 python™
The Language

A bit of history

- ▶ Python was conceived in 1980 and started being implemented in 1989 by Guido van Rossum at CWI (Netherlands), entitled BDFL (Benevolent Dictator For Life) by the community
- ▶ Focus on code readability, providing a syntax that allows more concepts in fewer lines of code
- ▶ Fully open-source with a extremely active and wide community



A bit of history – Versions

The first main community backed version:

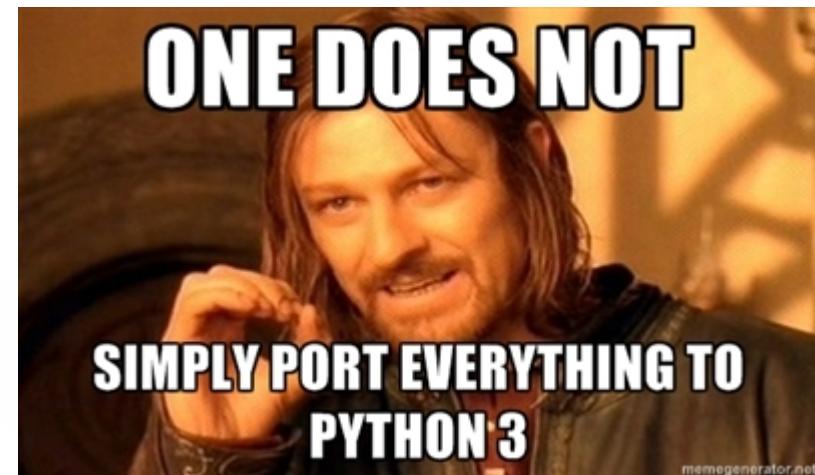
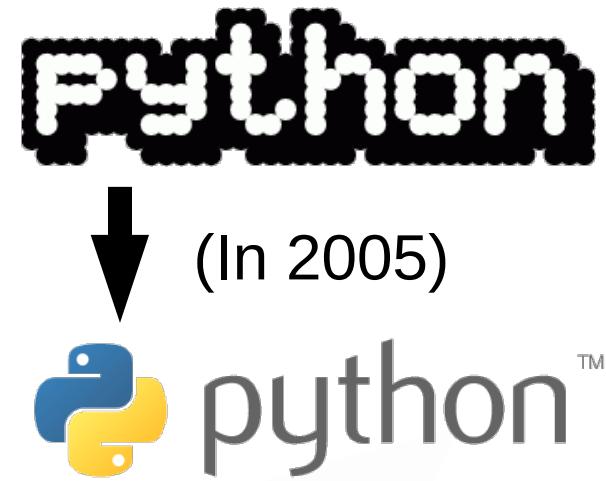
- ▶ Python 2.0, October 16, 2000
 - ▶ Cycle-detecting garbage collector for memory management
 - ▶ Unicode support



A bit of history – Versions

The first main community backed version:

- ▶ Python 2.0, October 16, 2000
 - ▶ Cycle-detecting garbage collector for memory management
 - ▶ Unicode support
- ▶ Python 3.0, December 3, 2008
 - ▶ Backwards-**incompatible**
 - ▶ Major features backported to python 2.6 & 2.7

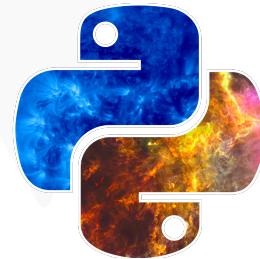


A bit of history – Use of python

Python is currently being used for “everything”



Large-scale servers
working 24/7



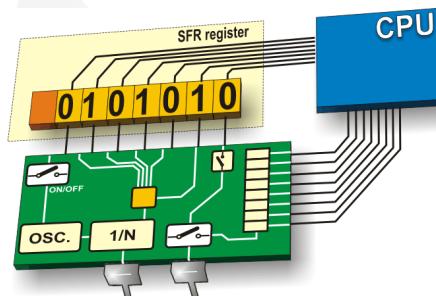
Astronomy



Python 1



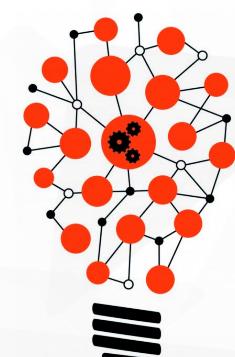
Throw-away
scripts



Microcontrollers



Databases



Machine
learning

A bit of history – Use of python

(Some) Companies currently massively using python

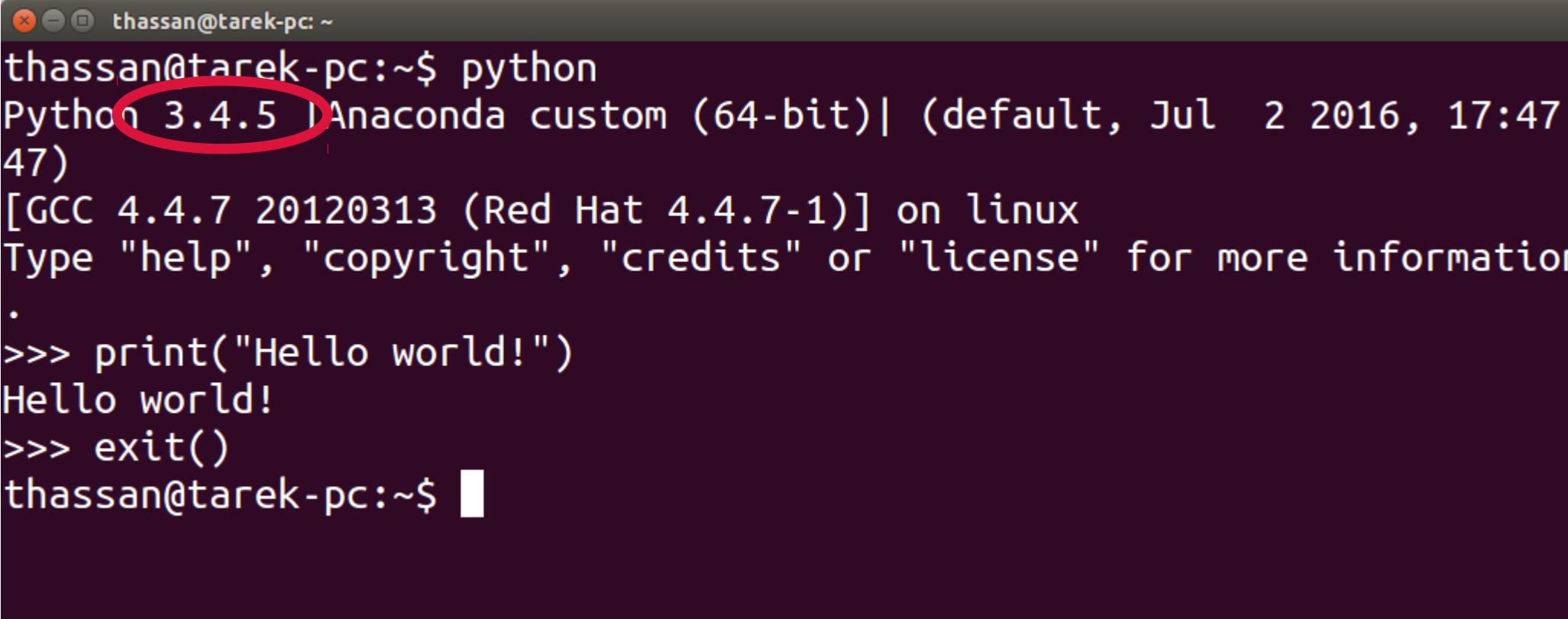


A bit of history – Python 3

- ▶ Python 3 was designed to correct some fundamental design flaws
(see https://en.wikipedia.org/wiki/History_of_Python#Features)
 - ▶ Not possible to retain full backwards compatibility
- ▶ What does this mean?
 - ▶ If you start now with python 3, you will learn the “good way”
 - ▶ Some packages may still be not fully functional in python 3
(give them some time!)
 - ▶ You can check the version of python you are using simply by:
import sys; print(sys.version)

Getting started with python

- ▼ Open a terminal and type “python”



```
thassan@tarek-pc:~$ python
Python 3.4.5 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:47:47)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information
.
>>> print("Hello world!")
Hello world!
>>> exit()
thassan@tarek-pc:~$
```

Getting started with python

- ▼ To use an external library, you need to “import” it:

```
thassan@tarek-pc:~$ python
Python 3.4.5 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:47:47)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> print (math.pi)
3.141592653589793
>>> █
```

- ▼ Python standard library: <https://docs.python.org/3/library/index.html>

Getting started with python

- ▼ You have more interactive options, like “ipython”:

```
IPython: home/thassan
thassan@tarek-pc:~$ ipython
Python 3.4.5 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:47:47)
Type "copyright", "credits" or "license" for more information.

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]: import math

In [2]: math.p[  
        math.pi  
        math.pow
```

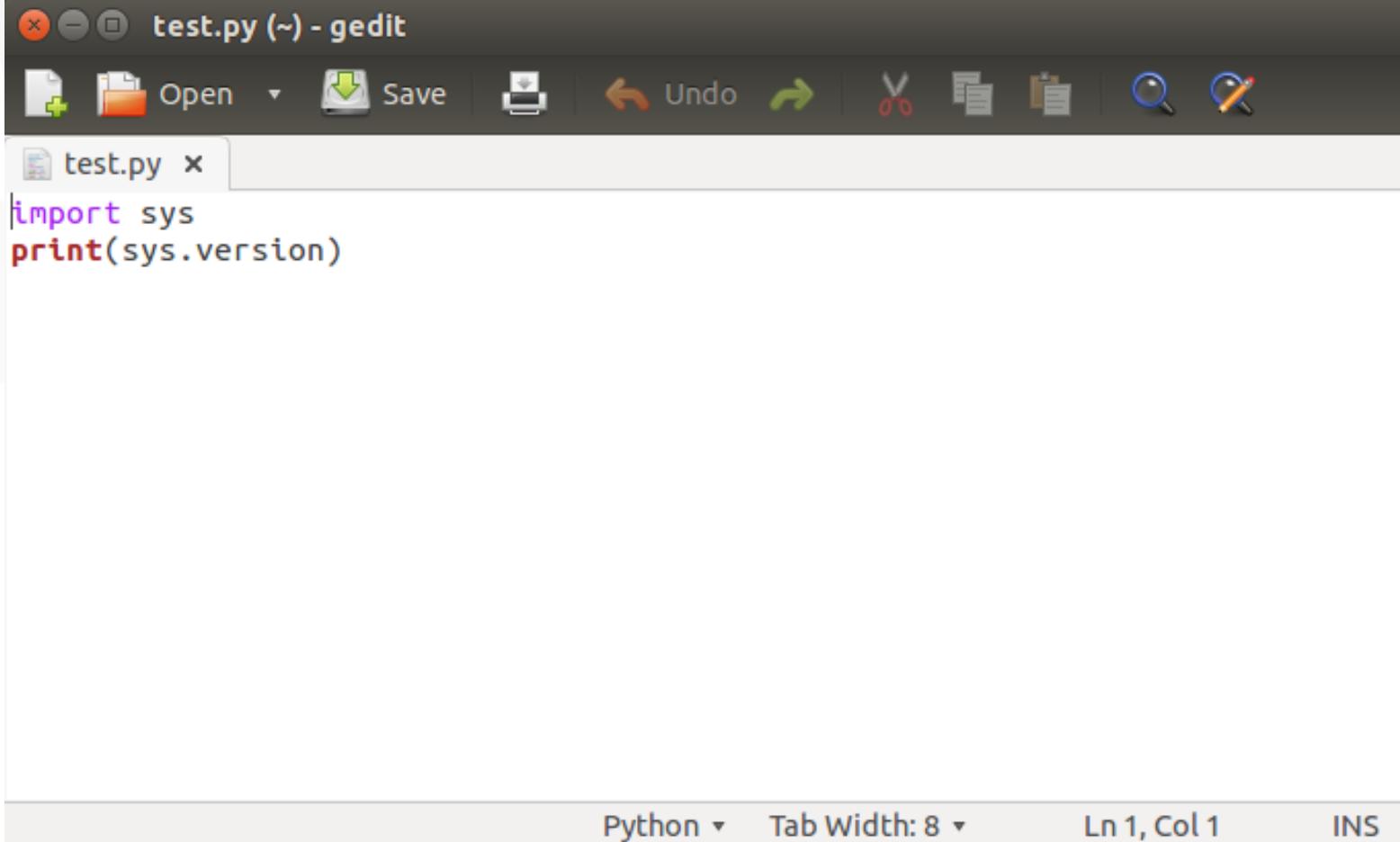
Getting started with python

- ▼ For writing scripts, you can use any “plain” text editor:

```
thassan@tarek-pc:~$ vim test.py
thassan@tarek-pc:~$ cat test.py
import sys
print(sys.version)
thassan@tarek-pc:~$ python test.py
3.4.5 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:47:47)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
thassan@tarek-pc:~$ █
```

Getting started with python

- For writing scripts, you can use any “plain” text editor:



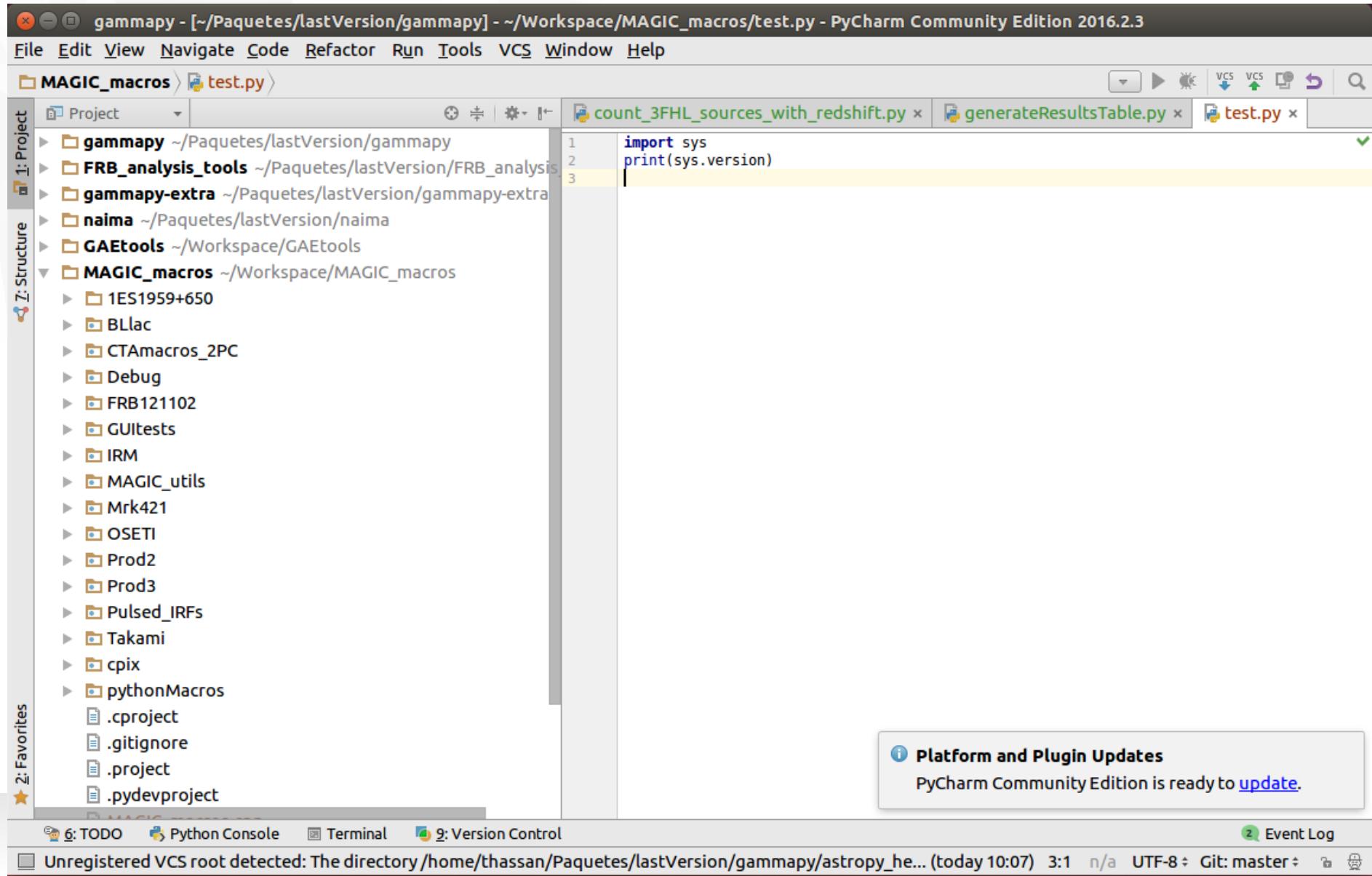
The screenshot shows the gedit text editor interface. The title bar reads "test.py (~) - gedit". The toolbar includes standard icons for file operations (Open, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace). The main window displays the code for a Python script:

```
import sys
print(sys.version)
```

The status bar at the bottom shows "Python" and "Tab Width: 8". The cursor position is "Ln 1, Col 1" and the insertion mode is "INS".

Getting started with python

- Or use an IDE: (e. g. pyCharm, eclipse, vim, wing, spyder...)



How python works: binaries

- As it is possible (and usual) to have several “pythons” installed, some clarifications are useful
 - When you enter the python console, you execute a python binary (you may have several installed!)

```
thassan@tarek-pc:~$ which python
/home/thassan/Paquetes/anaconda3/bin/python
thassan@tarek-pc:~$ python
Python 3.4.5 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:47:47)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
thassan@tarek-pc:~$ /usr/bin/python
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

How python works: binaries & libraries

- ▼ Certain libraries may be installed only in some python environments:

The image shows two terminal windows side-by-side. Both windows have a dark background and white text. The left window shows a Python environment where the user has run the command to find the path to the Python binary. The right window shows a standard Python 2.7.6 environment where the user attempts to import a module named 'gammappy' but receives an ImportError.

```
thassan@tarek-pc:~$ /home/thassan/Paquetes/anaconda3/envs/cta/bin/python
Python 3.5.1 |Continuum Analytics, Inc.| (default, Jun 15 2016, 15:32:45)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> import gammappy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named gammappy
>>>
```

How python works: binaries & libraries

- ▼ To install a library:

- ▼ You may do it manually (usually not encouraged):

- ▼ Download package, “compile” and install

```
thassan@tarek-pc:~$ cd Paquetes/lastVersion/naima/
thassan@tarek-pc:~/Paquetes/lastVersion/naima$ python setup.py install
Initializing astropy_helpers submodule with: `git submodule update --init -- astropy_helpers'
/home/thassan/Paquetes/anaconda3/lib/python3.4/site-packages/Cython/Distutils/old_build_ext.py:30: UserWarning: Cython.Distutils.old_build_ext does not properly handle dependencies and is deprecated.
  "Cython.Distutils.old_build_ext does not properly handle dependencies "
Freezing version number to naima/version.py
running install
running bdist_egg
running egg_info
creating naima.egg-info
writing pbr to naima.egg-info/pbr.json
writing top-level names to naima.egg-info/top_level.txt
writing requirements to naima.egg-info/requirements.txt
writing naima.egg-info/PKG-INFO
writing dependency_links to naima.egg-info/dependency_links.txt
writing manifest file 'naima.egg-info/SOURCES.txt'
```

How python works: binaries & libraries

- ▶ To install a library:
 - ▶ You may do it manually (usually not encouraged):
 - ▶ Download package, “compile” and install
 - ▶ You may use a package manager such as “pip”:
 - ▶ “`pip install <package_name>`”
 - ▶ This will install all dependencies, in the version required by the package
 - ▶ Unless if you are a developer of a specific package, you will probably always use this method (or analog ones)

How python works: binaries & libraries

► Working with environments:

- If you followed the tutorial we sent to install python, you installed **anaconda** and created an environment named “py36”
- These virtual environments allow to have a clean installation of python where all third-party packages are in the required version
 - Different python project may (and probably will) require different dependencies (either different libraries, or same libraries with different versions)
 - Usually, you will just need to activate the desired environments: “source activate py36”

<https://conda.io/docs/user-guide/tasks/manage-environments.html>

Using Jupyter notebooks

- ▶ For simplicity, to follow the tutorials of this course we will be using Jupyter notebooks
 - ▶ Open-source web application that allows to edit and execute code
 - ▶ All code is executed locally within your python environment (even if it is shown within your browser)
- ▶ To begin the course, clone the gitHub repository of the course and execute the jupyter notebook “02_01_TheLanguage.ipynb”



Using Jupyter notebooks

- ▼ To begin the course (**if you did not do it already**), clone the gitHub repository of the course and execute the jupyter notebook “02_01_TheLanguage.ipynb”

```
thassan@tarek-pc:~$ cd /tmp/
thassan@tarek-pc:/tmp$ git clone https://github.com/Python4AstronomersAndParticlePhysicists/PythonWorkshop-ICE.git
Cloning into 'PythonWorkshop-ICE'...
remote: Counting objects: 503, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 503 (delta 13), reused 23 (delta 9), pack-reused 474
Receiving objects: 100% (503/503), 113.19 MiB | 2.72 MiB/s, done.
Resolving deltas: 100% (281/281), done.
Checking connectivity... done.
thassan@tarek-pc:/tmp$ cd PythonWorkshop-ICE/
thassan@tarek-pc:/tmp/PythonWorkshop-ICE$ jupyter notebooks/02_01_TheLanguage.ipynb
```

Using Jupyter notebooks

- ▶ To begin the course (if you did not do it already), clone the gitHub repository of the course and execute the jupyter notebook “02_01_TheLanguage.ipynb”
- ▶ If you already cloned it, make sure that you have the latest version by typing within your project “git pull”:

```
thassan@tarek-pc:~/Paquetes/lastVersion/PythonWorkshop-ICE$ git pull
remote: Counting objects: 16, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 16 (delta 8), reused 10 (delta 5), pack-reused 0
Unpacking objects: 100% (16/16), done.
From https://github.com/Python4AstronomersAndParticlePhysicists/PythonWorkshop-ICE
  2b24994..bfd2b69  master      -> origin/master
Updating 2b24994..bfd2b69
Fast-forward
 README.md                  |  11 +- 
 examples/use_system_latex/plot.py |   7 +- 
 examples/use_system_latex/proceeding.tex |   4 +- 
 notebooks/05_01_matplotlib.ipynb | 719 ++++++----- 
 4 files changed, 179 insertions(+), 562 deletions(-)
thassan@tarek-pc:~/Paquetes/lastVersion/PythonWorkshop-ICE$ █
```

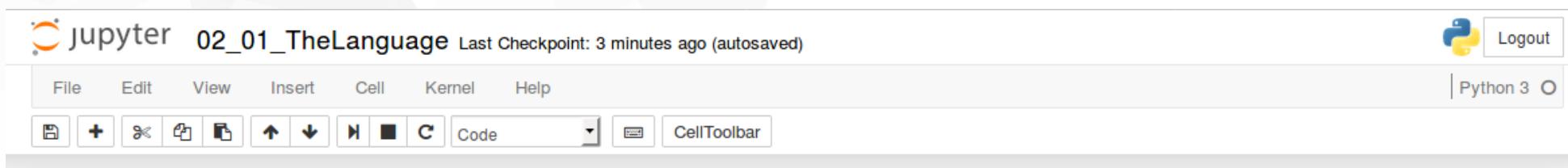
Using Jupyter notebooks

- ▶ To begin the course (if you did not do it already), clone the gitHub repository of the course and execute the jupyter notebook “02_01_TheLanguage.ipynb”
- ▶ If you already cloned it, make sure that you have the latest version by typing within your project “git pull”:
- ▶ If conflicts appear while updating (probably caused by local modifications you did to the notebooks), you may “stash” your changes and try to update again:

```
> git stash  
> git pull
```

Using Jupyter notebooks

- From now on, the tutorial (and whole workshop) continues through notebooks (in your browser, similar to this):



The language (1h)

- 1) Short historical overview (5m)
- 2) Briefly talk about python 2/3 (5m)
- 3) Getting started with pyhton (15m)
 - Python terminal
 - iPython
 - Script with plain editors
 - Scripts with pycharm
- 4) How python works (20m)
 - binaries, libraries and environments
 - exercises/examples
 - example: import library included
 - example: import library not installed
 - exercice: install library manually
 - example: install library pip
- 5) Explain/show what jupyter notebooks are (15m)
 - Code is running locally
 - Typical shortcuts
 - Is possible to add comments, etc...