python

Cosmology with python:

Beginner to Advanced in one week

Tiago Batalha de Castro

What is Python? (From python.org)

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics
- It is very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together
- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance
- Python supports modules and packages
- It is Free

Python is an interpreted, object-oriented (...)

- Python does not require compilers.
  - Easier to debug
    - No waste of time in the eternal loop code → compile → check → code ...
  - But also easier to write bad codes
    - No warnings, clues, or tips
- About Object-oriented programming: well… We will see what it means next class

```
import this
"""The Zen of Python, by Tim Peters. (poster by Joachim Jablon)"""

Beautiful is better than ugly.
Explicit is better than impl..
Simple is better than cσмplεx.
Cσмplεx is better than cOmp1|c@ted.
Flat is better than nested.
S p a r s e  is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
raise PythonicError("Errors should never pass silently.")
# Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
# Although that way may not be obvious at first unless you're Dutch.
Now is better than ...                    never.
Although never is often better thanrightnow.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Coding…

What else do we have?

- Python has 5 **standard** data types:
  - Numbers
  - Strings
  - List
  - Tuple
  - Dictionary

What else do we have?

- Python has 5 **standard** data types:
  - ~~Numbers~~ (can be complex too)
  - ~~Strings~~
  - List
  - Tuple
  - Dictionary

Lists

- Python's lists look like arrays
- They are limited by [ ]’s
- They are indexed by Position (0 up to N-1)
- Lists elements can be updated
- Lists can **not** be upgraded!
  - Well...They can be appended but that’s a hack that I really really don’t recommend!

Tuples

- Sequence data type that is similar to the list:
- Tuples are enclosed by parenthesis
- Like lists tuple's elements are indexed by position (starting from 0 up to N-1)
- Unlike lists tuple's elements can not be updated

```
IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: tuple=(1,1)

In [2]: tuple[0]
Out[2]: 1

In [3]: tuple[0]=2
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-e7a35d070e9f> in <module>()
----> 1 tuple[0]=2

TypeError: 'tuple' object does not support item assignment
```

Dictionaries

- Python's dictionaries are associative tables
- They are limited by { }
- They are indexed (Hashed) by any immutable Python object instance
- Like lists they can be updated
- Unlike lists they can be upgraded!

Coding…

List Comprehension

- Lists can be written in a comprehensive fashion
  - Remember Python's Zen:
    - There should be one-- and preferably only one --obvious way to do it
  - Usually when your result best fit on a list, List Comprehension is the obvious way to do it.

Coding…

Loops, Conditionals, and Functions

- Any structure on python is delimited by **INDENTATION**
  - So, indentation on python is **mandatory**!

```
In [1]: print "Hello World"
   ...: print "Hello World"
   ...:
Hello World
Hello World

In [2]: print "Hello World"
   ...:   print "Hello World"
  File "<ipython-input-2-31a6ca30e5ab>", line 2
    print "Hello World"
    ^
IndentationError: unexpected indent
```

Functions

```
In [1]: def my_first_function (x):
   ...:     return x**2
   ...:

In [2]: my_first_function(2)
Out[2]: 4

In [3]: my_first_function(2.)
Out[3]: 4.0

In [4]: my_first_function(2)/5
Out[4]: 0

In [5]: my_first_function(2.)/5
Out[5]: 0.8
```

Conditionals

- If, elif, else structure:

```
if (boolean operation 1):
    Do something if True
elif (boolean operation 2):
    Do something if True
            .
            .
            .
else:
    Do something if none of the others are True
```

Conditionals

- Python has the boolean operators:
  - The common **<, <=, >, >=, ==, !=** operators
    - The pythonic version of this operators can be chained
      - Ex.: x<y<z means (x<y) AND (y<z)
    - The operators AND/OR/NOT are… **and**/**or**/**not** ;)
    - The Membership operator **in**
    - Identity operators **is**
      - **is** is different than **==** !!!

Challenge (1)

Using the boolean operators write a pure list comprehension to print all the prime numbers between 1...1000

Loops

- Python has two loops: while and for
  - Just that!?
    - 1 - Remember, there should be **one** obvious way to do anything on Python; any algorithm using other loop structure (like do...while) can be adapted to Python using for and while with minimum changes
    - 2 – Both Pythonic versions of do and while are more powerful than cpp/c versions

Coding…

https://www.learnpython.org/en/Loops

Challenge (2)

- Write a function that given an integer “Z” the function prints the Fibonacci series up to “Z”
- Would you be able to rewrite it as a list comprehension?(*)

Advanced parenthesis

- xrange vs range

xrange(*start*, *stop*[, *step*])

This function is very similar to `range()`, but returns an xrange object instead of a list. This is an opaque sequence type which yields the same values as the corresponding list, without actually storing them all simultaneously. The advantage of `xrange()` over `range()` is minimal (since `xrange()` still has to create the values when asked for them) except when a very large range is used on a memory-starved machine or when all of the range's elements are never used (such as when the loop is usually terminated with `break`). For more information on xrange objects, see XRange Type and Sequence Types — str, unicode, list, tuple, bytearray, buffer, xrange.

CPython implementation detail: `xrange()` is intended to be simple and fast. Implementations may impose restrictions to achieve this. The C implementation of Python restricts all arguments to native C longs ("short" Python integers), and also requires that the number of elements fit in a native C long. If a larger range is needed, an alternate version can be crafted using the `itertools` module: `islice(count(start, step), (stop-start+step-1+2*(step<0))//step)`.

Advanced parenthesis

- xrange vs range

xrange(*start*, *stop*[, *step*])

This function is sequence type which yields the same advantage of `xrange()` over `range()` is a very large range is used on a memory when the loop is usually terminated with es — str, unicode, list, tuple, bytearray

CPython implementation detail: mpose restrictions to achieve this. The C implementation of Python restricts all arguments to native C longs ("short" Python integers), and also requires that the number of elements fit in a native C long. If a larger range is needed, an alternate version can be crafted using the `itertools` module: `islice(count(start, step), (stop-start+step-1+2*(step<0))//step)`.

More details next class!!

What is Python? (From python.org)

- ~~Python is an interpreted, object-oriented, high-level programming language with dynamic semantics~~
- It is very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together
- ~~Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance~~
- Python supports modules and packages
- ~~It is Free~~

Python supports modules and packages

- Python has a huge community:
  - Before starting any project search for ongoing projects that you can join and collaborate;
  - Even if you think it is too technical
    - For instance, Steven Murray's packages for cosmology are great and include several technicalities like HMF calculation, Halo-Model implementations, Observational and Theoretical Cosmology codes…

Numpy

- Fundamental package for **array** operations:
  - N-dimensional sophisticated array object (like valarray, for c++ experts)
  - Numerical Integration
  - Linear algebra, Fourier transform, and random number capabilities
  - Efficient multi-dimensional container of generic data

  Numpy is probably the reason why Python is very famous among scientist while Ruby is not!

Coding…

Scipy

- Fundamental package for **scientific** computing with Python
  - Basic functions and Special functions (scipy.special), Integration (scipy.integrate), Optimization (scipy.optimize), Interpolation (scipy.interpolate), Fourier Transforms (scipy.fftpack), Signal Processing (scipy.signal), Linear Algebra (scipy.linalg), Statistics (scipy.stats), Multidimensional image processing (scipy.ndimage), File IO (scipy.io), and more...

Coding…

Matplotlib

- Fundamental package for publication quality figures
  - Very powerful
  - Very cumbersome for advanced plotting, though…
    - On the other hand the documentation is really complete and friendly (https://matplotlib.org)
    - As always, with a huge community someone has already tried to make a plot similar to what you want… Google it!

Coding…

10-Errors should never pass silently (?)

- Error handling is a fundamental step on high-level coding:
  - For instance, what happens if you try to save complex numbers as float32 on Mathematica?


:(
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)
If you'd like to know more, you can search online later for this error: HAL_INITIALIZATION_FAILED

10-Errors should never pass silently (?)

- Error handling is a fundamental step on high-level coding:
  - For instance, what happens if you try to save complex numbers as float32 on Mathematica?
  - You can always solve that problem with some if/else structures… Boring/Ugly :P!
- Python has a very sophisticated way to handle errors

Exceptions

- When the Python interpreter find an expression synthetically wrong or ill-defined it returns an exception:
  - Common exceptions are: SyntaxError, ZeroDivisionError, NameError, TypeError (https://docs.python.org/2/library/exceptions.html)

Handling Exceptions

- The try statement works as follows:
  - First, the try clause is executed.
  - If no exception occurs, the except clause is skipped and execution of the try statement is finished.
  - If an exception occurs, the rest of the clause is skipped. Then if its type matches one of the exception named, the except clause is executed, and then execution continues after the try statement.
  - If an exception occurs that does not match the exception named, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops with a message.

Raising Exceptions

- The raise statement allows the programmer to force a specified exception to occur:
  - Essential when developing an open code
- numpy.seterr(‘raise’) raises Raise a FloatingPointError when a ill-defined operation is executed using a numpy object

Coding…

Homework:

- Using the integration routines of scipy write functions that calculate for a LCDM scenario:
  - Comoving Distance
  - Angular Diameter Distance
  - Luminosity Distance
- Implement Einsenstein & Hu fitting function for Power Spectrum and produce comparison plots with CAMB (https://lambda.gsfc.nasa.gov/toolbox/tb_camb_form.cfm)


That's all Folks!