Introduction to Scientific Computing in Python

219116/219117 Computer Programming II

Piyamate Wisanuvej

March 2021

Based on materials from http://scipy-lectures.org/

Scientific computing in Python

- The scientist’s needs
  - Get data
  - Process data
  - Visualize results
- Why Python?
  - Rich collection of already existing bricks of classic numerical methods, plotting or data processing tools
  - Efficient code: quick development times and quick execution times

How does Python compare to other solutions?

- Compiled languages: C, C++, Fortran…
  - Very fast
  - Painful usage: no interactivity during development, mandatory compilation steps, verbose syntax, manual memory management
- Matlab scripting language
  - Very rich collection of libraries with numerous algorithms
  - Commercial support is available, Matlab is not free
- Other scripting languages: Scilab, Octave, R, IDL, etc.
  - Open-source, free, or at least cheaper than Matlab
  - Fewer available algorithms than in Matlab
- Python
  - Very rich scientific computing libraries
  - Free and open-source software
  - A variety of powerful environments to work in: IPython, Spyder, Jupyter notebooks, Pycharm, Visual Studio Code

The Scientific Python ecosystem

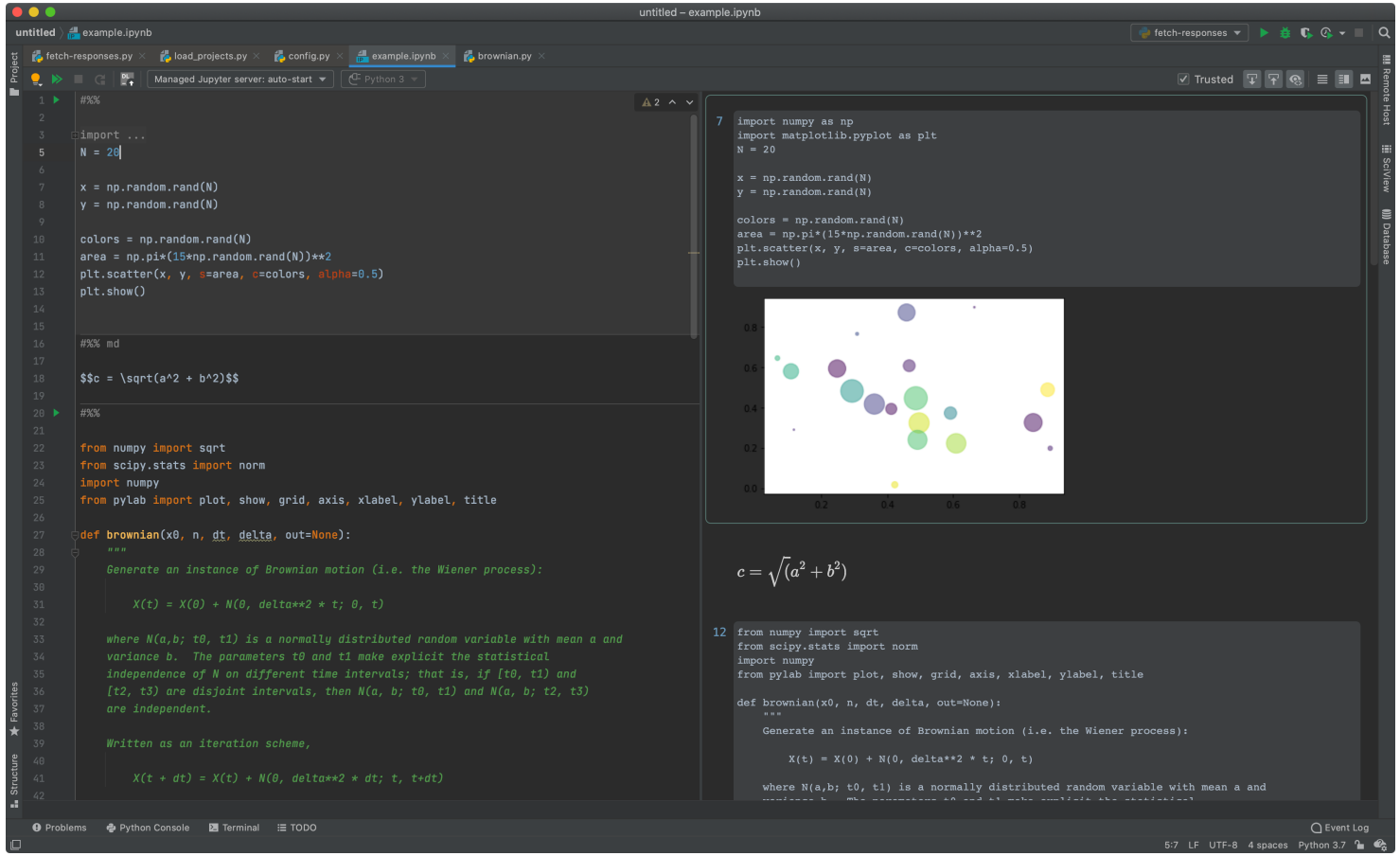
- Core numeric libraries
  - **NumPy**
  - **SciPy**
  - **Matplotlib**
- Domain-specific packages, for example:
  - **Mayavi** for 3-D visualization
  - **pandas, statsmodels, seaborn** for statistics
  - **sympy** for symbolic computing
  - **scikit-image** for image processing
  - **scikit-learn** for machine learning
- Advanced interactive environments
  - **IPython**, an advanced Python console
  - **Jupyter**, notebooks in the browser

Development Environments / Tools

- IPython
- Jupyter
- Spyder
- Anaconda
- PyCharm
- Google Colab

PyCharm

PyCharm IDE supports Jupyter and IPython.



Google Colab

- Python on web browser optimized for scientific computation
- Access to packages via PyPI.
- Your code is executed in a virtual machine private to your account.
- Free with some limitations of computing resources.

NumPy

- NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.
- Cheat Sheet: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf
- Reference: https://numpy.org/doc/stable/reference/index.html#reference

Development Environment Installation

Anaconda (https://www.anaconda.com/products/individual#Downloads) is recommended as it contains most relavant tools in one package:

- IPython
- Jupyter
- Spyder

Alternatively, if you already have a working Python environment and you prefer each component separately, they are available on PyPI.

$ pip3 install ipython

$ pip3 install jupyter

$ pip3 install spyder