Using Python to run NEURON
Everything is the same (but different)

- All underlying simulation objects must be same
- All underlying simulation calls must be same
- Various useful and important ancillary things are available
- Most of these (e.g., vectors, lists) map 1-to-1 on native Python objects
Compiling –with-nrnpython

- many configuration options

- –with-nrnpython Python interpreter can be used (default is NO) Probably need to set PYLIBDIR to find libpython and PYINCDIR to find Python.h

- –with-paranrn
What environment?

- nrniv -python
- python: from neuron import h
- python: from neuron import gui
- python vs ipython
- sage
Python advantages

- Widely used
- Readily extensible
- Many plugins/toolboxes
- Easy yet hard
- Easy arg calls
- Elegant (maybe too much so)
Advantages for NEURON

- All legacy models work
- Better representation of concepts
- Uniform approach to objects
- Connect to other tools: numpy, scipy, ...
Helpful: `dir(soma)`, `help(soma)`

- `h` is a function or object
- `h()` executes arbitrary hoc
- `h.thing` then accesses `thing`
An example

```python
>>> h('''
x = 5
... strdef s
... s = "hello"
... func square() { return $1*$1 }
''')

now access with `h`.

print h.x, h.s # print is a python command

h("print x,s // print is a hoc command")

x=h.square(10) # x is a python variable

Looks silly (and confusing) but needed for legacy
Making new things

- 'create soma' → soma=h.Section()
- 'stim=new IClamp()' → stim=h.IClamp()
- *soma insert hh* → soma.insert('hh')
Note potential for massive screw-ups

2 languages → can make 2 different things with same name

soma = h.Section(); soma.L = 20; h('create soma')

print soma.L, h.soma.L  # different

To avoid this: h('create soma'); soma = h.soma

Important for shape plots etc.
Accessing segments

- soma for (x) gnabar_hh(x)=3e-3*x →
- for seg in soma: seg.gnabar_hh=3e-3*seg.x;
- Can also still access segments relatively:
- soma print gnabar_hh(0.5) →
- soma(0.5).hh.gnabar
More placement: insert vs PPs

- soma stim = new IClamp(0.5)
- stim = h.IClamp(soma(0.5))
- Setting is the same:
- stim.amp=0.1; stim.dur=10; stim.delay=100
in hoc:

begintemplate Cell
proc init() {
  topology()
  subsets()
  ...
}
...
endtemplate Cell
in python:

class Cell(object):
    def __init__(self):
        self.topology()
        self.subsets()
    ...

Template (hoc) \rightarrow class (py)
def topology(self):
    self.soma = h.Section(cell = self)
    self.dend = h.Section(cell = self)
    self.dend.connect(self.soma)
Some things are hard to find

- how to find optional arguments?
- inspect.isbuiltin(h.Section) # true
- inspect.getmembers(h.Section())
- src/nrnpython/nrnpy_nrn.cpp