Introduction to Python

Arno Proeme, ARCHER CSE Team

aproeme@epcc.ed.ac.uk

Attributed in part to Jussi Enkovaara &

Martti Louhivuori, CSC Helsinki

Reusing this material

http://www.archer.ac.uk
support@archer.ac.uk

Python origins

- Created early 1990s (Guido van Rossum, CWI)
- Driven by desire to provide more programmer-friendly alternative to C to speed up application development
- Inspired by an earlier interactive programming environment and language (ABC)
- Not created specifically for scientific computing (unlike e.g. Fortran)

Python now

- Most popular first taught programming language at top 39 US computer science departments

- Used by Youtube, Dropbox, Google, Industrial Light & Magic, Quant Finance, …

- Version 3.x breaks backwards compatibility with 2.x
  - 2.x still most widely used, including in this course

In natural sciences & engineering?

- Used mainly:
  - As a multipurpose workflow environment for data analysis and visualisation
  - As "glue", i.e. interface code, to heavy numerical kernels written in a compiled language like C/C++ or Fortran (e.g. Fluidity, ASE)
  - For rapid prototyping of algorithms
  - For non-HPC simulations
- Though performance continues to improve and there are some 100% Python codes (e.g. GPAW), these are still not widely used for heavy numerics.

Python characteristics

- Python is a **high-level** language (compared e.g. to C),
  - Simple syntax, more easily readable code and shorter programs

  but
  - Sacrifice some performance due to abstraction overheads
  - Development time considered more valuable than compute time

- Python is a fully-featured general purpose programming language (like C, C++, Fortran, Java, etc.)

- Python supports (but does not enforce) different programming styles, e.g. object-oriented

- Python is open source

The Python interpreter

- Python code is not generally compiled into a standalone executable, but executed by the Python interpreter, `python`

- Python code contained in a script file (ending in .py) can be execute by the interpreter as follows:

```
aproeme$ cat hello.py
print(“Hello World”)
aproeme$ python hello.py
Hello World
```

Interactive Python

- If not supplied with an input script file, the Python interpreter runs as an interactive Python runtime environment (a Python shell session)

```
aproeme$ python
```

Interactive Python

- If not supplied with an input script file, the Python interpreter runs as an interactive Python runtime environment (a Python shell session)

```
aproeme$ python
Python 2.7.7 |Anaconda 2.0.1 (x86_64)| (default, Jun  2 2014, 
12:48:16)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more 
information.
Anaconda is brought to you by Continuum Analytics.
>>>
```

Interactive Python

- If not supplied with an input script file, the Python interpreter runs as an interactive Python runtime environment (a Python shell session)

```
aproeme$ python
Python 2.7.7 |Anaconda 2.0.1 (x86_64)| (default, Jun  2 2014,
12:48:16)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
Anaconda is brought to you by Continuum Analytics.
>>> print(“Hello World”)
```

Interactive Python

- If not supplied with an input script file, the Python interpreter runs as an interactive Python runtime environment (a Python shell session)

```
aproeme$ python
Python 2.7.7 |Anaconda 2.0.1 (x86_64)| (default, Jun  2 2014,
12:48:16)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
Anaconda is brought to you by Continuum Analytics.
>>> print("Hello World")
Hello World
```

Interactive Python

- Python shell lets you explore Python functionality directly without needing to compile your code
- This is useful for incremental / progressive code development and rapid prototyping
- In case of any errors, debugging (TraceBack) information is provided within the Python shell (which usually does not simply crash)
- Once you have worked out how to get Python to do what you want it to, save the code as a Python script (.py file)

Interactive Python vs Matlab *et al*

- The experience of using interactive Python to work, especially iPython, is similar to using other scripting languages e.g. Matlab, Mathematica, Maple, R, etc.

- As well as having a good range of scientific libraries Python is more easily extendable

- As popularity grows more and more packages become available, Python becomes the preferred workflow shell to tie everything together

Data types

- Variables in Python are dynamically typed
  - i.e. don't specify explicitly whether int, string, etc.
  - Type is determined based on format of assigned value or other variables involved in calculation

```
X = 1.0
my_name = Arno

Y = my_name + X
```

The slides that follow are attributed to:

Jussi Enkovaara & Martti Louhivuori, CSC Helsinki

Numerical data types

- Integers
- Floats
- Complex numbers
- Basic operations
  - + and –
  - *, / and **
  - Implicit type conversions
  - Be careful with integer division!

```
x = 4
y = 6.0
z = 1.4 + 4.2j
```

```
>>> 4.0 + 5 – 2
7.0
>>> 2.0**2 / 2.0*(4.2-2j)
(8.4-4j)
>>> 2/5
0
>>> 2./5
0.4
```

String

- Strings are enclosed by “ or '
- Multiline strings can be defined with three double quotes

```
s1 = “very simple string”
s2 = 'same simple string'
s3 = “this isn't so simple string”
s4 = 'is this “complex” string?’
s5 = “””This is a long string
expanding to multiple lines,
so it is enclosed by three “'s”””
```

+ and * operators with strings:

```
>>> "Strings can be " + "combined"
'Strings can be combined'
>>> "Repeat! " * 3
'Repeat! Repeat! Repeat!
```

Data structures

- Lists
- Tuples
- No arrays! (wait for NumPy)

Lists

- Python lists are dynamic arrays
- List items are indexed (index starts from 0)
- List item can be any Python object, items can be of different type
- New items can be added to any place in the list
- Items can be removed from any place in the list

Lists

- Defining lists

```
>>> l1 = [3, “egg”, 6.2, 7]
>>> l2 = [12, [4, 5], 13, 1]
```

- Accessing list elements

```
>>> l1[0]
3
>>> l2[1]
[4, 5]
>>> l1[-1]
7
```

- Modifying list items

```
>>> 1[-2] = 4
>>> l1
[3, 'egg', 4, 7]
```

Lists

- Adding items to list

```
>>> l1 = [9, 8, 7, 6]
>>> l1.append(11)
>>> l1
[9, 8, 7, 6, 11]
>>> l1.insert(1,16)
>>> l1
[9, 16, 8, 7, 6, 11]
>>> l2 = [5, 4]
>>> l1.extend(l2)
>>> l1
[9, 16, 8, 7, 6, 11, 5, 4]
```

- + and * operators with lists:

```
>>> [1, 2, 3] + [4, 5, 6]
[1, 2, 3, 4, 5, 6]
>>> [1, 2, 3] * 2
[1, 2, 3, 1, 2, 3]
```

Lists

- It is possible to access slices of lists
- `>>> l1 = [0, 1, 2, 3, 4, 5]`
-
```
>>> l1[0:2]
[0, 1]
>>> l1[:2]
```
-
```
[0, 1]
>>> l1[3:]
[3, 4, 5]
>>> l1[0:6:2]
```
-
```
[0, 2, 4]
>>> l1[::-1]
[5, 4, 3, 2, 1, 0]
```
- Removing list items

```
>>> second = l1.pop(2)
>>> l1
[0, 1, 3, 4, 5]
>>> second
2
```

Tuples

- A tuple is number of comma-separated values, e.g.:
- `>>> t = ‘a’,2,3`
- `t[0]= bla`
- `Traceback (most recent call last):`
- `File “<stdin>”, line 1, in <module>`
- `TypeError: ‘tuple’ object does not support item assignment`

Variables

- Python variables are references

```
>>> l1 = [1,2,3,4]
>>> l2 = l1
```

- l1 and l2 are references to the same list
- Modifying l2 changes also l1!
  - `>>> l2[0] = 0`
  - `>>> l1`
    `[0, 2, 3, 4]`
- Copy can be made by slicing the whole list
  - >>> l3 = l1[:]
  - >>> l3[-1] = 66
  - >>> l1
    [0, 2, 3, 4]
  - >>> l3
    [0, 2, 3, 66]

Objects

- Object is a software bundle of data (=variables) and related methods
- Data can be accessed directly or only via the methods (=functions) of the object
- In Python, everything is an object
- Methods of object are called with the syntax
    - obj.method
- Methods can modify the data of object or return new objects

Standard Library

- Standard library includes:
  - OS interface
  - Basic Maths functions & random number generator
  - Performance measurement
  - Output formatting
  - Data compression
  - Internet access
  - Simple multithreading
  - Logging

Misc.

- Third party Python packages (modules) are loaded with
- `import modulename`
- Code blocks are indented
- Documentation:
  - https://docs.python.org/2.7/
  - http://scipy-lectures.github.io/