



Learning Python

Getting results for beamlines and scientific programming

0. Overview: getting started

The Advanced Photon Source is an Office of Science User Facility operated for the U.S. Department of Energy Office of Science by Argonne National Laboratory



What will these presentations cover?

1. Basic Python
 - a) Variables & simple data types
 - b) Higher-level data types
 - c) Assignments & expressions
 - d) File I/O
 - e) Error handling
 - f) Built-in functions & modules
 - g) Writing and calling functions
2. Using Python
 - a) Simple graphics with matplotlib
 - b) Beamline controls via EPICS
 - c) Computations with NumPy and SciPy

The Advanced Photon Source is an Office of Science User Facility operated for the U.S. Department of Energy Office of Science by Argonne National Laboratory

5/10/11 2

What will these presentations cover?

3. Intermediate Python
 - a) Understanding classes (objects)
 - b) Creating classes
 - c) Advanced parameter lists
4. Graphical User Interface (GUI) programming with wxPython
 - a) Basic concepts in wxPython
 - b) Using sizers for layout
 - c) Using matplotlib and wxPython together
5. Good programming practices in Python:
 - Modules vs classes/Unit tests/Avoiding loops
6. Use of MySQL and Python

The Advanced Photon Source is an Office of Science User Facility operated for the U.S. Department of Energy Office of Science by Argonne National Laboratory

5/10/11 3

Other references

- If you have never written a computer program, I urge you to read the textbook, *Python for Software Design: How to Think Like a Computer Scientist* by Allen B. Downey (2009). In parts of the lectures, I will reference sections in this book.
 - ~\$35 paperback
 - Online: <http://www.greenteapress.com/thinkpython/thinkCSPy/html/index.html>
 - Download: <http://www.greenteapress.com/thinkpython/thinkCSPy/thinkCSPy.pdf>
- Other books:
 - I used *Learning Python: Powerful Object-Oriented Programming* by Mark Lutz
 - Dive into Python 3 by Mark Pilgrim (online at <http://diveintopython3.org/>). ~ Same price as Downey. Assumes Python 3 – which I will not use here.
- Web sites:
 - “Helping scientists make better software:” http://software-carpentry.org/4_0/python/
 - The canonical Python tutorial: <http://docs.python.org/tutorial/index.html>
 - Complete Python documentation: <http://docs.python.org> (N.G. as language reference)

The Advanced Photon Source is an Office of Science User Facility operated for the U.S. Department of Energy Office of Science by Argonne National Laboratory

5/10/11 4

Why Python?

Python is a scripting language, code is interpreted as it is executed

- Scripting allows for quicker code development (much quicker IMHO!)
- Good error handling
- Test code out as you type it
- Python is free (will talk about how to get it later; note some dists cost \$)
- but slower execution than compiled languages (C, C++, Fortran)
 - Note, however, for most uses, computer speed is not as important as the human investment in coding

Python includes packages that offer the following features

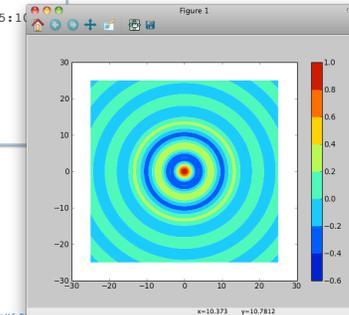
- Fast matrix & scientific computations: NumPy & SciPy
- Powerful GUI tools (wxPython and PyQt; not recommended: Tk)
- Powerful plotting packages (Matplotlib, Chaco)
- 3D visualization (mayavi)
- Access to beamline (EPICS) controls (ca_util or PyEPICS)

Python code can allow one to write short programs quickly that do complex things

```
# load packages
import numpy
import scipy.special
import matplotlib.pyplot as mpl

# compute Bessel function
x,y = numpy.mgrid[-25:25:100j,-25:25:100j]
r = numpy.sqrt(x**2 + y**2)
B = scipy.special.j0(r)

# now do plotting
mpl.contourf(x,y,B)
mpl.colorbar()
mpl.show()
```

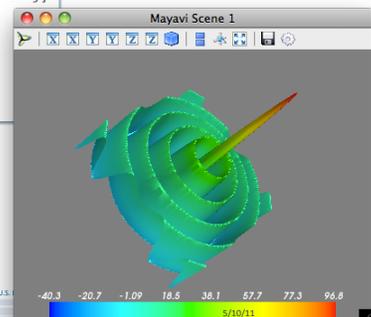


Or with some very minor changes

```
# load packages
import numpy
import scipy.special
import enthought.mayavi.mlab as mmlab
```

```
# compute Bessel function
x,y = numpy.mgrid[-25:25:100j,-25:25:100j]
r = numpy.sqrt(x**2 + y**2)
B = 100*scipy.special.j0(r)
```

```
# now do plotting
mmlab.surf(B)
mmlab.colorbar()
mmlab.show()
```



Python is changing

In the past few years, a decision was made to redo aspects of the Python language to make it more systematic, this was implemented in Python 3.0 (now up to 3.2) however, the older version of Python is still being updated (current is 2.7.1 from 10/2010)

This course will use Python 2.x for several reasons:

- Many add-on packages are not yet ready for Python 3
- BCDA (ca_util) is still at Python 2.5
- I personally have not started working with Python 3

Changes in the language between 2.x and 3.x are not very large and I will try to note them as we go. For a complete list see

http://pitgmedia.pearsoncmg.com/imprint_downloads/informit/promotions/python/python2python3.pdf

Getting Python

Python distributed for free and is provided on Macs and most Linux distributions – but may not have all packages to be discussed here:

- wxPython (GUI)
- Matplotlib (plotting)
- NumPy and SciPy (scientific computation)
- cautils (EPICS) – only on beamline computers
 - On beamline computers, use /APSShare/bin/python to start Python with all of above

For a full-featured Python distribution, I recommend purchasing the \$200 EPD distribution (all platforms) <http://www.enthought.com/products/epd.php>
AES has purchased a limited number of EPD licenses

- Free alternate for Windows & Linux: pythonxy <http://www.pythonxy.com/>
- Also <http://www.activestate.com/activepython/downloads> (free, not tested)

Running Python

One can run Python interactively by simply starting the program and typing in program statements or by supplying statements in a file

```
s11bmsrv1:- toby$ /APSShare/bin/python
Python 2.5.2 (r252:60911, Jan 28 2009, 15:33:22)
[GCC 4.1.2 20070626 (Red Hat 4.1.2-13)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello world"
hello world
>>>
```

```
s11bmsrv1:- toby$ cat > /tmp/demo.py
print "hello world"

s11bmsrv1:- toby$ cat /tmp/demo.py
print "hello world"
s11bmsrv1:- toby$ /APSShare/bin/python /tmp/demo.py
hello world
s11bmsrv1:- toby$
```

Running Python interactively

Typing python statements in at the command line gets old fast (no editing features); better choices exist:

ipython: good basic shell, works well as a matlab replacement (not in APSShare)

idle: older shell, IDE (not in APSShare)

Eclipse/PyDev: sophisticated program development environment. Worth learning. See talk from Pete Jemian (works with APSShare)

https://cmsdocs.aps.anl.gov/docs/idcplg?idcService=DISPLAY_URL&dDocName=APS_1419152

emacs: includes a mode for coding and running Python (not the best choice unless you already know and love emacs)

Wingware: This is a commercial IDE that has a free version (Wing IDE 101 <http://wingware.com/downloads/wingide-101>)

Editing Python in a standard text editor is not recommended

Use a Python-aware editor

As you will see later, Python is particular about getting indents right and an extra space can matter. If you use notepad, etc. to edit Python scripts you will waste a lot of time. A smart editor will take care of this for you.

The previous IDEs mostly include Python-aware editors

There are lots of good free choices:

- Windows:
 - Notepad++
 - Context

See <http://wiki.python.org/moin/PythonEditors> for an extensive discussion

If I were starting today, I would probably try out Eclipse/PyDev.

Python trivia

- Python was created by mathematician/computer scientist Guido van Rossum in 1989 in the Netherlands
 - He now works for Google
- Python is now developed by a large community but Guido has been named by them "Benevolent Dictator For Life" (BDFL) and still plays a central role.
- Python was named after Monty Python (BBC comedy series, 1969-1974) 🐍

Homework

1. Figure out what distribution of Python you plan to use
2. Have it installed
3. Figure out what editor/IDE you want to use
4. Have it installed