# A bit on Python

Dirk Haupt

# If programming languages were cars...

MATLAB is what scientists use to do special scientist things.

Python is great for everyday tasks: easy to drive, versatile, comes with all the conveniences built in. It isn't fast or sexy, but neither are your errands

http://crashworks.org/if_programming_languages_were_vehicles/

# Top-tier companies using Python

- ***Google*** makes extensive use of Python in its web search systems.
- The popular ***YouTube*** video sharing service is largely written in Python.
- The ***Dropbox*** storage service codes both its server and desktop client software primarily in Python.
- The ***Raspberry Pi*** single-board computer promotes Python as its educational language.
- ***EVE Online***, a massively multiplayer online game (MMOG) by CCP Games, uses Python broadly.
- The widespread ***BitTorrent*** peer-to-peer file sharing system began its life as a Python program.
- *Industrial Light & Magic*, ***Pixar***, and others use Python in the production of animated movies.
- ***ESRI*** uses Python as an end-user customization tool for its popular GIS mapping products.
- Google's ***App Engine*** web development framework uses Python as an application language.
- The ***IronPort*** email server product uses more than 1 million lines of Python code to do its job.
- ***Maya***, a powerful integrated 3D modeling and animation system, provides a Python scripting API.
- The ***NSA*** uses Python for cryptography and intelligence analysis.
- ***iRobot*** uses Python to develop commercial and military robotic devices.
- The ***Civilization IV*** game's customizable scripted events are written entirely in Python.
- The One Laptop Per Child (*OLPC*) project built its user interface and activity model in Python.
- ***Netflix*** and ***Yelp*** have both documented the role of Python in their software infrastructures.
- ***Intel***, ***Cisco***, ***Hewlett-Packard***, ***Seagate***, ***Qualcomm***, and ***IBM*** use Python for hardware testing.
- ***JPMorgan Chase***, ***UBS***, ***Getco***, and ***Citadel*** apply Python to financial market forecasting.
- ***NASA***, ***Los Alamos***, ***Fermilab***, ***JPL***, and others use Python for scientific programming tasks.

# Top-tier companies using Matlab

# Some Python tips

- Use os. Use it.
- memory mapping can make working with big files a dream
- try/catch/else. Control flow with errors.
- list comprehensions
- Use Gohike’s site
- pyqt and pyinstaller
- Pyqtgraph
- qsettings

# Always use **os.path** for your path manipulations

```
self.data_file_path1 = "/media/Cage1/TextFiles/file1.tiff"
...
self.data_file_path100 = "/media/Cage3/Video//Works/file100.mat"
```

This can get really annoying really fast…

Os.path = *import os*

- os.path.**abspath**(*"folder/textfile.txt"*)
- os.path.**basename**("/media/Cage1/TextFiles/file1.tiff")
- os.path.**exists**("/media/Cage1/TextFiles/file1.tiff")
- os.path.**dirname**("/media/Cage1/TextFiles/file1.tiff")
- os.path.**join**("/media/Cage1/TextFiles/file1.tiff", "\folder1/folder/2\\", "'text.txt'")
- os.path.**splitext**("/media/Cage3/Video//Works/file100.mat")
- `os.makedirs('/tmp')`

# Numpy. Always. Here's one reason why.

np.save('/tmp/123', np.random.randn(1000, 256,256))
loaded_from_file = `np.load('/tmp/123.npy')`

A memory-mapped array is kept on disk. However, it can be accessed and sliced like any ndarray. Memory mapping is especially useful for accessing small fragments of large files without reading the entire file into memory.

```
self.fp = np.load(filename, mmap_mode='r')
```

'c' - Copy-on-write: assignments affect data in memory, but changes are not saved to disk. The file on disk is read-only.
'w+' - Create or overwrite existing file for reading and writing.
'r+' - Open existing file for reading and writing.

# List comprehensions

```
li=[]
for x in 'catty':
 for y in 'pot':
  li.append(x+y)

[x+y for x in 'cat' for y in 'potty']
['cp', 'co', 'ct', 'ct', 'cy', 'ap', 'ao', 'at', 'at', 'ay', 'tp', 'to', 'tt', 'tt', 'ty']

    anatomy_rois = {"M1": [3, (1.0, 2.5)], "M2": [3, (1.5, 1.75)],
                    "AC": [3, (0.5, 0.0)], "HL": [3, (2.0, 0.0)],
                    "BC": [3, (3.5, -1.0)], "RS": [3, (0.5, -2.5)], "V1": [3, (2.5, -2.5)]}

    roi_names = anatomy_rois.keys()
    roi_sizes = [anatomy_rois[x][0] for x in anatomy_rois.keys()]
    roi_coord_x = [anatomy_rois[x][1][0] for x in anatomy_rois.keys()]
    roi_coord_y = [anatomy_rois[x][1][1] for x in anatomy_rois.keys()]
    roi_anterior = [anatomy_rois[x] for x in anatomy_rois if anatomy_rois[x][1][1] > 0]
    load_example = [np.load(x, mmap_mode='w+') for x in os.listdir("somedirectory") if 'x' in
os.path.splitext(x)[1]]


Recommended over both filter and map!
```

# Try, catch, else

In the Python world, using exceptions for flow control is common and normal.

```
try:
  filename = self.import_file(filename)
except NotConvertedError:
  qtutil.warning('Failure. Trying conversion method 2’)
  self.conversion_method_two()
except FileAlreadyInProjectError as e:
  qtutil.warning('Skipping file \'{}\' since already in project.'.format(e.filename))
except:
  qtutil.critical('Import of \'{}\' failed:\n'.format(filename) +traceback.format_exc())
else:
  self.listview.model().appendRow(QStandardItem(filename))
<Do other stuff…>
```

# Gohike is Python’s unofficial librarian

http://www.lfd.uci.edu/~gohlke/pythonlibs/

Pip install <path to whl. Don’t forget you can and should press tab>

# QSettings

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
```

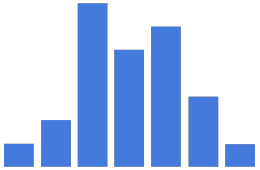
```
def new_video(self):
    filenames = QFileDialog.getOpenFileNames(
      self, 'Load images', QSettings().value('last_load_data_path'),
      'Video files (*.npy *.tif *.raw)')
    if not filenames:
      return
    QSettings().setValue('last_load_data_path', os.path.dirname(filenames[0]))
    self.import_files(filenames)
```

# PyQt and Pyinstaller...

- Quick EDS_conversion example
- Zetcode tutorial -> tetris in a week
- PythonBo -> Very comprehensive YouTube playlist

# Quick overview of my favourite Python visualization tools

- Plotly
- pyqtgraph
- Vispy (Currently in 0.4.0)



# Plotly

Plot.ly is differentiated by being an online tool for doing analytics and visualization. It has robust API's and includes one for python. Browsing the website, you'll see that there are lots of very rich, interactive graphs. They are known to have excellent documentation.

Pros

- Superb documentation. New (2012) Monteal-based company trying to prove themselves
- R, Python, JSON, MATLAB, javascript, Julia all supported. Pick any one and you can use Plotly no problem! Don't need any of the others!
- Stream data online straight from a pi! https://plot.ly/javascript-graphing-library/streaming-data/
- Neat built-in data analytic and beautification-tools. Even has a "Facebook wall": https://plot.ly/304/~Frikster/ https://plot.ly/feed/

Cons

- Not for app building. Can't design your own UI

# pyqtgraph

- imageAnalysis
- Scatterplot
- Hdf5 - Big Data

Simply go to your Python console and type in this to explore more examples yourself:

```
import pyqtgraph.examples
pyqtgraph.examples.run()
```


## Luke Campagnola, Ph.D.

**Scientist I**

Luke joined the Allen Institute for Brain Science in 2015 as part of a team investigating circuits and synaptic properties in the mouse neocortex. Prior to joining the institute, his pre- and postdoctoral research at the University of North Carolina at Chapel Hill focused on understanding the circuitry that underlies the lowest levels of auditory information processing. For his doctoral research under Dr. Paul Manis, Luke used *in vitro* photostimulation to map and characterize the synaptic connections in the cochlear nucleus. As a postdoc, he used these characterizations to develop a physiological model of the cochlear nucleus circuit. Luke received his BS in physics from the Colorado School of Mines and his PhD in neurobiology from the University of North Carolina.

## Research

### Research Interests

The vast complexity of the brain makes it one of the most interesting and difficult topics in science today. Many hypotheses about the brain can only be tested by constructing a physiological model of comparable complexity. This, in turn, relies on experiments that yield detailed information about the substrate of the brain—intrinsic cell properties, circuit connectivity, and synapse properties. These two aspects of neuroscience—modelling and experimentation—form a feedback loop that moves us iteratively toward better models: experimental output drives new models, and model output guides the design of new experiments. I am interested in the use and development high-throughput methods to characterize synapse properties such as connectivity, strength, kinetics and short term plasticity, with emphasis on generating the data we need to constrain circuit models. Ultimately, collecting this data will require a combination of techniques including parallel patch-clamp recordings, photostimulation mapping, tracer injections, and a variety of genetic tools.

**Expertise**

- Systems neuroscience
- Patch electrophysiology and optophysiology
- Physiological circuit modelling

**Research Programs**

- Structured Science
- Research Science -- Cell Types


# The next big Python thing for Neuroscience

Vispy is the sister project of glumpy. It is a high-performance interactive 2D/3D data visualization library. Vispy leverages the computational power of modern Graphics Processing Units (GPUs) through the OpenGL library to display very large datasets.

Pros

- Built to display very large datasets
- Uses the industry standard for high performance graphics

http://articles.beltoforion.de/article.php?a=spiral_galaxy_renderer&hl=en

http://mrob.com/pub/comp/xmorphia/index.html

- Both new. Website have only been around since 2013 (just like R Shiny)

Cons

- Not for app building (at least not any time soon). Build more for creating complex models, simulating etc more than for creating an app that will have different data to deal with every day
- Complex and documentation is barely alive yet https://glumpy.github.io/api-graphics.html#

# 2 out of the 4 lead developers are neuroscientists.

## About

Research software engineer specialized in big data and computational modeling for neuroscience.

Strong background in **mathematics** and **computer science** as a former student of the Ecole Normale Supérieure, PhD in neuroscience, 20+ years of programming experience.

Author of **several books on Python for data science**.

**Specialties**: neuroscience, neuroinformatics, spiking neural networks, machine learning, signal processing, software engineering, scientific computing, parallel computing, data visualization, Python, NumPy, GPU (CUDA, OpenCL, OpenGL), MATLAB, C#, C++, front-end development (web, mobile, Qt)

## Professional experience

- 2012-now: **Postdoc** at University College London, Cortical Processing Laboratory with Kenneth D. Harris. *Spike sorting for large dense arrays.*
- 2009-2012: **PhD** at École Normale Supérieure with Romain Brette. *Correlations in neural coding.*
- 2008: **Undergraduate internship** at Princeton University with Michael Berry. *Statistics of retinal spiking activity.*
- 2007: **Undergraduate internship** at the Collège de France with Alain Berthoz. *Analysis of human locomotor trajectories.*

Cyrille Rossant, PhD CV
data | software | neuroscience
*first.last*@gmail.com
Paris, France

#neuroscience #python #dataviz #maths #gpu #opendata

### books

Two books on **Python for data science**


- High-quality interactive scientific plots with millions of points.
- Direct visualization of real-time data.
- Fast interactive visualization of 3D models (meshes, volume rendering).
- OpenGL visualization demos.
- Scientific GUIs with fast, scalable visualization widgets (Qt or IPython notebook with WebGL).

# Pycharm - the best Python IDE invented by a JetBrains, they are to IDEs what Google is to web search

- Autosaves persistently
- Tools -> python console
- VCS -> checkout
- VCS -> quick example of  pushing to another branch temp and then another temp1, and then commit a bunch to temp1, and then merge with temp
- View -> tool windows (version control -> log to see all commits and branches)
- Legend: red = not added. Blue = uncommitted changes. White = no changes from most recent commit
- Debugging example using filter -> console (show the show current execution point button!)
- Click shift twice to search everywhere
- Heads up that ‘redo’ is ctrl-shift-z and NOT ctrl-y which… deletes the line at the caret
- Invalidate cache/restart
- Settings -> install module, choose Python version (3.4 better than 3.5 for now for me)
- It will make you a better coder

# Time permitting, tutorial on implementing a plugin

1. Go here: https://github.com/Frikster/Mesoscale-Brain-Explorer/releases
2. Download the latest Mesoscale-Brain-Explorer-version.zip
3. Extract it and open dist/pipegui and run pipegui.exe. It should work on Windows. Please submit an issue if it doesn't.

Step through df_f0 to get an idea of what it takes to make your own plugin (show that you can pass functions in python, enumerate)

# Summary

- Use OS
- Use memory mapping and numpy
- It is ok and normal to use try/catch/else for flow control
- List comprehensions enhance code readability and reduce coding time
- Gohike's site is the python world's unofficial librarian
- PyQt and Pyinstaller are industry-grade options for creating intractable applications
- Plotly is great if you want a lighter learning curve, but still want beautiful interactive graphs online
- PyQtGraph is great if you want a lighter learning curve, but still want beautiful interactive graphs offline or integrated with a PyQt app
- Vispy is the future of big data Python visualization
- Pycharm. Pycharm. Pycharm.