialized with the necessary parameters (like $c_{\mathrm{M}}, \ldots$, don't forget an initial value for V). Except for the external current it should not be possible to change the neuron parameters after instanciation.
   Cf. the hint in exercise 2.1.

2. Use
   $$\dot{V} \approx \frac{V(t+h) - V(t)}{h}$$
   to derive an update rule $V(t+h) = \ldots$.

3. Implement a method which updates V according to this rule until a time T has passed. In each step append the newly calculated V to a list `trace`.

4. Now consider that V is reset everytime it crosses the threshold.

5. *For Friday:* Add a method that uses "matplotlib" to plot the voltage trace.

6. Test your neuron for different initial values for V and different external currents i_{ext}.

7. *If you are quick:* Simulate a network of neurons. Each neuron is connected to some others (e.g. with a predefined probability). The equation to simulate is now
   $$c_{\mathrm{M}}\dot{V}_j = -g_{\mathrm{L}}\left(V_j - V_{\mathrm{L}}\right) + i_{\mathrm{ext}} + i_{\mathrm{j,syn}}$$
   where
   $$i_{\mathrm{j,syn}} = \sum_{t_j^{(k)}} w_{j,k} K(t - t_j^{(k)})$$
   and $K(t) = \delta_{t,0}$. $w_{j,k}$ are weight factors and $t_j^{(k)}$ is the time when neuron j receives its k-th spike.

4 Zoo

Special methods
We want to collect some animals in a `Zoo` object. It will consist of a number of animals and each of them should have a name. Provide a way to add new animals to the zoo. Then implement some special methods like

- the length-operator `__len__(self)`, to see how many inhabitants the zoo has. Test with `len(myzoo)` (where `myzoo` is an instance of `Zoo`)
- the comparison operator `__cmp__(self,rhs)` which should return a negative integer if `self<rhs`, zero if `self==rhs`, or a positive integer if `self>rhs`. Test with `myzoo1<myzoo2` ...
- the subscripting operator `__getitem__(self,name)` to access an animal from the zoo by name (e.g. `z[''blub''].speak()`),
- ...

5 Temperature

Write a class that stores a temperature in one unit and allows accessing it in several other ones (cf. `http://en.wikipedia.org/wiki/Conversion_of_units_of_temperature` for conversion formulas).
Hint: Have a look at `__getattr__`, `__setattr__` to access and set temperatures.