notebook_1_why_python

March 21, 2018

```
In [1]: %run talktools

ERROR: File `'talktools.py'` not found.
```

1 Why Python?

1.1 Objectives

- Present the advantages of Python
- Provide references to other resources
  - Installation
  - Editors

1.2 The Scientists Needs

- Get data
- Manipulate/process data
- Analyze data
  - Visualize
  - Model
  - Search for patterns

1.3 Getting data - Python Strengths

- Reading data as a stream
- String processing
- Libraries for cloud computing
  - Jupyter notebooks for parallel processing
  - Pyspark for Hadoop

1.4 Other strengths

- **Batteries included** Has a full-features standard library
- **Easy to learn** Built to read like pseudo code
- **Numerical tools** Speed up the slower parts of your code

  - **Cython** Compile python to C
  - **Numba** Just-in-time compiler

- **Used by software developers** Allows access to tools/libraries for other tasks

2 The Scientific Python ecosystem

- Python consists of
  - General purpose language and libraries
  - Third-party libraries for computation
  - Third-party libraries for data science

2.1 Python

A generic and modern computing language

- The language:
  - flow control,
  - data types (`string`, `int`),
  - data collections (lists, dictionaries), etc.
- The standard library:
  - string processing,
  - file management,
  - simple network protocols.
- Many specialized modules or applications written in Python:
  - web framework, etc. ... and
  - scientific computing.
- Development tools (automatic testing, documentation generation)

Core numeric libraries

- **Numpy**: numerical computing with powerful **numerical arrays** objects, and routines to manipulate them. http://www.numpy.org/
- **Scipy** : high-level numerical routines. Optimization, regression, interpolation, etc http://www.scipy.org/
- **Matplotlib** : 2-D visualization, "publication-ready" plots http://matplotlib.org/

Advanced interactive environments:

- **IPython**, an advanced **Python console** http://ipython.org/
- **Jupyter**, **notebooks** in the browser http://jupyter.org/

Domain-specific packages,

- **Mayavi** 3-D visualization
- **pandas, statsmodels, seaborn** statistics
- **sympy** symbolic computing
- **scikit-image** image processing
- **scikit-learn** machine learning
- **Pyspark** Interface to Hadoop/Spark framework

2.2 Before starting: Installing a working environment

- Python for data science
- Complex installation
- Many inter-related components
- **Solution** Use a pre-packaged distribution like
- Anaconda https://www.continuum.io/downloads
- EPD https://store.enthought.com/downloads
- WinPython https://winpython.github.io

2.3 Python 3 or Python 2?

- Python 2
  - No longer in development
  - Has some worts
- Python 3
  - Released in 2008
  - Fixed and improved the language
  - A few packages still don't work for Python 3
- **Recommendation** Use Python 3

3 Writing Python Code

- A number of way to process code in Python
  - Interactive interpreter
  - Notebook
  - Editor

3.1 Working Interactively with Jupyter

- Jupyter Console
  - Good for quickly testing code
  - Can be used like bash
- Jupyter notebook

  - Literate programming
  - Renders text, images, math
  - Like Rmarkdown with live code
  - Jupyterhub - Classroom environment

- Features of both

  - Useful magic commands
  - Program in many languages at once (R, Julia, Octave)
  - Remote connection to server
  - Access to parallel processing

3.2 Writing Program in an IDE

- Integrated Development Environment

  - Editor
  - Interpreter
  - Other features

- Some examples
- Spyder https://pythonhosted.org/spyder/

  - IPython console,
  - a debugger,
  - a profiler...

- PyCharm https://www.jetbrains.com/pycharm

  - IPython console,
  - notebooks,
  - a debugger... (freely available, but commercial)

- Atom https://atom.io