Tips to develop reusable and collaborative software

Cecilia Jarne

cecilia.jarne@unq.edu.ar

To start

What is this talk about?

- Try to remove bad habits of coding alone
- Adopt useful tools that can make the collaborative development task easier
- At least allow you to crosscheck your work with others!

WTPC
workshop en técnicas de programación científica

Summary of the general ideas:

- Group work
- Re-usable code
- Version control Software
- Clear documentation
- Do not reinvent gunpowder (that is, use tested libraries for standard calculus)

The actual problem:

First step: On the Documentation: My code, our code, group code...

Need to consider all levels for different audiences:

- Code annotations: formatting, comments
- Structure-level documentation (functions, classes)
- End-user reference material
- Introduction for new users...
- ... and new developers (often left out!)

Code writing

Code will be read much more often than written.

- Clarity wins over cleverness
- Choose a style and stick with it: block structure, indentation, line length, variable naming, ...
- Know the conventions of the language community (in Python, look for PEP 8)

Be consistent with others!

WTPC workshop en técnicas de programación científica

Code comments

In python:

```
# Here my nice and elegant comment
    of my code in Python
```

In fortran 90:

```
!* All characters after an
    exclamation mark are
    considered as comments *
```

In C:

```
/* Here my nice and elegant comment
     of my code in C*/

//also like this if is only one
    line
```

Code comments

Explain the intention and not the work:

```
1  def tripleTuple(x):
2  # assign x to y
3  y = x
4  # assign x to z
5  z = x
6  # double y
7  y = 2
8  # triple z
9  z = 3
10 # create tuple
11 t = (x, y, z)
12 # return the tuple
13 return t
```

```
1  def tripleTuple(x):
2      y = z = x
3      # apply foo scaling, see [34]
       eq (2.3)
4      y = 2
5      z = 3
6      return (x, y, z)
7
```

- Deviations from standard
- Unexpected choices of implementation

WTPC Workshop en técnicas de programación científica

Documentation:

- Dual audience: developers and end users
- Describe role of function
- Intended input / output
- Mention side effects!

Documentation tools

Extract documentation from code

- pydoc
- Doxygen
- Sphinx

Documentation:Pydoc

Pydoc extracts docstrings defined in
`http://www.python.org/dev/peps/pep-0257/`

- A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition.
- Such a docstring becomes the *__doc__* special attribute of that object.
- Every Python installation has pydoc, no extra work required.

Documentation: Doxygen

- Supports C++, C, Obj-C, C#, PHP, Java, Python,
- IDL, Fortran, VHDL, Tcl
- Can extract structure from undocumented files
- Graphical visualization of dependencies
- Can write general pages, too

Documentation: Sphinx

- Automatic extraction of Python docstrings conversion from Doxygen available via "Breathe" project
- Great support for additional structure
- Used in standard library, matplotlib, scipy, numpy
- Like most Python projects, excellent support for beginners:

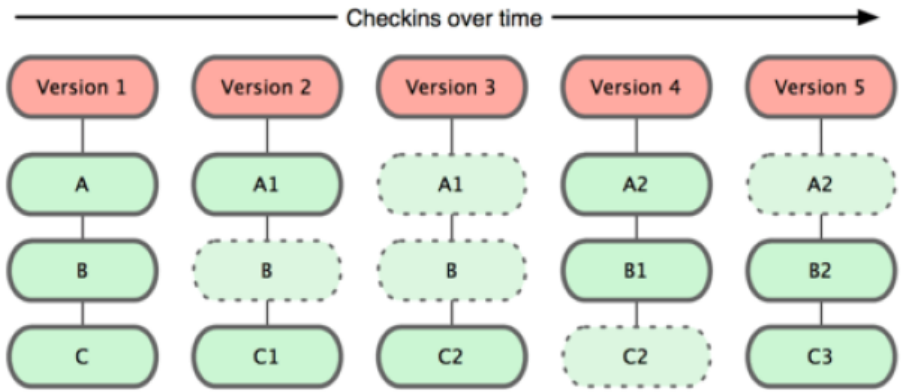
```
sphinx-quickstart
```

Some consideration

- New developers need an intro
- Very different requirements than users
- Doxygen alone is not enough
- At minimum, prepare some paths to follow through the code docs

Other good practices

Version control system: git



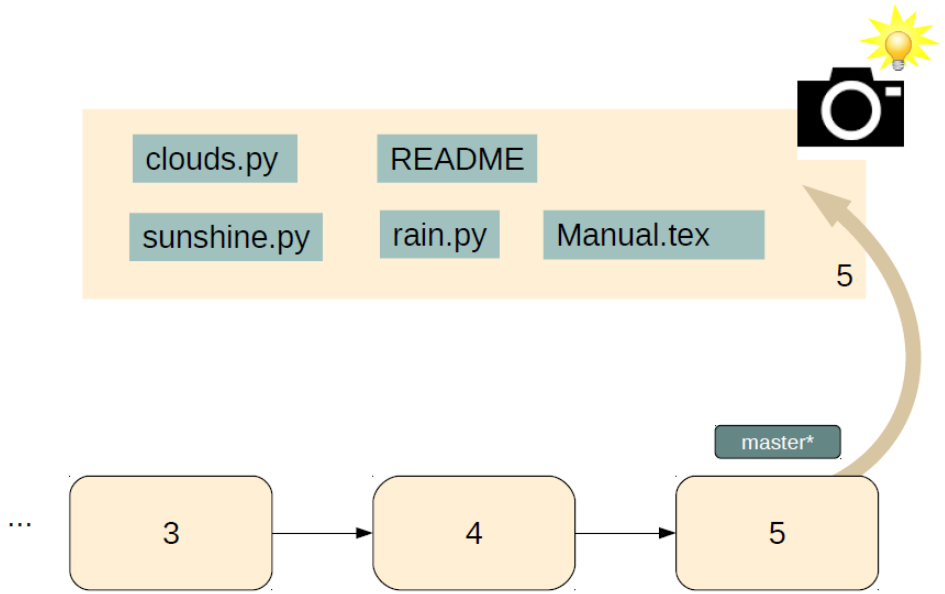
`www.git-scm.com`

Other good practice: Version control software

- Storing Versions Properly (Allows to document changes on the code)
- Allows a team of people to work programing together, all using the same files.
- Ordered way of multiple people editing the same files.
- Allows restoring Previous Versions
- Allows shear the changes for all contributors

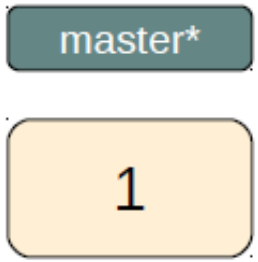
Other good practice: Version control software

An example of a project:



An example for version control system: git

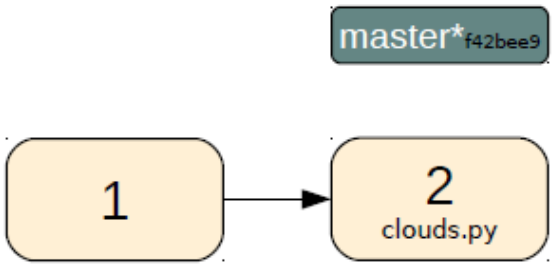
```
> git init
```




```
> git status
On branch master
Initial commit
nothing to commit (create/copy files and use "git add" to track)
```

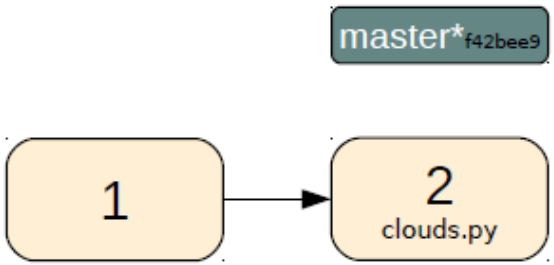
An example for version control system: git

```
> vim clouds.py
> git add clouds.py
> git commit -m 'test cases added'
[master (root-commit) f42bee9] test case
1 file changed, 1 insertion(+)
create mode 100644 clouds.py
```



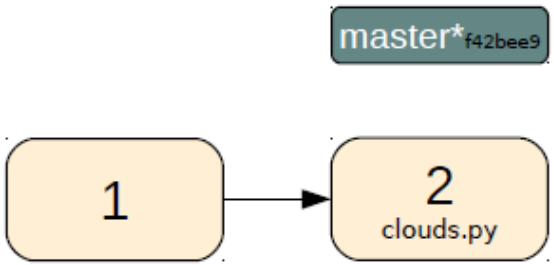
An example for version control system: git

```
> git log
commit f28ef8b10951a4ecc95dc911bbe7018e3c0225fc
Author: Cecilia Jarne <cecilia.jarne@unq.edu.ar>
Date: Mon Oct 26 12:36:46 2015 -0300
test cases added
```



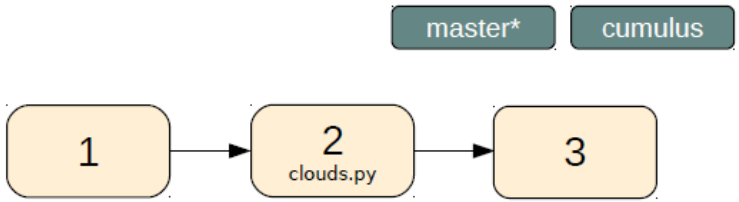
An example for version control system: git

```
> vim wind.py
> git status
On branch master
Untracked files:
(use "git add <file>..." to include in what will be committed)
wind.py
nothing added to commit
```



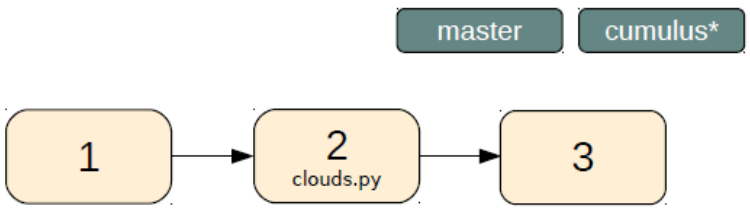
An example for version control system: git

```
> git branch cumulus
```



It is a pointer (does not copy the code!)

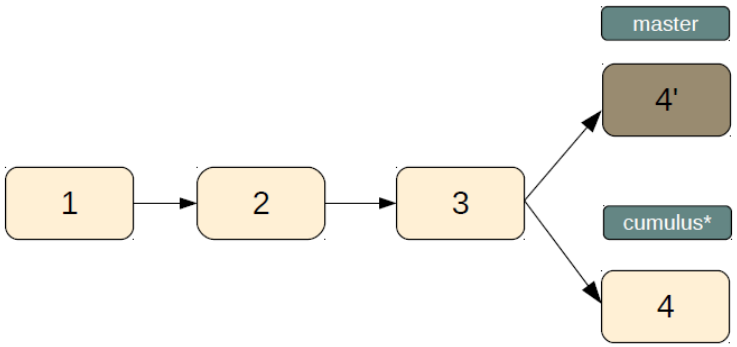
```
> git checkout cumulus
```



WTPC
Workshop en técnicas de programación científica

An example for version control system: git

```
>vim clouds.py #to modify clouds.py
> git add clouds.py
> git commit -m 'added parametrization [cumulus]'
> git commit -m 'added parametrization
```



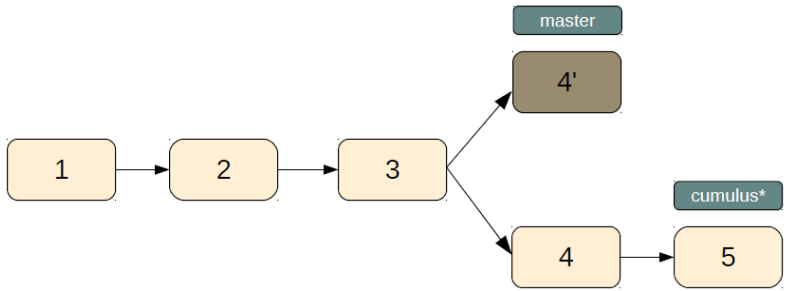
WTPC
Workshop en técnicas de programación científica

An example for version control system: git

```
> git merge cumulus master
```

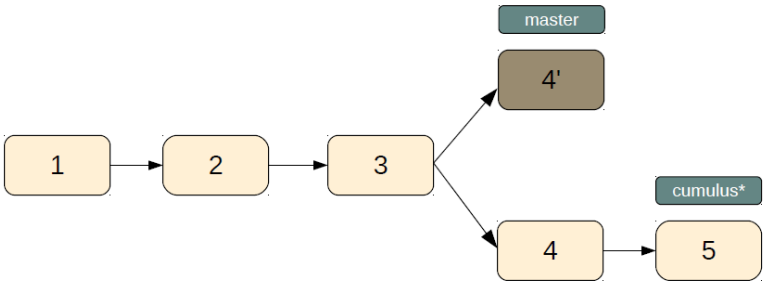
or

```
> git merge master
```



An example for version control system: git

I am in the branch cumulus and I want to merge it with master



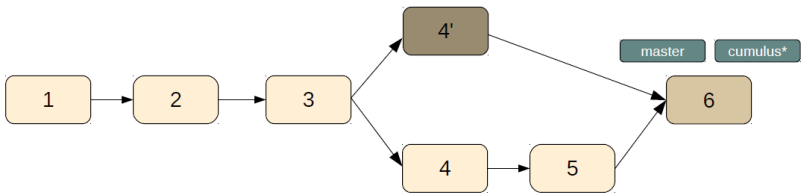
An example for version control system: git

I am in the branch cumulus and I want to merge it with master

```
> git merge cumulus master
```

or

```
> git merge master
```



An example for version control system: git

To delete files

```
> rm wind.py
> git status
On branch master
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory
    )
deleted: wind.py
no changes added to commit (use "git add" and/or "git commit -a")
```

An example for version control system: git

Some commands on workspace, staging area, local repository

- init #to create repository
- status # repository status
- add #add to the stage
- commit #save in the repository what is in stage
- branch #new branch
- checkout #switch branch
- diff #show differences between branches
- log #show commits

WTPC Workshop en técnicas de programación científica

Remote work

```
> git clone https://github.com/<USER>/HOgit.git
```

```
> git push
```

```
> git pull
```

So ....

Credits

WTPC group (especially Rodrigo Lugones)
Distributed Version Control David Grellscheid
Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages ICPT SAIFR
GIT's hands on
Axel Kohlmeyer https://sites.google.com/site/akohlmey/ website git https://git-scm.com/book/en/v2