 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:

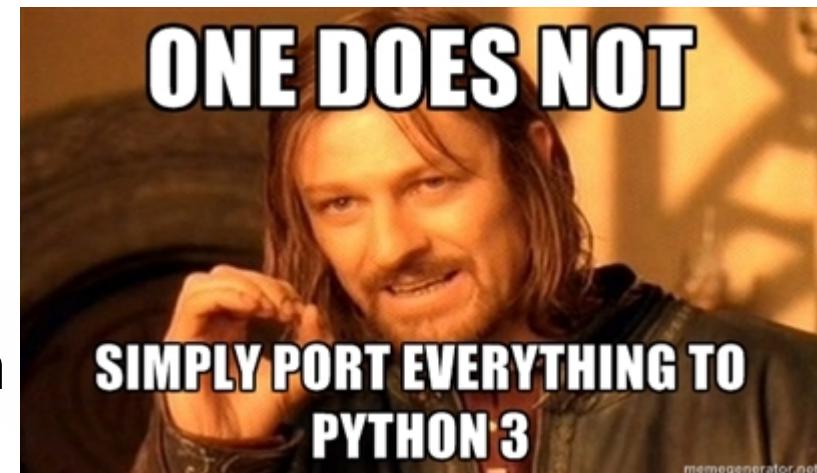
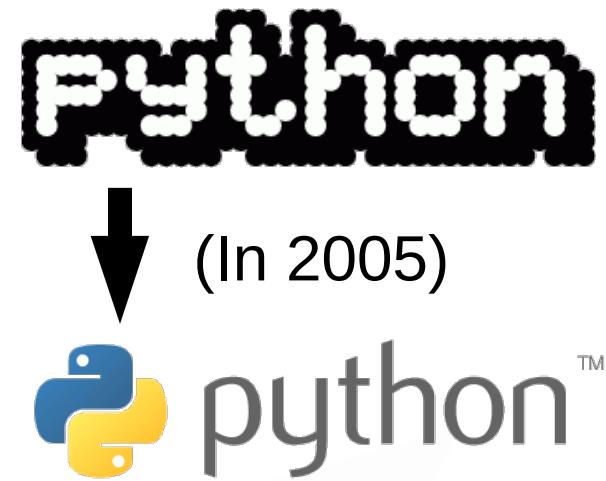
- ▶ Python 2.0, October 16, 2000
 - ▶ Cycle-detecting garbage collector for memory management
 - ▶ Unicode support



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

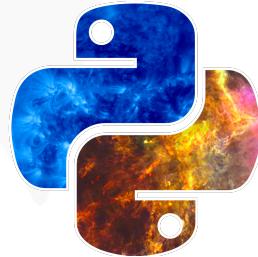


A bit of history – Use of python

Python is currently being used for “everything”



Large-scale servers
working 24/7



Astronomy



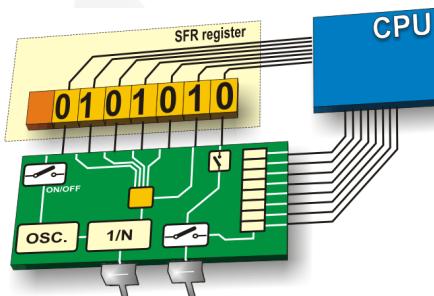
Python 1



Teaching Kids



Throw-away
scripts



Microcontrollers



Databases

TODO:
MORE?
ANY IDEAS?

A bit of history – Use of python

(Some) Companies currently massively using python

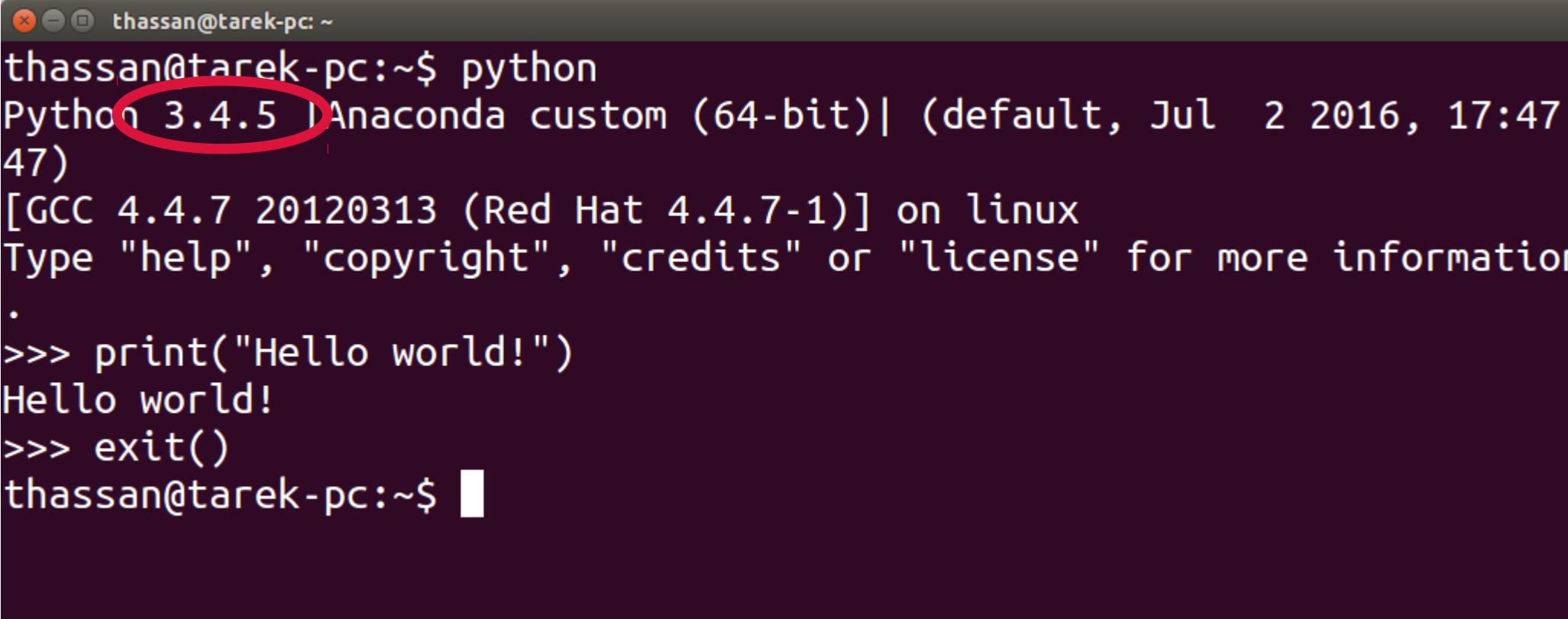


A bit of history – Python 3

- ▶ 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
```

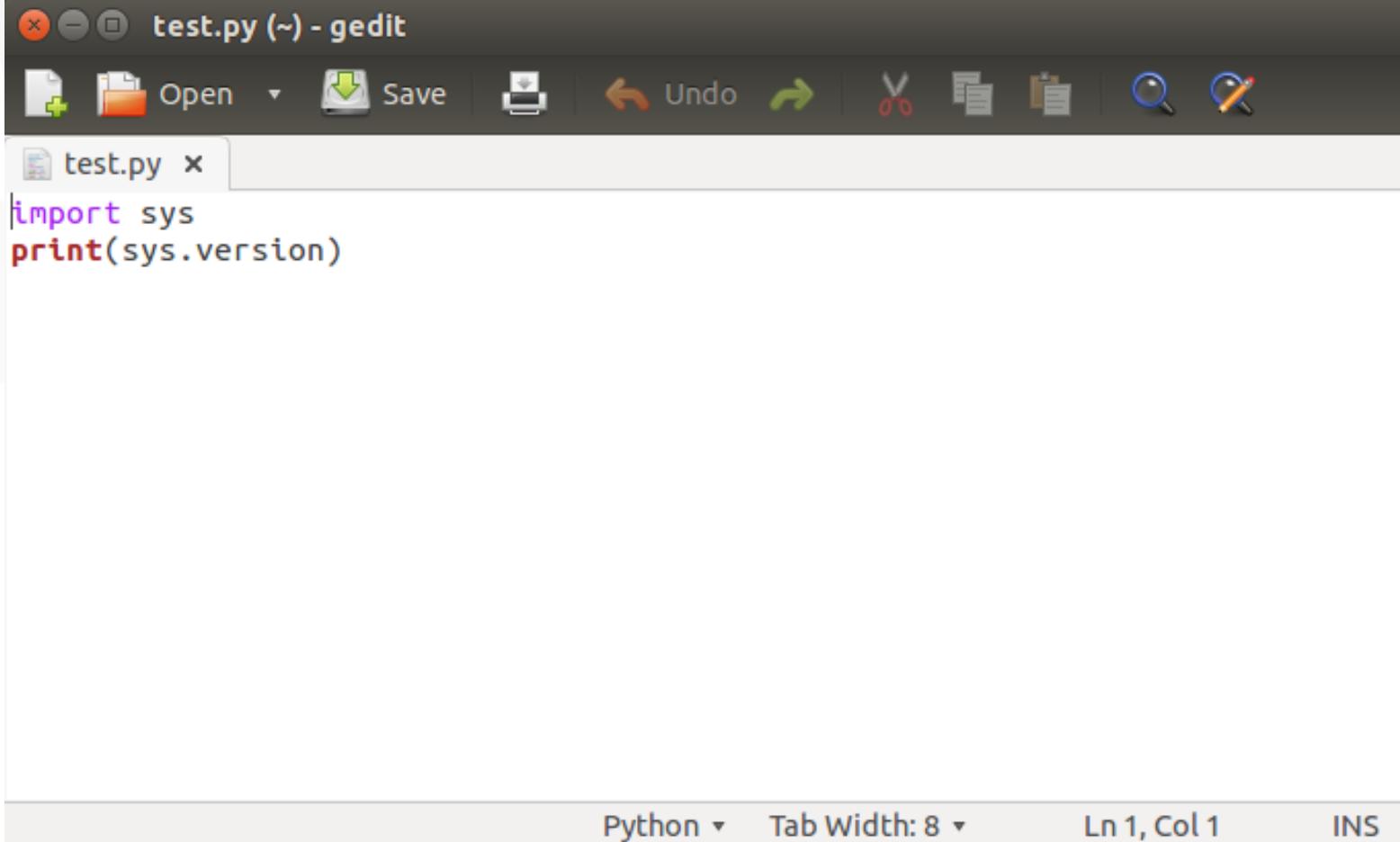
Getting started with python

- ▼ 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...)

The screenshot shows the PyCharm IDE interface. The title bar indicates the project is 'gammapy' located at '~/Paquetes/lastVersion/gammapy', and the file being edited is 'test.py' located at '~/Workspace/MAGIC_macros/test.py'. The menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar has icons for running, stopping, and saving. The left sidebar shows the 'Project' view with a tree of Python packages like gammapy, FRB_analysis_tools, and MAGIC_macros, along with sub-directories such as 1ES1959+650, BLLac, CTA macros, Debug, FRB121102, GUITests, IRM, MAGIC_utils, Mrk421, OSETI, Prod2, Prod3, Pulsed_IRFs, Takami, cpix, and pythonMacros. It also lists configuration files like .cproject, .gitignore, .project, and .pydevproject. The 'Structure' tab is selected, showing the code editor with the following Python code:

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

The status bar at the bottom shows 'Unregistered VCS root detected: The directory /home/thassan/Paquetes/lastVersion/gammapy/astropy_he...' (today 10:07) 3:1 n/a UTF-8 Git: master. A message box in the bottom right corner says 'Platform and Plugin Updates' and 'PyCharm Community Edition is ready to update.' A green '13' is visible in the bottom right corner.

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

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

```
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 gammipy
>>> 
```

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

How python works: binaries & libraries

- ▶ To install a library:
 - ▶ You may do it manually (usually not encouraged):
 - ▶ Download package, “compile” and install

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”



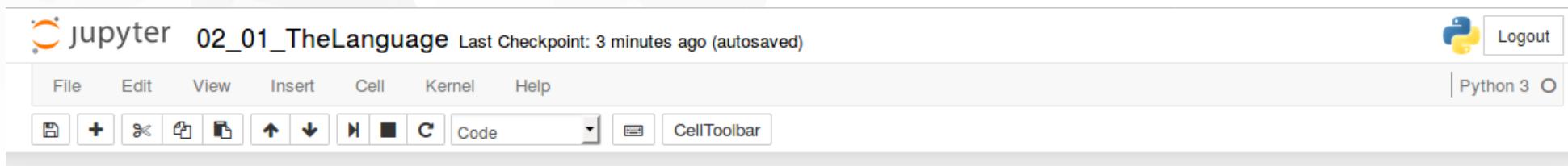
Using Jupyter notebooks

- ▼ To begin the course, 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

- 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 python (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
 - exercise: 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...