Computer lab, hands-on

**Topics**

1. Installing Python
   1. **Anaconda**
   2. Miniconda
2. Jupyter/IPython
   1. Using locally
   2. Markdown, math
   3. Rich display
   4. Saving to html / pdf
   5. Sharing static with nbviewer / github
   6. Sharing dynamic with mybinder (<http://mybinder.org/repo/yoavram/FGMProb>)
   7. Running [Matlab](https://gist.github.com/hururtoriya/a886473e34525a120b1e) / [R](https://www.continuum.io/blog/developer/jupyter-and-conda-r)
   8. Clusters: Wakari, ScienceCluster, JupyterHub
3. Editor
   1. ~~Sublime text~~
   2. [~~Rodeo~~](https://www.yhat.com/products/rodeo/)
   3. ~~PyCharm~~
   4. ~~Visual Studio~~
   5. **Spyder**
4. Introduction to Python
   1. Types (int, float, str, list, tuple, dict, set, complex)
   2. Conditionals (if, elif, else), loops (for, while, break, continue)
   3. Functions, recursion
   4. By value, by reference, scopes, pythontutor, mutable/immutabe
   5. Batch mode (editor), import, assert
   6. IO (input, files (os, os.path, text and binary like fread fwrite), web (urllib/requests), sqlite/json/csv/config/mongo)
   7. **comprehensions**, **lambda (anynomous function)**
   8. Advanced and Functional programming: map, reduce, apply, filter, **decorators**, **generators**, collections module, itertools module, eval, introspection, globals
   9. **OOP**
   10. Exceptions
   11. Debugger
   12. Profiler
5. Installing packages:
   1. Package managers:
      1. conda – install, create envs
      2. pip – install
   2. Install from code
   3. Install from Gohlke's site
   4. \_\_version\_\_
   5. requirements.txt
6. **Unit testing**: nose
7. **\*Creating packages:**
   1. \_\_init\_\_
   2. setup.py, MANIFEST.in, automatic package generation
   3. conda build
   4. deploy: pypi / anaconda.org
   5. Versioning with versioneer
   6. Travis-ci / wercker
8. \*Py2/Py3: convert with futurize
9. Numpy introduction (travis' presentation)
   1. By ref by value
   2. Access
   3. Operators
   4. A.sum same as np.sum(A)
10. Long calculations (matrix inversion)
    1. Timing runtime
    2. Alert: email, beep, notification
11. Plotting
    1. Matplotlib, \*mplot3d (notebook/inline)
    2. Seaborn
    3. Bokhe / plotly / ggplot?
    4. Maps (elections 2009)
    5. ivisual: solar system
    6. animations (Moran)
    7. widgets (Logistic)
    8. backends - figures without hanging
12. Event driven programming
    1. watchdog
13. UI
    1. CLI – click
    2. GUI – tkinter, **~~pyqt~~**
    3. ~~Web – flask + jQuery / Angular.js~~
    4. ~~Wooey, Gooey – not ready yet~~
    5. ~~Hosting on digitalocean / heroku~~
14. Data analysis
    1. Py4Life examples: animals (visualizing, pandas), growth models (regression, curve fitting)
    2. Elections 2009
    3. Machine learning - irises
    4. Scikit-learn website
    5. Think Stats
    6. Think Bayes
    7. [Bayesian model fitting and selection](http://jakevdp.github.io/blog/2014/06/14/frequentism-and-bayesianism-4-bayesian-in-python/)
    8. [Traffic fatality data](http://blog.yhat.com/posts/traffic-fatalities-in-us.html)
15. API
    1. requests
    2. Python APIs
16. Image analysis
    1. Scikit-image website
    2. Scipy.ndimage website
    3. Seeds
    4. **Denoising** (Obama etc.)
17. DSP
    1. Violin example
    2. Think DSP
    3. Convolution, linear filter
18. Numerical analysis
    1. Gillespie algorithm (SSA)
    2. Spatial dynamics: Moran process
    3. ODE, SDE: Logistic growth
    4. Py4Life: growth, stochastic processes
19. Symbolic math with Sympy
    1. <https://github.com/raoulb/sympy_ipython_notebooks>
20. More resources:
    1. <http://Matlab2python.netlify.com>
    2. <https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks>