Tutorial on Classes and Objects

Instructor: Salvador Dura-Bernal
Classes are useful to encapsulate variables and functions:
- Class variables = attributes
- Class functions = methods

Objects are instances of a class

Class Animal
Attributes:
- size, sound
Methods:
talk()
class Animal():
    def __init__(self, size, sound):
        self.size = size
        self.sound = sound

    def talk(self, length):
        print(self.sound * length)

cat = Animal(size='small', sound='meow')
dog = Animal(size='small', sound='woof')
cow = Animal(size='big', sound='mooo')

cat.size
'small'
cow.size
'big'
dog.talk(3)
'woofwoofwoof'
cat.talk(10)
'meowmeowmeowmeowmeowmeowmeowmeowmeowmeow'

- Constructor method `__init__()` initializes object attributes.
- Methods must have explicit object reference (`self`) as the first parameter.
- Attribute names are common to all objects but have different values for each one.
- Method is shared by all objects, but produces different outputs.
- Method can have arguments.
class Contact(object):
    """A given person for my database of friends."""

    def __init__(self, first_name=None, last_name=None, email=None, phone=None):
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.phone = phone

    def print_info(self):
        """Print all of the information of this contact."""
        my_str = "Contact info:
        if self.first_name:
            my_str += "  " + self.first_name
        if self.last_name:
            my_str += "  " + self.last_name
        if self.email:
            my_str += "  " + self.email
        if self.phone:
            my_str += "  " + self.phone
        print my_str

bob = Contact('Bob', 'Smith')
joe = Contact(email='someone@somewhere.com')
Encapsulate properties (eg. diam) and methods (eg. recording) of a neuron so we can create many of them easily.

```python
from neuron import h, gui
from matplotlib import pyplot

soma = h.Section(name='soma')
dend = h.Section(name='dend')
dend.connect(soma(1))
h.topology()

# Surface area of cylinder is 2*pi*r*h (sealed ends are implicit).
soma.L = soma.diam = 12.6157  # Makes a soma of 500 microns squared.
dend.L = 400  # microns
dend.diam = 1  # microns

for sec in h.allsec():
    sec.Ra = 100  # Axial resistance in Ohm * cm
    sec.cm = 1   # Membrane capacitance in micro Farads / cm^2

# Insert active Hodgkin-Huxley current in the soma
soma.insert('hh')
soma.gnabar_hh = 0.12  # Sodium conductance in S/cm2
soma.gkbar_hh = 0.036  # Potassium conductance in S/cm2
soma.gl_hh = 0.003    # Leak conductance in S/cm2
soma.el_hh = -54.3    # Reversal potential in mV

# Insert passive current in the dendrite
dend.insert('pas')
dend.g_pas = 0.001   # Passive conductance in S/cm2
dend.e_pas = -65     # Leak reversal potential mV

dend.nseg = 10

# Change the maximum sodium conductance of the middle segment of the soma to 0.13
soma(0.5).hh.gnabar = 0.13

# Change the equilibrium potential of the passive mechanism in the middle segment of the dend to -65
dend(0.5).pas.e = -65
```
Creating an HH cell class

The aim is to put inside a cell class (check code for lab2) all the operations required to create, define and stimulate a cell (check code for lab3 and lab4). Follow these steps

1) Create a class called HHCell with an empty constructor (for now).
2) Add a method create_sections that creates the soma and dend sections. Make sure these sections are an attribute of the class (use self !)
3) Add a method build_topology that connects the dendrite to the soma (at location 1)
4) Add a method define_biophysics that sets the biophysical of the soma and dendrite: axial resistance and membrane capacitance; adds active HH channels to the soma, and passive channels to the dendrite (use same params as in lab 3).
5) Add a method add_current_stim to apply a current clamp stimulation to the dendrite (location 1.0), with amplitude 0.3 nA, duration of 1 ms, and delay of 20 ms. Make sure the current clamp object is an attribute of the class.
6) Add a method record_voltage to record the voltage from the soma and dend (at location 0.5).
7) Add a method plot_voltage to plot the voltage at the soma and dendrite (location 0.5), in black and red color.
8) Call the create_sections, build_topology, define_geometry, and define_biophysics methods from the class constructor.
Creating an HH cell class

1) Create a cell object called `cell1` of class `HHCell`.
2) Add current clamp stimulation to the `cell1` object.
3) Set up the voltage recording of the `cell1` object.
4) Set the simulation duration to 60 mV, and run the simulation.
5) Plot the voltages of the `cell1` object.
Cell Class

HHCell

Attributes:
• soma
• dend
• stim
• soma_v_vec
• dend_v_vec
• t_vec

Methods:
• __init__()
• create_sections()
• build_topology()
• define_geometry()
• define_biophysics()
• add_current_stim()
• set_recording()
• plot_voltage()