

Python Advance Course via Astronomy street



Sérgio Sousa (CAUP)

ExoEarths Team (<http://www.astro.up.pt/exoearths/>)

Why Python?

What do we need?

- Get data – Observations / Simulations
- Manipulate and process the data
- Analysis and Visualization of the data
- Making plots for papers/presentations/reports etc...
- Other stuff???

Why Python?

Some Languages Available in the market:

	Readability	Speed	Vectorized Operations	Astronomy Routines	Multithreading	User Community	Cost & Copyright
Python	:D	Interpreted – Slow Compiled C/ Fortran modules – Medium	With NumPy	Many separate modules*, PyRAF	Easy, several modules	Gigantic, multi-discipline (science, web, games...)	Free & Open
IDL	:/	Interpreted – Slow Built-ins – Medium	Yes	Astrolib	Difficult	Astronomers: Large Others: Tiny	Expensive & Annoying
IRAF	:(Interpreted – Slow Built-ins – Medium, but disk-write bound	evvexpr?	...IRAF	None	Only Astronomers	Free & Open
C	:(::	Compiled – Fast	No	Many libraries	Difficult	Gigantic, multi-discipline	Free & Open
Fortran	*@#*&	Compiled – Fast	Yes	Many libraries	Easy	Scientists, Septuagenarians	Free & Open

*There are current attempts to make a monolithic Astrolib-like package for Python

Why Python?

Advantages

- Many scientific libraries (modules) available; (Maybe to many!)
- Simple and intuitive sintaxe; very readable;
- Many other libraries available (web, interfaces, system, ...);
- Free and open-source software, widely spread;

Drawbacks:

- ~~Still no pleasant development enviroment (missing a good IDE);~~
- Dependence on modules that slowly follow Python updates
- Careful with system python module version (you can use anaconda like systems)

Scientific Python

Building Blocks

- Python
- IPython
- Numpy
- Scipy
- Matplotlib
- Pyraf
- Mayavi

Bibliography

- Python Scientific lecture notes
<http://scipy-lectures.github.io/>
- Python Reference
<http://docs.python.org/2/reference/>
- <http://www.python.org/>
- <http://www.astropy.org/>

Python Advance Course via Astronomy street

Advance Course Outline:

- Lesson 1: Python basics (T + P)
- Lesson 2: Python with Numpy and Matplotlib (T + P)
- Lesson 3: Science and Astronomy modules (Scipy, Astropy/Pyfits) (T + P)
- ~~Lesson 4: Interfacing with C (P)~~

Python Advance Course via Astronomy street

Advance Course Outline:

- Lesson 1: Python basics (T + P)
- Lesson 2: Python with Numpy and Matplotlib (T + P)
- Lesson 3: Science and Astronomy modules (Scipy, Pyfits/Astopy, Pyraf) (T + P)

Python Advance Course via Astronomy street

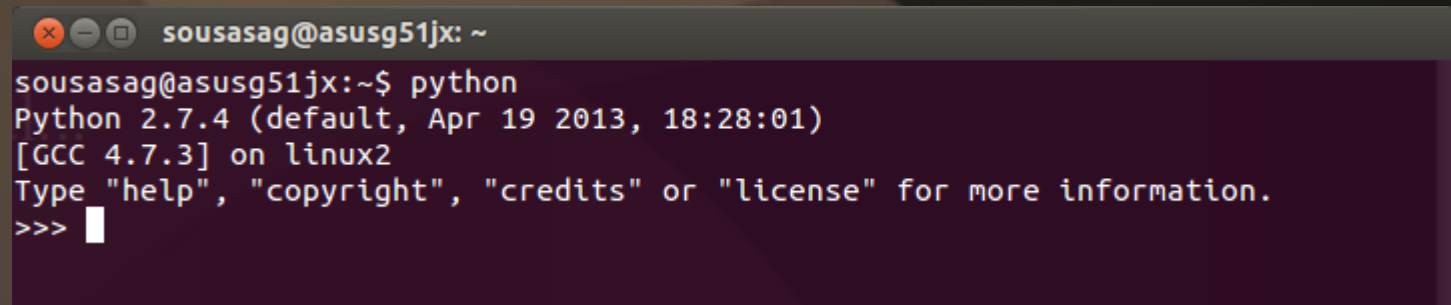
Lesson 1: Python basics

- Starting Python
- Syntax & Basic data types
- Input and Output - Files
- Scripting & Modules
- Common Syntax Structures
- Standard Libraries

Starting Python

Python:

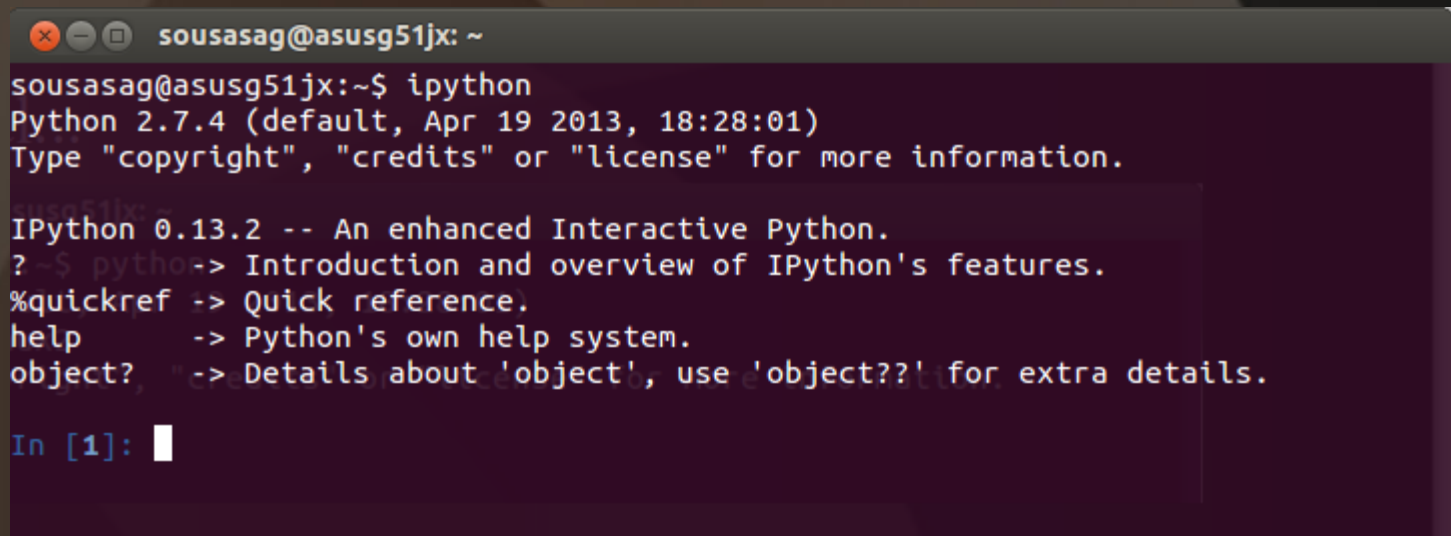
type python in terminal...

A terminal window with a dark purple background and white text. The window title is 'sousasag@asusg51jx: ~'. The user has entered 'python' at the prompt. The output shows 'Python 2.7.4 (default, Apr 19 2013, 18:28:01)' and '[GCC 4.7.3] on linux2'. It also displays instructions: 'Type "help", "copyright", "credits" or "license" for more information.' and the interactive prompt '>>>' with a cursor.

```
sousasag@asusg51jx: ~  
sousasag@asusg51jx:~$ python  
Python 2.7.4 (default, Apr 19 2013, 18:28:01)  
[GCC 4.7.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

IPython:

type ipython in terminal...

A terminal window with a dark purple background and white text. The window title is 'sousasag@asusg51jx: ~'. The user has entered 'ipython' at the prompt. The output shows 'Python 2.7.4 (default, Apr 19 2013, 18:28:01)' and 'Type "copyright", "credits" or "license" for more information.' followed by 'IPython 0.13.2 -- An enhanced Interactive Python.' and a list of shortcuts: '? -> Introduction and overview of IPython's features.', '%quickref -> Quick reference.', 'help -> Python's own help system.', and 'object? -> Details about 'object'', use 'object??' for extra details.' The prompt 'In [1]:' is shown with a cursor.

```
sousasag@asusg51jx: ~  
sousasag@asusg51jx:~$ ipython  
Python 2.7.4 (default, Apr 19 2013, 18:28:01)  
Type "copyright", "credits" or "license" for more information.  
  
IPython 0.13.2 -- 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]: █
```

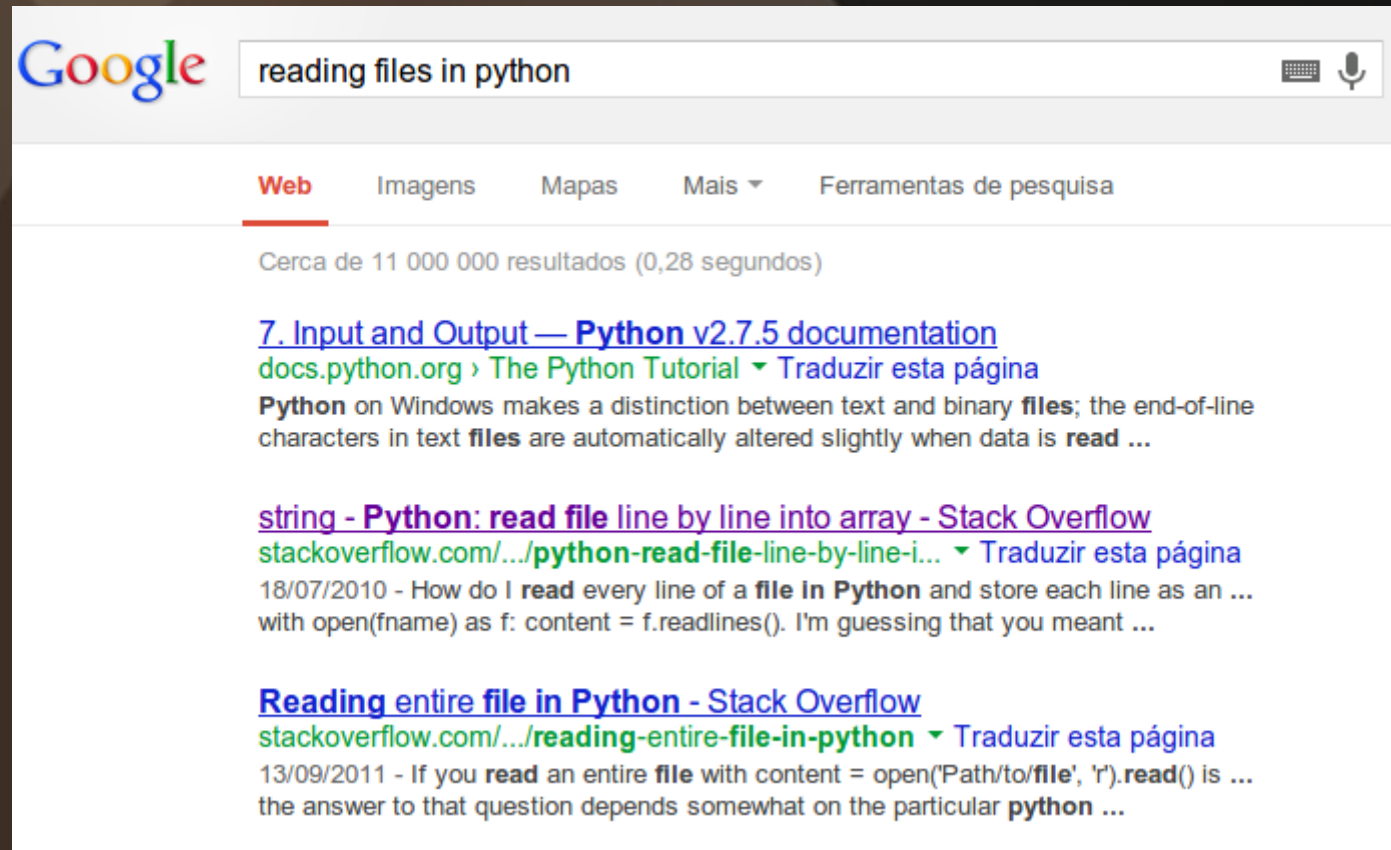
Python finding help

The geeky way:

Interactive Help in Python Shell

<code>help()</code>	Invoke interactive help
<code>help(<i>m</i>)</code>	Display help for module <i>m</i>
<code>help(<i>f</i>)</code>	Display help for function <i>f</i>
<code>dir(<i>m</i>)</code>	Display names in module <i>m</i>

The easy way:



The screenshot shows a Google search interface with the query "reading files in python". The search results are displayed under the "Web" tab. The first result is "7. Input and Output — Python v2.7.5 documentation" from docs.python.org. The second result is "string - Python: read file line by line into array - Stack Overflow" from stackoverflow.com, dated 18/07/2010. The third result is "Reading entire file in Python - Stack Overflow" from stackoverflow.com, dated 13/09/2011. The search results are in Portuguese, with links to "Traduzir esta página" (Translate this page).

Google reading files in python

Web Imagens Mapas Mais Ferramentas de pesquisa

Cerca de 11 000 000 resultados (0,28 segundos)

[7. Input and Output — Python v2.7.5 documentation](#)
docs.python.org > The Python Tutorial > Traduzir esta página
Python on Windows makes a distinction between text and binary **files**; the end-of-line characters in text **files** are automatically altered slightly when data is **read** ...

[string - Python: read file line by line into array - Stack Overflow](#)
stackoverflow.com/.../python-read-file-line-by-line-i... > Traduzir esta página
18/07/2010 - How do I **read** every line of a **file** in **Python** and store each line as an ... with open(fname) as f: content = f.readlines(). I'm guessing that you meant ...

[Reading entire file in Python - Stack Overflow](#)
stackoverflow.com/.../reading-entire-file-in-python > Traduzir esta página
13/09/2011 - If you **read** an entire **file** with content = open("Path/to/file", 'r').read() is ... the answer to that question depends somewhat on the particular **python** ...

Python importing modules

Module Import

```
import module_name  
from module_name import name , ...  
from module_name import *
```

CAUTION

Interactive Help in Python Shell

help()	Invoke interactive help
help(<i>m</i>)	Display help for module <i>m</i>
help(<i>f</i>)	Display help for function <i>f</i>
dir(<i>m</i>)	Display names in module <i>m</i>

Python Sintaxe and Basic Types

Python as a calculator

Using Python as a Calculator

Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J

```
>>> 2 + 5
7
>>> 2.1*(2**3-1/2.3)
15.886956521739132
>>> █
```

**	Exponentiation
*, /, %	Multiply, divide, mod
+, -	Add, subtract

Python Sintaxe and Basic Types

Python as a calculator

Using Python as an advanced Calculator

Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J

**	Exponentiation
----	----------------

*, /, %	Multiply, divide, mod
---------	-----------------------

+, -	Add, subtract
------	---------------

Common Math Module Functions

Function	Returns (all float)
ceil(x)	Smallest whole nbr $\geq x$
cos(x)	Cosine of x radians
degrees(x)	x radians in degrees
radians(x)	x degrees in radians
exp(x)	$e ** x$
floor(x)	Largest whole nbr $\leq x$
hypot(x, y)	$\text{sqrt}(x * x + y * y)$
log(x [, base])	Log of x to <i>base</i> or natural log if <i>base</i> not given
pow(x, y)	$x ** y$
sin(x)	Sine of x radians
sqrt(x)	Positive square root of x
tan(x)	Tangent of x radians
pi	Math constant pi to 15 sig figs
e	Math constant e to 15 sig figs

```
Python 2.7.4 (default, Apr 19 2013, 18:28:01)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> from math import *
```

```
>>> dir()
```

```
['__builtin__', '__doc__', '__name__', '__package__',
'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh',
'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e',
'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinfinite',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf',
'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh',
'trunc']
```

```
>>> cos(radians(60))**2+sin(radians(60))**2
```

```
1.0
```

Python Sintaxe and Basic Types

Small Operator Precedence Table

<i>func_name(args, ...)</i>	Function call
<i>x[index : index]</i>	Slicing
<i>x[index]</i>	Indexing
<i>x.attribute</i>	Attribute reference
**	Exponentiation
*, /, %	Multiply, divide, mod
+, -	Add, subtract
>, <, <=, >=, !=, ==	Comparison
in, not in	Membership tests
not, and, or	Boolean operators NOT, AND, OR

Python Sintaxe and Basic Types

Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J
bool	Boolean	True, False
str	Character sequence	"Python"
tuple	Immutable sequence	(2, 4, 7)
list	Mutable sequence	[2, x, 3.1]
dict	Mapping	{ x:2, y:5 }

Special Note: 32bit vs 64bit systems have different variable sizes:

max int for 32bit: $2^{31}-1$ for 64bit: $2^{63}-1$

you can check this with `sys.maxint`

Python Sintaxe and Basic Types

Strings

And more:

- **strip()** - Return a copy of the string *s* with leading and trailing whitespace removed.

Other useful functions:

- **atof(s)** - Return the floating point number represented by the string *s*.
- **atoi(s)** - Return the int number represented by the string *s*.
- **atol(s)** - Return the long number represented by the string *s*.

Alternatively you can use casting of variables:

```
>>> from string import *
>>> s='1.23'
>>> df=atof(s)
>>> type(df)
<type 'float'>
>>> df2=float(s)
>>> type(df2)
<type 'float'>
>>> print df,df2
1.23 1.23
```

Common String Methods

S.method()	Returns (str unless noted)
capitalize	S with first char uppercase
center(w)	S centered in str w chars wide
count(sub)	int nbr of non-overlapping occurrences of <i>sub</i> in <i>S</i>
find(sub)	int index of first occurrence of <i>sub</i> in <i>S</i> or -1 if not found
isdigit()	bool True if <i>S</i> is all digit chars, False otherwise
islower() isupper()	bool True if <i>S</i> is all lower/upper case chars, False otherwise
join(seq)	All items in <i>seq</i> concatenated into a str, delimited by <i>S</i>
lower() upper()	Lower/upper case copy of <i>S</i>
lstrip() rstrip()	Copy of <i>S</i> with leading/ trailing whitespace removed, or both
split([sep])	List of tokens in <i>S</i> , delimited by <i>sep</i> ; if <i>sep</i> not given, delimiter is any whitespace

Python Sintaxe and Basic Types

Strings – Formating numbers

```
>>> x=123;y=456.789
>>> s='%6d' % x
>>> s
'   123'
>>> '%06d' % x
'000123'
>>> '%8.2f' % y
'  456.79'
>>> '%8.2e' % y
'4.57e+02'
```

Formatting Numbers as Strings

Syntax: *"format_spec" % numeric_exp*

***format_spec* syntax:** *% width.precision type*

- *width* (optional): align in number of columns specified; negative to left-align, precede with 0 to zero-fill
- *precision* (optional): show specified digits of precision for floats; 6 is default
- *type* (required): d (decimal int), f (float), s (string), e (float – exponential notation)
- Examples for $x = 123$, $y = 456.789$
 - "%6d" % x -> ... 123
 - "%06d" % x -> 000123
 - "%8.2f" % y -> .. 456.79
 - "8.2e" % y -> 4.57e+02
 - "-8s" % "Hello" -> Hello ...

Python Sintaxe and Basic Types

Lists

list	Mutable sequence	[2, x, 3.1]
------	------------------	-------------

```
>>> list = [1,2,3,4,5,6,7]
>>> list[1]
2
>>> a.pop(0)
1
>>> list [-1]
7
>>> list[3:5]
[4, 5]
>>> list[4:]
[5, 6, 7]
>>> list[: -3]
[1, 2, 3, 4]
```

```
>>> a
['Primeiro', 'a']
>>> a=[1,'a',2L]
>>> a.insert(0,'Primeiro')
>>> a
['Primeiro', 1, 'a', 2L]
>>> ele=a.pop()
>>> ele
2L
>>> a
['Primeiro', 1, 'a']
>>> a.reverse()
>>> a
['a', 1, 'Primeiro']
>>> a.sort()
>>> a
[1, 'Primeiro', 'a']
>>> a.pop(0)
1
>>> a
['Primeiro', 'a']
```

```
>>> a=['Helton','Jackson','Moutinho','Iturbe']
>>> a.sort()
>>> a
['Helton', 'Iturbe', 'Jackson', 'Moutinho']
>>> a.sort(reverse=True)
>>> a
['Moutinho', 'Jackson', 'Iturbe', 'Helton']
```

Common List Methods

L.method()	Result/Returns
append(<i>obj</i>)	Append <i>obj</i> to end of <i>L</i>
count(<i>obj</i>)	Returns int nbr of occurrences of <i>obj</i> in <i>L</i>
index(<i>obj</i>)	Returns index of first occurrence of <i>obj</i> in <i>L</i> ; raises ValueError if <i>obj</i> not in <i>L</i>
pop([<i>index</i>])	Returns item at specified <i>index</i> or item at end of <i>L</i> if <i>index</i> not given; raises IndexError if <i>L</i> is empty or <i>index</i> is out of range
remove(<i>obj</i>)	Removes first occurrence of <i>obj</i> from <i>L</i> ; raises ValueError if <i>obj</i> is not in <i>L</i>
reverse()	Reverses <i>L</i> in place
sort()	Sorts <i>L</i> in place

Python Sintaxe and Basic Types

Tuples

tuple	Immutable sequence	(2, 4, 7)
-------	--------------------	-----------

Common Tuple Methods

<i>T.method()</i>	Returns
count(<i>obj</i>)	Returns nbr of occurrences of <i>obj</i> in <i>T</i>
index(<i>obj</i>)	Returns index of first occurrence of <i>obj</i> in <i>T</i> ; raises ValueError if <i>obj</i> is not in <i>T</i>

Tuples vs. Lists

Tuples are immutable, what you define you cannot change it.

Lists is a dynamic data structure, you can add, remove, change elements.

Tuples are faster than Lists

Python Sintaxe and Basic Types

Dictionary

dict	Mapping	{ x:2, y:5 }
------	---------	--------------

```
>>> dict={'Helton':1,'Jackson':9,'Lucho':3}
>>> dict.keys()
['Helton', 'Lucho', 'Jackson']
>>> dict.values()
[1, 3, 9]
>>> for key in dict:
...     print key,dict[key]
...
Helton 1
Lucho 3
Jackson 9
```

Common Dictionary Methods

<i>D</i>.method()	Result/Returns
clear()	Remove all items from <i>D</i>
get(<i>k</i> [, <i>val</i>])	Return <i>D</i> [<i>k</i>] if <i>k</i> in <i>D</i> , else <i>val</i>
has_key(<i>k</i>)	Return True if <i>k</i> in <i>D</i> , else False
items()	Return list of key-value pairs in <i>D</i> ; each list item is 2-item tuple
keys()	Return list of <i>D</i> 's keys
pop(<i>k</i> , [<i>val</i>])	Remove key <i>k</i> , return mapped value or <i>val</i> if <i>k</i> not in <i>D</i>
values()	Return list of <i>D</i> 's values

Python Sintaxe and Basic Types

Common Built-in Functions

```
>>> list=range(0,8,2)
>>> list
[0, 2, 4, 6]
>>> len(list)
4
>>> max(list)
6
>>> min(list)
0
>>> sum(list)
12
>>> type(list)
<type 'list'>
>>> str='1.234'
>>> num=float(str)
>>> num
1.234
>>> round(num,2)
1.23_
```

Common Built-in Functions

Function	Returns
abs(x)	Absolute value of x
dict()	Empty dictionary, eg: d = dict()
float(x)	int or string x as float
id(obj)	memory addr of <i>obj</i>
int (x)	float or string x as int
len(s)	Number of items in sequence s
list()	Empty list, eg: m = list()
max(s)	Maximum value of items in s
min(s)	Minimum value of items in s
open(f)	Open filename <i>f</i> for input
ord(c)	ASCII code of <i>c</i>
pow(x,y)	$x ** y$
range(x)	A list of x ints 0 to x - 1
round(x,n)	float x rounded to <i>n</i> places
str(obj)	str representation of <i>obj</i>
sum(s)	Sum of numeric sequence s
tuple(items)	tuple of <i>items</i>
type(obj)	Data type of <i>obj</i>

Input – Output - Files

```
open(...)  
open(name[, mode[, buffering]]) -> file object
```

```
>>> file = open('file.tmp','r')  
>>> lines_list=file.readlines()  
>>> lines_list  
['line 1\n', 'before an empty line\n', '\n', 'final line\n']  
>>> file.close()  
>>> fileout = open('output.file','w')  
>>> fileout.write("First Line to fine\n")  
>>> fileout.write("Last Line to file\n")  
>>> fileout.close()  
>>> fileap = open('file.tmp', 'a')  
>>> fileap.write("Extra line to file\n")  
>>> fileap.close()  
>>>
```

```
sousasag@asusg51jx:~$ more file.tmp  
line 1  
before an empty line
```

```
final line  
Extra line to file
```

```
sousasag@asusg51jx:~$
```

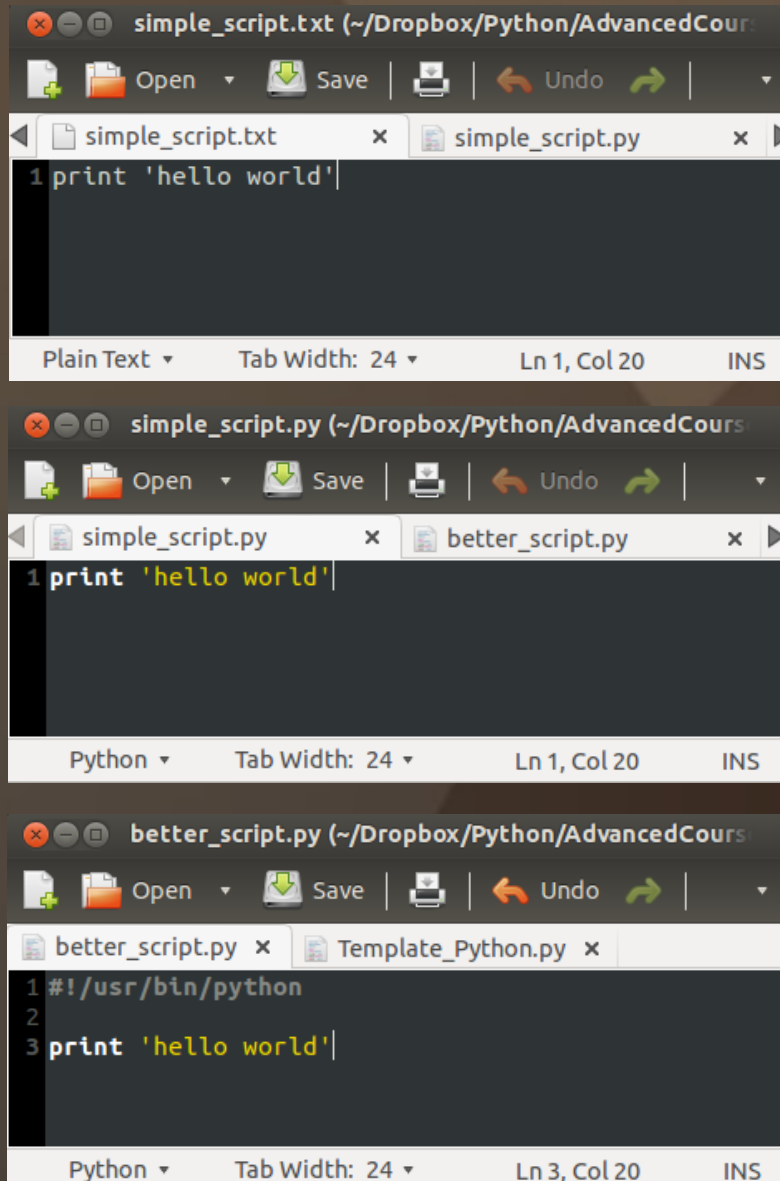
File modes

- Read-only: r
- Write-only: w
 - Note: Create a new file or *overwrite* existing file.
- Append a file: a
- Read and Write: r+
- Binary mode: b
 - Note: Use for binary files, especially on Windows.

Common File Methods

<i>F</i> .method()	Result/Returns
read([<i>n</i>])	Return str of next <i>n</i> chars from <i>F</i> , or up to EOF if <i>n</i> not given
readline([<i>n</i>])	Return str up to next newline, or at most <i>n</i> chars if specified
readlines()	Return list of all lines in <i>F</i> , where each item is a line
write(<i>s</i>)	Write str <i>s</i> to <i>F</i>
writelines(<i>L</i>)	Write all str in seq <i>L</i> to <i>F</i>
close()	Closes the file

Scripting in Python



```
simple_script.txt (~/.Dropbox/Python/AdvancedCourse)
1 print 'hello world'

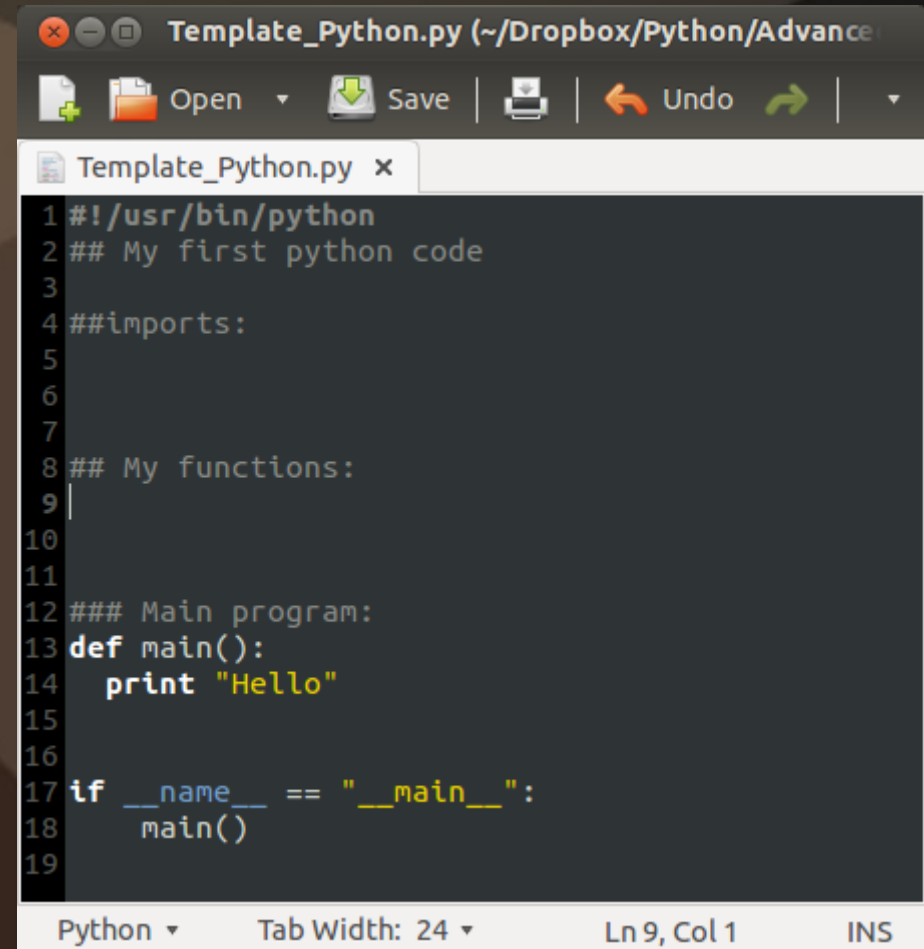
Plain Text Tab Width: 24 Ln 1, Col 20 INS

simple_script.py (~/.Dropbox/Python/AdvancedCourse)
1 print 'hello world'

Python Tab Width: 24 Ln 1, Col 20 INS

better_script.py (~/.Dropbox/Python/AdvancedCourse)
1 #!/usr/bin/python
2
3 print 'hello world'

Python Tab Width: 24 Ln 3, Col 20 INS
```



```
Template_Python.py (~/.Dropbox/Python/AdvancedCourse)
1 #!/usr/bin/python
2 ## My first python code
3
4 ##imports:
5
6
7
8 ## My functions:
9
10
11
12 ### Main program:
13 def main():
14     print "Hello"
15
16
17 if __name__ == "__main__":
18     main()
19

Python Tab Width: 24 Ln 9, Col 1 INS
```

To run the script/program:

- python filename
- ./filename (given exec permission)

Python Sintaxe

Common Syntax Structures

Assignment Statement

```
var = exp
```

Console Input/Output

```
var = input( [prompt] )
```

```
var = raw_input( [prompt] )
```

```
print exp[,] ...
```

Selection

```
if (boolean_exp):
```

```
    stmt ...
```

```
[elif (boolean_exp):
```

```
    stmt ...] ...
```

```
[else:
```

```
    stmt ...]
```

Repetition

```
while (boolean_exp):
```

```
    stmt ...
```

Traversal

```
for var in traversable_object:
```

```
    stmt ...
```

```
for i in range(init,end,step):
```

```
    stmt ...
```

Indentation define the blocks

(no brackets needed)

Function Call

```
function_name( arguments )
```

Class Definition

```
class Class_name [ (super_class) ]:
```

```
    [ class variables ]
```

```
    def method_name( self, parameters ):
```

```
        stmt
```

Comprehension Lists sintaxe:

```
new_list = [expression(i) for i in old_list if filter(i)]
```

```
>>> list = [1,2,3,4,-1,2,-3,-4.3]
>>> list
[1, 2, 3, 4, -1, 2, -3, -4.3]
>>> listnegatives = [ val for val in list if val < 0 ]
>>> listnegatives
[-1, -3, -4.3]
```

Standard Libraries

os - Miscellaneous operating system interfaces

dir(os) or <http://docs.python.org/2/library/os>

Some examples:

```
>>> import os
>>> os.getcwd()
'/home/sousasag/Dropbox/Python/AdvancedCourse/examples'
>>> os.listdir(os.curdir)
['Template_Python.py', 'simple_script.py', 'better_script.py', 'simple_script.txt', 'tmpdir']
>>> os.mkdir('junkdir')
>>> os.rename('junkdir', 'tmpdir2')
>>> os.listdir(os.curdir)
['Template_Python.py', 'simple_script.py', 'tmpdir2', 'better_script.py', 'simple_script.txt', 'tmpdir']
>>> 'file.tmp' in os.listdir(os.curdir)
False
>>> a = os.path.abspath('simple_script.py')
>>> a
'/home/sousasag/Dropbox/Python/AdvancedCourse/examples/simple_script.py'
>>> os.path.split(a)
('/home/sousasag/Dropbox/Python/AdvancedCourse/examples', 'simple_script.py')
>>> os.path.dirname(a)
'/home/sousasag/Dropbox/Python/AdvancedCourse/examples'
>>> os.path.basename(a)
'simple_script.py'
>>> os.path.isfile(a)
True
>>> os.path.isfile(os.path.basename(a))
True
>>> os.path.isfile(tmpdir)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'tmpdir' is not defined
>>> os.path.isfile('tmpdir')
False
```

Standard Libraries

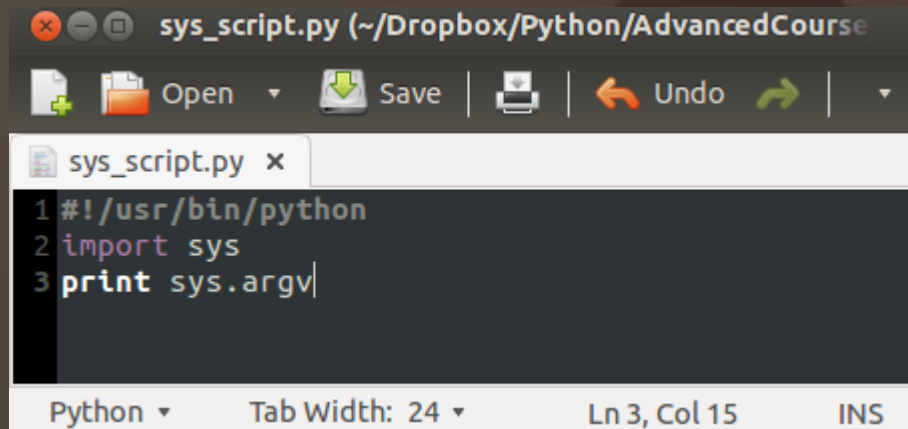
glob - Filename globbing utility

dir(glob) or <http://docs.python.org/library/glob>

```
>>> import glob
>>> glob.glob('*.py')
['Template_Python.py', 'simple_script.py', 'better_script.py']
```

sys – System information

dir(sys) - <http://docs.python.org/library/sys>

A screenshot of a text editor window titled 'sys_script.py (~/.Dropbox/Python/AdvancedCourse)'. The editor has a menu bar with 'Open', 'Save', 'Undo', and 'Redo' options. The code in the editor is:

```
1#!/usr/bin/python
2import sys
3print sys.argv
```

The status bar at the bottom shows 'Python', 'Tab Width: 24', 'Ln 3, Col 15', and 'INS'.

```
>>> sys.maxint
9223372036854775807
>>> sys.platform
'linux2'
>>> sys.version
'2.7.4 (default, Apr 19 2013, 18:28:01) \n[GCC 4.7.3]'
```

```
sousasag@asusg51jx:~/Dropbox/Python/AdvancedCourse/examples$ ./sys_script.py
['./sys_script.py']
sousasag@asusg51jx:~/Dropbox/Python/AdvancedCourse/examples$ ./sys_script.py a b 2 3.42
['./sys_script.py', 'a', 'b', '2', '3.42']
```

Standard Libraries

pickle - Create portable serialized representations of Python objects.
(similar to IDL save and restore)

dir(pickle)

or

<http://docs.python.org/library/pickle>

Some examples:

```
import pickle
pickle.dump(obj,file)
obj=pickle.load(file)
```

```
sousasag@asusg51jx:~/Dropbox/Python/AdvancedCourse/examples$ python
Python 2.7.4 (default, Apr 19 2013, 18:28:01)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pickle
>>> list=["El", 'terrorista']
>>> dic={"TimTim": 223421232, "Mickey":221232546}
>>> tuple=(list,dic)
>>> tuple
(['El', 'terrorista'], {'Mickey': 221232546, 'TimTim': 223421232})
>>> pickle.dump(tuple, file("filedump.pkl",'w'))
... )
>>> import pickle
>>> list=["El", 'terrorista']
>>> dic={"TimTim": 223421232, "Mickey":221232546}
>>> tuple=(list,dic)
>>> tuple
(['El', 'terrorista'], {'Mickey': 221232546, 'TimTim': 223421232})
>>> pickle.dump(tuple, file("filedump.pkl",'w'))
>>>
sousasag@asusg51jx:~/Dropbox/Python/AdvancedCourse/examples$ python
Python 2.7.4 (default, Apr 19 2013, 18:28:01)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> l
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'l' is not defined
>>> import pickle
>>> pickle.load(file("filedump.pkl"))
(['El', 'terrorista'], {'Mickey': 221232546, 'TimTim': 223421232})
>>> tuple=pickle.load(file("filedump.pkl"))
>>> (list,dic)=tuple
>>> list
['El', 'terrorista']
>>> dic
{'Mickey': 221232546, 'TimTim': 223421232}
```



Questions?



Exercises...