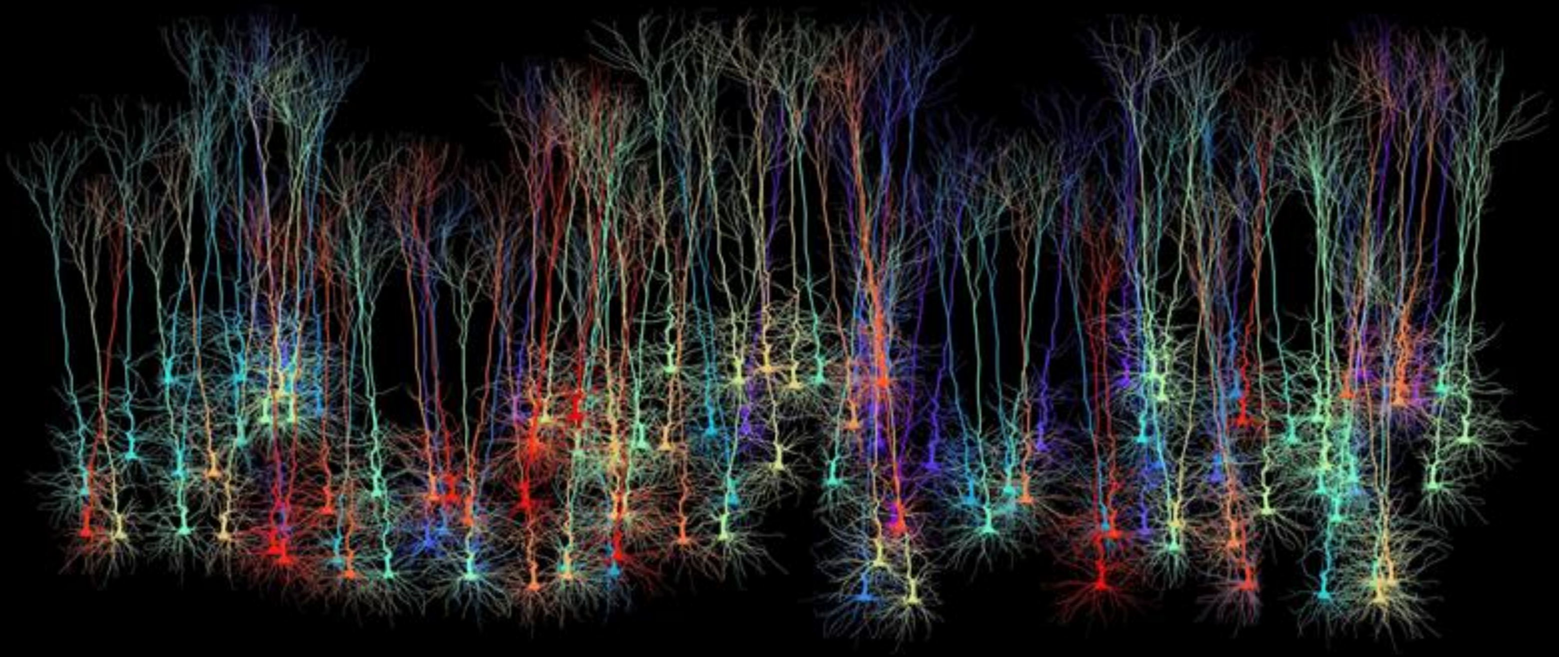
Python and NEURON



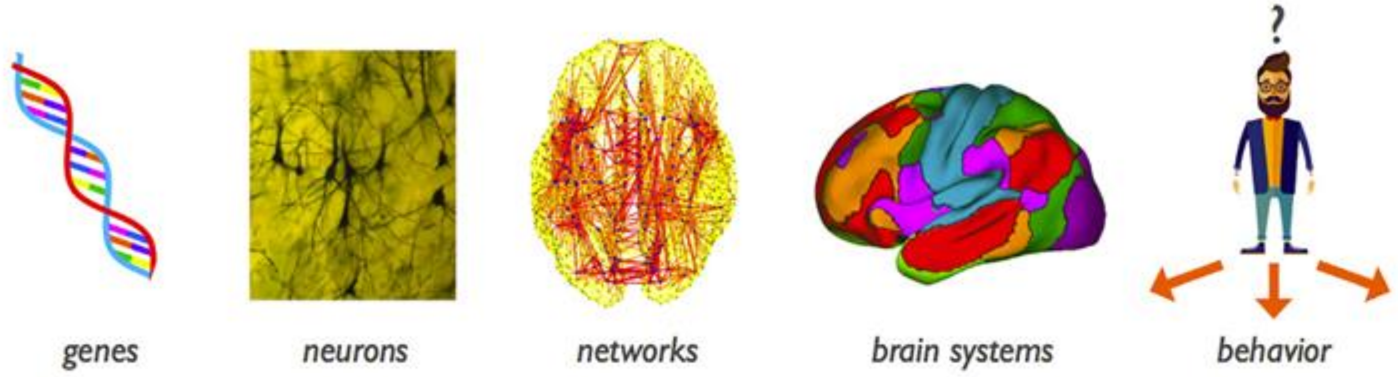
Torbjørn V Ness

- Python has become the default programming language in computational neuroscience
    - Many important tools can be used through python
- Efficient for humans
    - Very important
- Inefficient for computers
    - Less important
- Lot of useful python packages:
    - Matplotlib, numpy, pandas, scipy

- Plenty of tutorials available!

The NEURON simulator

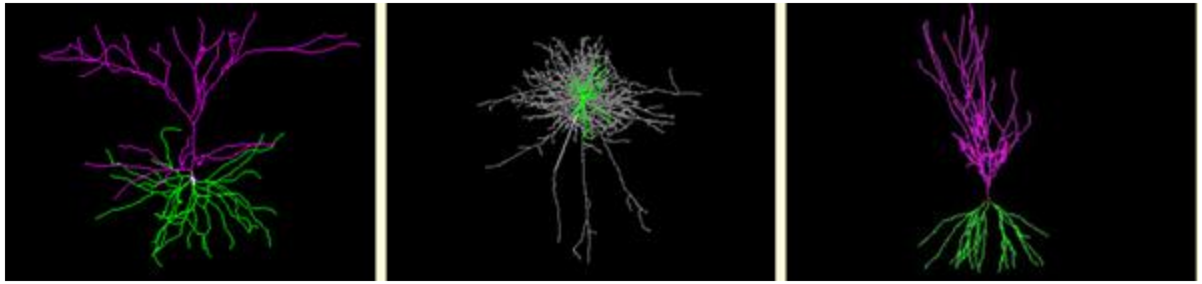
- The brain is studied at different levels
- NEURON is for simulating single cells and “small” neural networks



van den Heuvel & Yeo (2017) Neuron

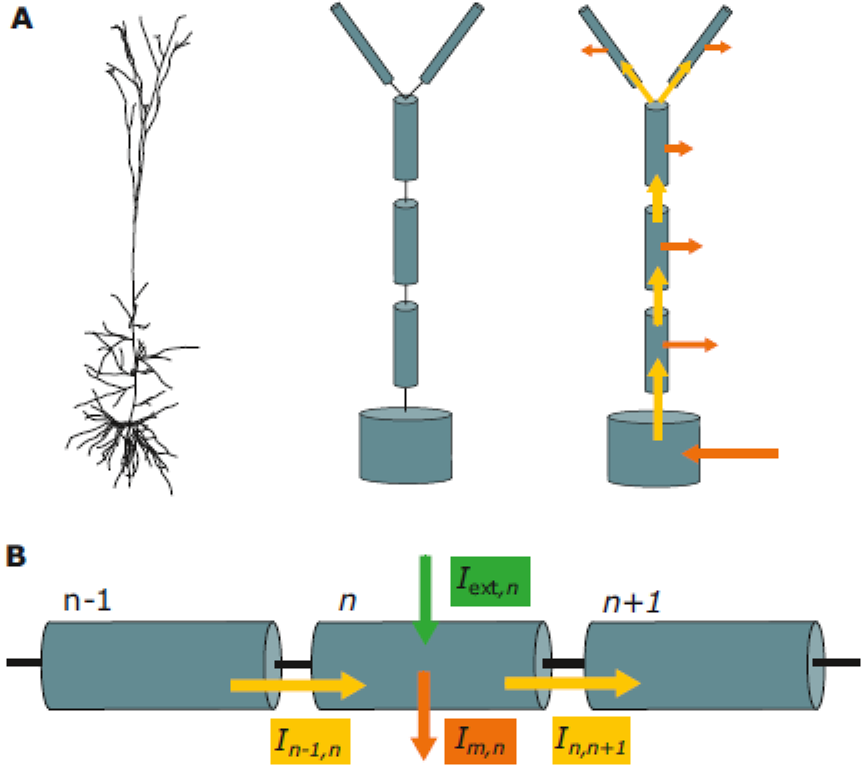
The NEURON simulator

- Most commonly used simulator for biologically detailed cell models
- From point neurons to very detailed cell models
- From single cells to large networks
- For larger networks: NEST etc.



The NEURON simulator

- Multi-compartmental modelling



Halnes et al. (2024) *Electric Brain Signals*

Cable Equation

Current conservation (Kirchhoff's 1st law):

$$i_{a,n+1} - i_{a,n-1} = \frac{d\,i_a}{dx} = -i_m \quad (1)$$

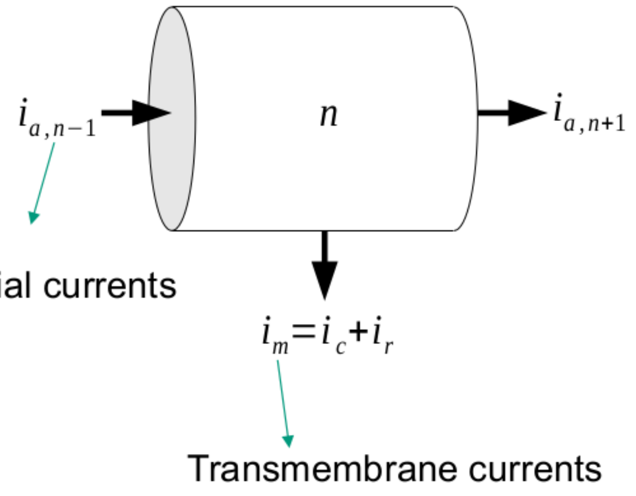
Ohm's law:

$$V_n - V_{n-1} = \frac{dV}{dx} = -r_a i_a \qquad (2)$$

$$i_r = \frac{V_n}{r_m}$$

Capacitive currents:

$$i_c = c_m \frac{dV}{dt}$$

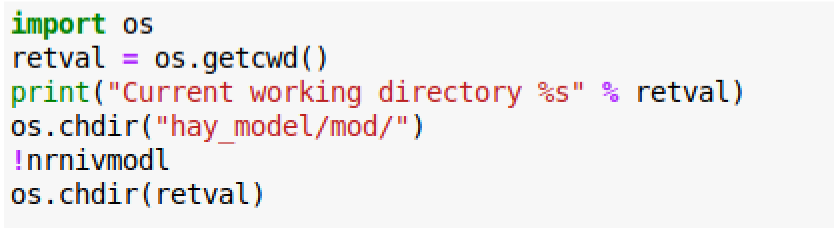


i_a from (2) into (1)

$$\boxed{\frac{1}{r_a}\frac{d^2V}{dx^2} = c_m \frac{dV}{dt} + \frac{V}{r_m}}$$

The NEURON simulator

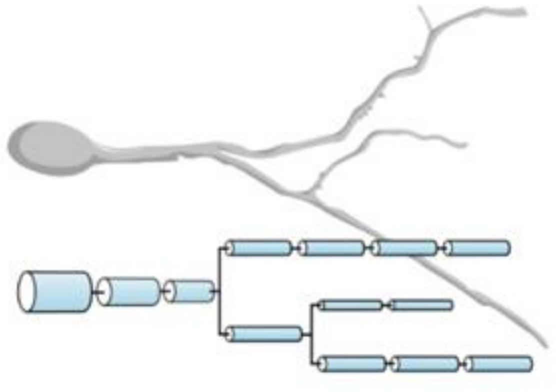
- HOC was the original programming language supported by NEURON
  - I recommend that you stick with Python when you can
- Ionic mechanisms (mod-files)
  - In a terminal at the location of the “*.mod” files, write “nrnivmodl”



```
import os
retval = os.getcwd()
print("Current working directory %s" % retval)
os.chdir("hay_model/mod/")
!nrnivmodl
os.chdir(retval)
```

The NEURON simulator

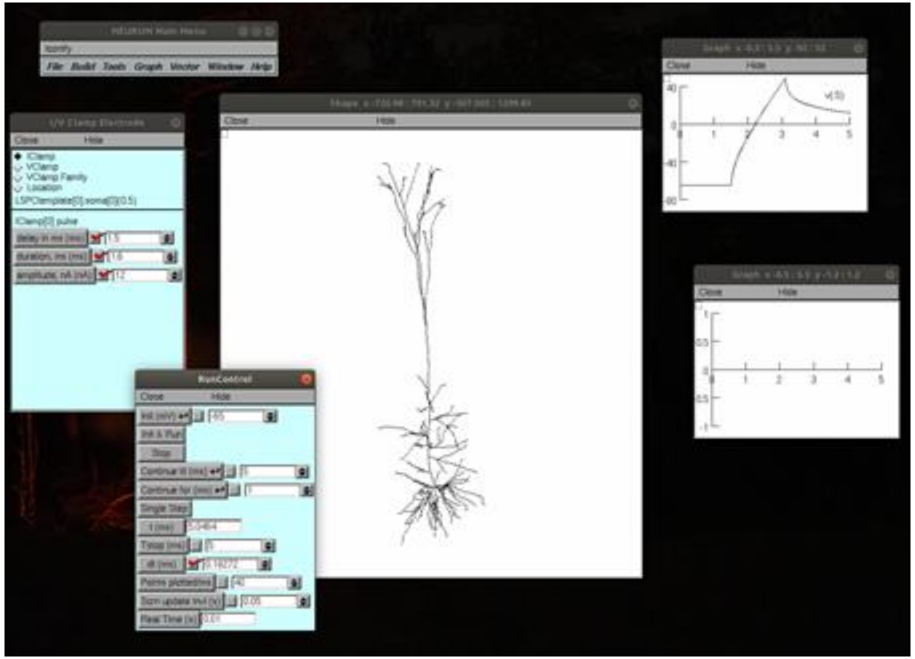
- Cells are represented are electrically connected cylinders
- Cells are divided into different non-branching **sections** (soma, dendrite, axon)
- Sections are subdivided into **segments**
- To access a part of the section, specify a value between 0 and 1, where 0 is typically the end closest to the soma and 1 is the distal end



Sterrat et al. (2011)

The NEURON simulator

- The NEURON simulator has been developed over more than 30 years
  - Developed by scientists, not programmers
  - Open source (, but not that open source)
  - Helpful developers



```
[c]#if VT125
case VT:
vtplot(mode, x, y);
break;
#endif[/c]
```

Now, I didnâ€™t know what VT125 was when I first saw this, but a quick search on the web reminded me that Iâ€™m still a young software developer. I present to you, the VT100:

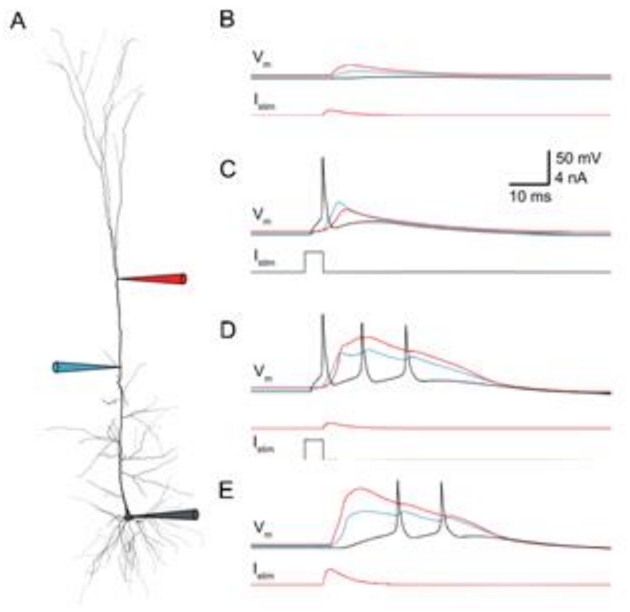
[caption id=â€attachment_1367â€ align=â€aligncenterâ€ width=â€640â€] I couldnâ€™t find a good picture of the VT125. I apologize if it looks way more modern than its older cousin.[/caption]

Svenn-Arne Dragly

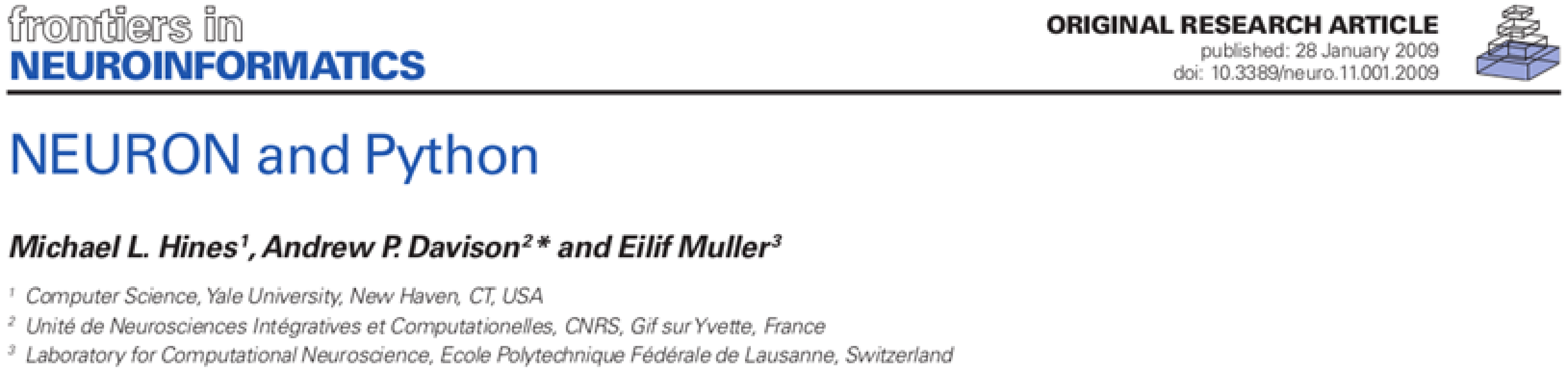
https://dragly.org/2015/10/26/straight-from-the-source-neurons-incredible-backwards-compatibility/

Online databases

- ModelDB
  - https://modeldb.science/
  - Large database of detailed cell models
  - Fully-developed cell models
- NeuroMorpho.org
  - ~268’000 reconstructed cell models
  - Only morphologies
- The Neocortical Microcircuit Collaboration Portal
  - https://bbp.epfl.ch/nmc-portal/microcircuit
  - A lot of detail about rat somatosensory barrel cortex
- Allen Brain Atlas
  - https://celltypes.brain-map.org/
  - Large database of mice and human cell models



Hay et al (2011) PLoS Comput Biol 7:1–18.



frontiers in
NEUROINFORMATICS

ORIGINAL RESEARCH ARTICLE
published: 28 January 2009
doi: 10.3389/neuro.11.001.2009

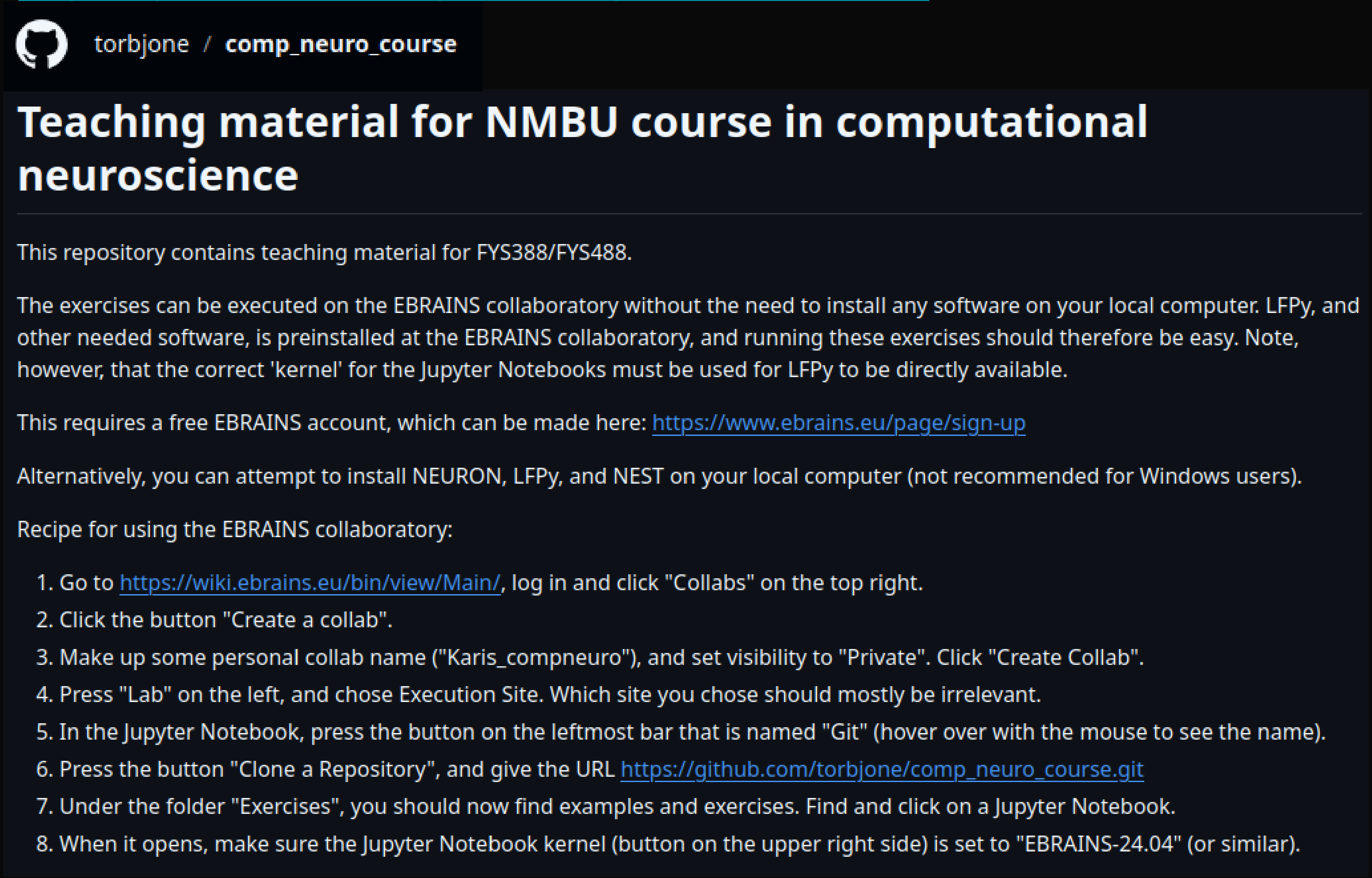
NEURON and Python

Michael L. Hines[1], Andrew P. Davison[2]* and Eilif Muller[3]

[1] Computer Science, Yale University, New Haven, CT, USA
[2] Unité de Neurosciences Intégratives et Computationelles, CNRS, Gif sur Yvette, France
[3] Laboratory for Computational Neuroscience, Ecole Polytechnique Fédérale de Lausanne, Switzerland

https://nrn.readthedocs.io/

https://github.com/torbjone/comp_neuro_course/



torbjone / **comp_neuro_course**

Teaching material for NMBU course in computational neuroscience

This repository contains teaching material for FYS388/FYS488.

The exercises can be executed on the EBRAINS collaboratory without the need to install any software on your local computer. LFPy, and other needed software, is preinstalled at the EBRAINS collaboratory, and running these exercises should therefore be easy. Note, however, that the correct 'kernel' for the Jupyter Notebooks must be used for LFPy to be directly available.

This requires a free EBRAINS account, which can be made here: https://www.ebrains.eu/page/sign-up

Alternatively, you can attempt to install NEURON, LFPy, and NEST on your local computer (not recommended for Windows users).

Recipe for using the EBRAINS collaboratory:

1. Go to https://wiki.ebrains.eu/bin/view/Main/, log in and click "Collabs" on the top right.
2. Click the button "Create a collab".
3. Make up some personal collab name ("Karis_compneuro"), and set visibility to "Private". Click "Create Collab".
4. Press "Lab" on the left, and chose Execution Site. Which site you chose should mostly be irrelevant.
5. In the Jupyter Notebook, press the button on the leftmost bar that is named "Git" (hover over with the mouse to see the name).
6. Press the button "Clone a Repository", and give the URL https://github.com/torbjone/comp_neuro_course.git
7. Under the folder "Exercises", you should now find examples and exercises. Find and click on a Jupyter Notebook.
8. When it opens, make sure the Jupyter Notebook kernel (button on the upper right side) is set to "EBRAINS-24.04" (or similar).