

# Scientific Computing

## Command line, GIT and version control

Peter Regner, Johannes Schmidt

Institute for Sustainable Economic Development, BOKU, Wien

2020-03-19



1. Organizational matters
2. Using a command line
3. GIT Version Control System
4. Exercise
5. Homework assignment

Organizational matters

# Organizational matters

- ▶ Let's try to write lecture notes collaboratively additional to the slides:  
<https://yourpart.eu/p/lecture-scientific-computing01-notes>

# Organizational matters

- ▶ Let's try to write lecture notes collaboratively additional to the slides:  
<https://yourpart.eu/p/lecture-scientific-computing01-notes>
- ▶ Have you found a group for the homework assignments?
  - ▶ Vote **YES** if you found a group
  - ▶ Vote **NO** if you are still looking for a group

# Organizational matters

- ▶ Let's try to write lecture notes collaboratively additional to the slides:  
<https://yourpart.eu/p/lecture-scientific-computing01-notes>
- ▶ Have you found a group for the homework assignments?
  - ▶ Vote **YES** if you found a group
  - ▶ Vote **NO** if you are still looking for a group
- ▶ Are you registered on github.com?
  - ▶ Vote **YES**
  - ▶ Vote **NO**, somewhere else

# Organizational matters

- ▶ Let's try to write lecture notes collaboratively additional to the slides:  
<https://yourpart.eu/p/lecture-scientific-computing01-notes>
- ▶ Have you found a group for the homework assignments?
  - ▶ Vote **YES** if you found a group
  - ▶ Vote **NO** if you are still looking for a group
- ▶ Are you registered on github.com?
  - ▶ Vote **YES**
  - ▶ Vote **NO**, somewhere else
- ▶ Outlook: next lectures:
  - ▶ Conda (Homework: encrypted mapping from Github account to real Name)
  - ▶ Python (Homework: #flattenthecurve - by the way: please stay home!)

# Organizational matters

- ▶ Let's try to write lecture notes collaboratively additional to the slides:  
<https://yourpart.eu/p/lecture-scientific-computing01-notes>
- ▶ Have you found a group for the homework assignments?
  - ▶ Vote **YES** if you found a group
  - ▶ Vote **NO** if you are still looking for a group
- ▶ Are you registered on github.com?
  - ▶ Vote **YES**
  - ▶ Vote **NO**, somewhere else
- ▶ Outlook: next lectures:
  - ▶ Conda (Homework: encrypted mapping from Github account to real Name)
  - ▶ Python (Homework: #flattenthecurve - by the way: please stay home!)
- ▶ Homework - anybody wants to present?

## Using a command line

## Command line: Live Demo

Live Demo:

```
cowsay -f snowman "I'm sweating"
```

```
cowthink "meh."
```

## What is a command line?

A command line is a computer interface, which allows you to type commands with parameters and to receive text output

# What is a command line?

A command line is a computer interface, which allows you to type commands with parameters and to receive text output:

Type something like:

```
command parameter1 parameter2[ENTER]
```

Then the **command** will run and output will be printed on the screen.

# What is a command line?

A command line is a computer interface, which allows you to type commands with parameters and to receive text output:

Type something like:

```
command parameter1 parameter2[ENTER]
```

Then the `command` will run and output will be printed on the screen.

- ▶ Often used as synonyms: terminal, shell, command line, console
- ▶ Everything about command lines refers to Linux command lines, but git-bash in Windows and the terminal in Mac OS are very similar, but the standard Windows terminal is quite different (don't use it: PITA).

## Command line: Syntax

- ▶ Command and all parameters are space-separated

## Command line: Syntax

- ▶ Command and all parameters are space-separated
- ▶ Parameters containing spaces (e.g. file paths) need to be wrapped in double or single quotes:

```
ls "path/to a folder with spaces/subfolder"
```

# Command line: Syntax

- ▶ Command and all parameters are space-separated
- ▶ Parameters containing spaces (e.g. file paths) need to be wrapped in double or single quotes:

```
ls "path/to a folder with spaces/subfolder"
```

- ▶ There are different standard forms of parameters:

- ▶ command parameter
- ▶ command --parameter-name=parameter
- ▶ command --parameter-name parameter
- ▶ command -p parameter
- ▶ command -p=parameter

## File and folder paths

- ▶ **working directory:** when using a terminal, you are always working in a directory, similar to opening a folder in a file browser

# File and folder paths

- ▶ **working directory:** when using a terminal, you are always working in a directory, similar to opening a folder in a file browser
- ▶ **absolute file paths** start with / (Linux, Mac OS) or something like C:\ (Windows), example:

```
ls /home/peter/lecture-scientific-computing/README.md
```

# File and folder paths

- ▶ **working directory:** when using a terminal, you are always working in a directory, similar to opening a folder in a file browser
- ▶ **absolute file paths** start with / (Linux, Mac OS) or something like C:\ (Windows), example:  
`ls /home/peter/lecture-scientific-computing/README.md`
- ▶ **relative paths** are relative to the current working directory of the terminal (with some few unimportant exceptions), example:  
`ls lecture-scientific-computing/README.md`

# Important command line commands

- ▶ print working directory

```
$ pwd
```

```
/home/peter
```

# Important command line commands

- ▶ print working directory

```
$ pwd
```

```
/home/peter
```

- ▶ change (working) directory

```
$ cd path/to/folder
```

# Important command line commands

- ▶ print working directory

```
$ pwd  
/home/peter
```

- ▶ change (working) directory

```
$ cd path/to/folder
```

- ▶ ls -l list directory content

```
$ ls -l  
-rw-rw-r-- 1 peter peter 2,0K Aug 3 2009 book_1.docx  
-rw-rw-r-- 1 peter peter 5,0K Aug 3 2009 book_2.docx
```

## Helpful tricks

- ▶ abort a running command:

CTRL + C

## Helpful tricks

- ▶ abort a running command:

CTRL + C

- ▶ hitting the TAB key auto-completes your command in many situations, e.g.:

ls path/to/fol<TAB>

## Helpful tricks

- ▶ abort a running command:  
CTRL + C
- ▶ hitting the TAB key auto-completes your command in many situations, e.g.:  
`ls path/to/fol<TAB>`
- ▶ previously run commands can be accessed using cursor keys or search:  
CTRL + R

# Helpful tricks

- ▶ abort a running command:  
CTRL + C
- ▶ hitting the TAB key auto-completes your command in many situations, e.g.:  
`ls path/to/fol<TAB>`
- ▶ previously run commands can be accessed using cursor keys or search:  
CTRL + R
- ▶ run command in background using a trailing &, useful for running gitk:  
`gitk&`

# Getting help

Many commands print a help text when called with parameter `-h` or `--help`:

```
$ sleep --help
Usage: sleep NUMBER[SUFFIX]...
      or: sleep OPTION
Pause for NUMBER seconds. SUFFIX may be 's' for
seconds (the default), 'm' for minutes, 'h' for
hours or 'd' for days.
```

# Getting help

Many commands print a help text when called with parameter `-h` or `--help`:

```
$ sleep --help
Usage: sleep NUMBER[SUFFIX]...
      or: sleep OPTION
Pause for NUMBER seconds. SUFFIX may be 's' for
seconds (the default), 'm' for minutes, 'h' for
hours or 'd' for days.
```

But googling is fine too!

# Getting help

Many commands print a help text when called with parameter `-h` or `--help`:

```
$ sleep --help
Usage: sleep NUMBER[SUFFIX]...
      or: sleep OPTION
Pause for NUMBER seconds. SUFFIX may be 's' for
seconds (the default), 'm' for minutes, 'h' for
hours or 'd' for days.
```

But googling is fine too!

Introduction tutorial:

<https://ubuntu.com/tutorials/command-line-for-beginners>

# GIT Version Control System

# GIT saves you if things go wrong

A screenshot of a web browser showing the GIT Insurance homepage. The URL https://www.gitinsurance.com is visible in the address bar. The page features a large banner image of a city skyline. Overlaid on the banner are two blue call-to-action buttons: "Request A Free Quote" and "Get A Quote >". At the top of the page, there is a navigation bar with links for "Our Agency", "Products", and "Contact Us". In the top right corner, there is contact information: a phone icon followed by the number 515-897-4448, and a fax number 888-651-7816. The GIT Insurance logo, which includes a stylized 'G' and 'I' icon, is located at the top left.



**GIT Insurance**

245 NE Venture Dr Suite 1  
Waukee, IA 50263



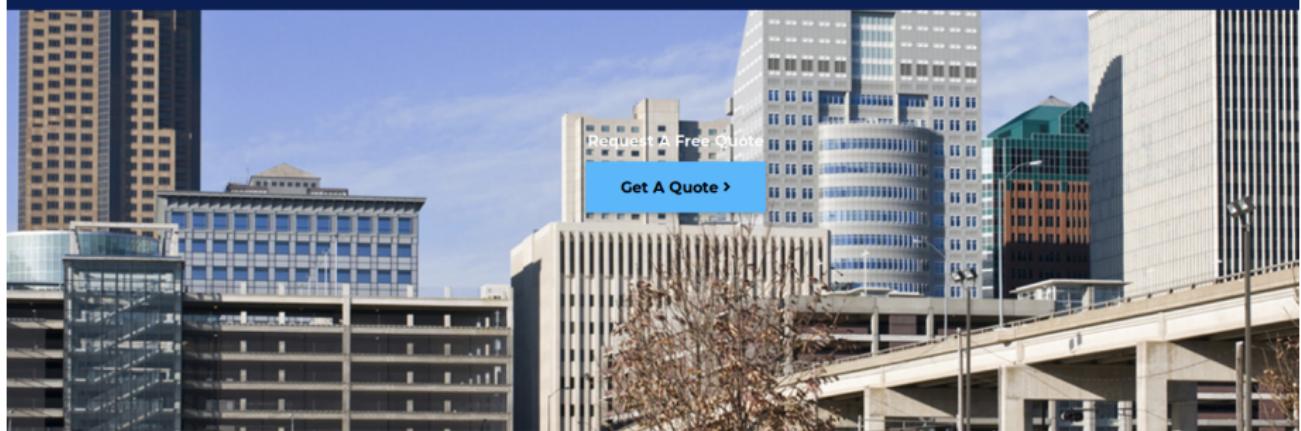
**515-897-4448**

Fax: 888-651-7816

Our Agency ▾

Products ▾

Contact Us ▾



**PROGRESSIVE®**

## GIT: Goals for today

- ▶ understand why version control is essential for programming

## GIT: Goals for today

- ▶ understand why version control is essential for programming
- ▶ get an overview of what can be done using GIT

## GIT: Goals for today

- ▶ understand why version control is essential for programming
- ▶ get an overview of what can be done using GIT
- ▶ be able to use GIT to collaborate with your group mates for the homework assignments, understand basic terms and commands

# GIT: Goals for today

- ▶ understand why version control is essential for programming
- ▶ get an overview of what can be done using GIT
- ▶ be able to use GIT to collaborate with your group mates for the homework assignments, understand basic terms and commands
  - ▶ repository
  - ▶ fork
  - ▶ configure collaborators in Github
  - ▶ git add
  - ▶ git status
  - ▶ git commit
  - ▶ git push
  - ▶ git pull
  - ▶ use gitk and Github to display the commit history
  - ▶ use Github issues
  - ▶ solve merge conflicts

# GIT: Goals for today

- ▶ understand why version control is essential for programming
- ▶ get an overview of what can be done using GIT
- ▶ be able to use GIT to collaborate with your group mates for the homework assignments, understand basic terms and commands
  - ▶ repository
  - ▶ fork
  - ▶ configure collaborators in Github
  - ▶ git add
  - ▶ git status
  - ▶ git commit
  - ▶ git push
  - ▶ git pull
  - ▶ use gitk and Github to display the commit history
  - ▶ use Github issues
  - ▶ solve merge conflicts

There may be a bit of bonus content if time permits.

# It's always a good idea to keep previous versions...! :)

/home/peter/predicting_boastful_resistance/			
Name	Size	Type	Date Modified
book FIXED IT.docx	99,0 KiB	Microsoft Word Document	2009-08-12 05:25:13
book_final-upload-this.docx	100,0 KiB	Microsoft Word Document	2009-08-12 03:55:12
book_final_-REALLY_FINALLL-uargh.docx	100,0 KiB	Microsoft Word Document	2009-08-11 23:12:34
book_final (Copy 1)_LAST (Copy 1).docx	102,0 KiB	Microsoft Word Document	2009-08-11 22:02:34
book_final.docx	101,0 KiB	Microsoft Word Document	2009-08-09 19:42:23
book_4-corrections.docx	97,0 KiB	Microsoft Word Document	2009-08-04 15:10:10
book_4 (Copy 2).docx	91,0 KiB	Microsoft Word Document	2009-08-04 14:00:54
book_4.docx	97,0 KiB	Microsoft Word Document	2009-08-04 11:37:23
book_2.docx	5,0 KiB	Microsoft Word Document	2009-08-03 17:34:14
book_1.docx	2,0 KiB	Microsoft Word Document	2009-08-03 10:04:03

# It's always a good idea to keep previous versions...! :)

/home/peter/predicting_boastful_resistance/			
Name	Size	Type	Date Modified
book FIXED IT.docx	99,0 KiB	Microsoft Word Document	2009-08-12 05:25:13
book_final-upload-this.docx	100,0 KiB	Microsoft Word Document	2009-08-12 03:55:12
book_final_-REALLY_FINALLL-uargh.docx	100,0 KiB	Microsoft Word Document	2009-08-11 23:12:34
book_final (Copy 1)_LAST (Copy 1).docx	102,0 KiB	Microsoft Word Document	2009-08-11 22:02:34
book_final.docx	101,0 KiB	Microsoft Word Document	2009-08-09 19:42:23
book_4-corrections.docx	97,0 KiB	Microsoft Word Document	2009-08-04 15:10:10
book_4 (Copy 2).docx	91,0 KiB	Microsoft Word Document	2009-08-04 14:00:54
book_4.docx	97,0 KiB	Microsoft Word Document	2009-08-04 11:37:23
book_2.docx	5,0 KiB	Microsoft Word Document	2009-08-03 17:34:14
book_1.docx	2,0 KiB	Microsoft Word Document	2009-08-03 10:04:03

If you don't see the problem here, read this:

<http://phdcomics.com/comics/archive.php?comicid=1531>

# GIT: Chronological order

book FIXED IT.docx	99,0 KiB Microsoft Word Document	2009-08-12 05:25:13
book_final-upload-this.docx	100,0 KiB Microsoft Word Document	2009-08-12 03:55:12
book_final_-REALLY_FINALLL-uargh.docx	100,0 KiB Microsoft Word Document	2009-08-11 23:12:34
book_final (Copy 1)_LAST (Copy 1).docx	102,0 KiB Microsoft Word Document	2009-08-11 22:02:34
book_final.docx	101,0 KiB Microsoft Word Document	2009-08-09 19:42:23
book_4-corrections.docx	97,0 KiB Microsoft Word Document	2009-08-04 15:10:10
book_4 (Copy 2).docx	91,0 KiB Microsoft Word Document	2009-08-04 14:00:54
book_4.docx	97,0 KiB Microsoft Word Document	2009-08-04 11:37:23
book_2.docx	5,0 KiB Microsoft Word Document	2009-08-03 17:34:14
book_1.docx	2,0 KiB Microsoft Word Document	2009-08-03 10:04:03

# GIT: Describe what really happened

• Add one more graphic to chapter 1	99,0 KiB Microsoft Word Document	2009-08-12 05:25:13
• Fix some grammar mistakes	100,0 KiB Microsoft Word Document	2009-08-12 03:55:12
• Change layout to final	100,0 KiB Microsoft Word Document	2009-08-11 23:12:34
• Add introduction	102,0 KiB Microsoft Word Document	2009-08-11 22:02:34
• Add chapter 5	101,0 KiB Microsoft Word Document	2009-08-09 19:42:23
• Fix some typos in chapter 2	97,0 KiB Microsoft Word Document	2009-08-04 15:10:10
• Rewrite some parts in chapter 2	91,0 KiB Microsoft Word Document	2009-08-04 14:00:54
• Add chapter 4	97,0 KiB Microsoft Word Document	2009-08-04 11:37:23
• Add chapters 2-3	5,0 KiB Microsoft Word Document	2009-08-03 17:34:14
• First version, includes Chapter 1	2,0 KiB Microsoft Word Document	2009-08-03 10:04:03

# GIT: Authorship is important for collaboration

• Add one more graphic to chapter 1	Peter <peter@suuf.cc>	2009-08-12 05:25:13
• Fix some grammar mistakes	Peter <peter@suuf.cc>	2009-08-12 03:55:12
• Change layout to final	Peter <peter@suuf.cc>	2009-08-11 23:12:34
• Add introduction	Peter <peter@suuf.cc>	2009-08-11 22:02:34
• Add chapter 5	Peter <peter@suuf.cc>	2009-08-09 19:42:23
• Fix some typos in chapter 2	Peter <peter@suuf.cc>	2009-08-04 15:10:10
• Rewrite some parts in chapter 2	Peter <peter@suuf.cc>	2009-08-04 14:00:54
• Add chapter 4	Peter <peter@suuf.cc>	2009-08-04 11:37:23
• Add chapters 2-3	Peter <peter@suuf.cc>	2009-08-03 17:34:14
• First version, includes Chapter 1	Peter <peter@suuf.cc>	2009-08-03 10:04:03

# gitk - a simple GIT viewer

File Edit View Help

- Local uncommitted changes, not checked in to index
- master** remotes/origin/master Make URLs colorful
- Add git.txt
- Add command line intro slides
- Move layout for slides to common folder
- Add link to list of issues in README
- Add re-compiled slides.pdf
- Clarify instructions for first homework
- Add re-compiled slides.pdf
- Make GIT instructions more explicit
- Add re-compiled slides.pdf
- Fix instructions for first commit
- tag...** Add re-compiled slides.pdf
- Remove link to menti.com

SHA1 ID: a9adcab97ea55c5630e5fa31fed072fc92f31270 ↵ Row 8/ 51

Find ↑ commit containing: Exact All fields.

Search

- Diff Old version New version Lines of context: 3 Ignore space change Line diff

Committer: lumbric <lumbric@gmail.com> 2020-03-16 11:13:42  
Parent: e20c759bc3e2b5d7517015ca66fb604c2baedfb (Add re-compiled slides.pdf)  
Child: 21f0bb8487b3118d30c4636224430e85c4923979 (Add re-compiled slides.pdf)  
Branches: master, remotes/origin/master  
Follows: 2020-03-12-lecture00  
Precedes:

Clarify instructions for first homework

----- lecture00-introduction/slides.tex -----  
index 1867a06..ac751c6 100644  
@@ -338,7 +338,7 @@ Homework (due on 19th of March):  
 \end{verbatim}  
 }  
  
- Then change a file of your choice and run:  
+ Then edit any arbitrary file, e.g. add "I was here" to README.md and run:  
  
 \begin{verbatim}  
 cd lecture-scientific-computing

Patch Tree  
Comments  
lecture00-introduction/slides.tex

# gitk - a simple GIT viewer

File Edit View Help

SHA1 ID	Author	Commit Message	Date
7bdc26543725e46b47e51c801a3c52e4a8cc29f7	Matti Picus <matti.picus@gmail.com>	Merge pull request #15766 from seberg/simple	2020-03-18 12:15:31
	Sebastian Berg <sebastian@sipsolutions.net>	BUG: Increase default string cast size of longdouble/clongdouble	2020-03-17 02:15:55
	Sebastian Berg <sebastian@sipsolutions.net>	BUG,MAINT: Remove incorrect special case in string to number casts	2020-03-17 00:32:19
	Anirudh Subramanian <anirudh2290@ufl.edu>	Merge pull request #15771 from anirudh2290/fix_dev_doc	2020-03-18 01:42:37
	Ralf Gommers <ralf.gommers@gmail.com>	DOC: Fix runtests example in developer docs	2020-03-18 01:02:41
	Charles Harris <charlesr.harris@gmail.com>	Merge pull request #15768 from charris/update-after-1.18.2-release	2020-03-17 19:57:37
	Matti Picus <matti.picus@gmail.com>	REL: Update master after 1.18.2 release.	2020-03-17 19:04:29
	Sebastian Berg <sebastian@sipsolutions.net>	BUG: add missing c_distributions.pxd, enables cython use of random	2020-03-16 23:02:06
	dependabot-preview[bot] <27856297+dependabot-preview[bot]>	Merge pull request #15762 from numpy/dependabot/pip/pytest!	2020-03-16 15:03:53
	Rakesh Vasudevan <rakesh.nvasudev@gmail.com>	MAINT: Bump pytest from 5.3.5 to 5.4.1	2020-03-16 09:17:52
	Warren Weckesser <warren.weckesser@gmail.com>	MAINT: Test during import to detect bugs with Accelerate(MacOS X)	2020-03-15 19:14:34
	Warren Weckesser <warren.weckesser@gmail.com>	Merge pull request #15747 from WarrenWeckesser/unique-with-	2020-03-15 19:00:19
	Warren Weckesser <warren.weckesser@gmail.com>	MAINT: lib: PEP-8 clean up in test_arraysetops.py.	2020-03-13 21:25:06
	Warren Weckesser <warren.weckesser@gmail.com>	BUG: lib: Handle axes with length 0 in np.unique.	2020-03-13 00:53:13
	Huon Wilson <Huon.Wilson@data61.csiro.au>	TST: lib: Add a unit test for np.unique applied to arrays with a lot	2020-03-12 01:07:33

SHA1 ID: 7bdc26543725e46b47e51c801a3c52e4a8cc29f7 Row: 5 / 26090

Find ↕ commit containing:  Exact All fields

Diff Old version New version Lines of context: 3 Ignore spaces Previews:

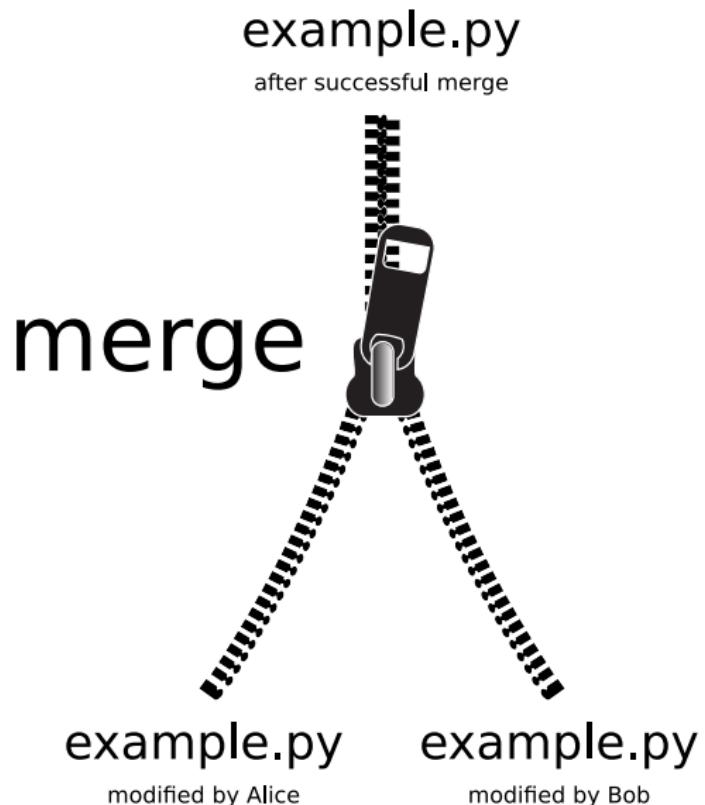
DOC: Fix runtests example in developer docs

```
----- doc/source/dev/development_environment.rst
index c20b6fe40..1f404d6e0 100644
@@ -44,7 +44,7 @@ arguments may be forwarded to the target embedd
the extra arguments after a bare ``--``. For example, to run a t
the ``--pdb`` flag forwarded to the target, run the following::
```

```
-   $ python runtests.py -t numpy/tests/test_scripts.py::test_f2p
+   $ python runtests.py -t numpy/tests/test_scripts.py::test_f2
```

When using pytest as a target (the default), you can  
`match test names using python operators`\_ by passing the ``-k``

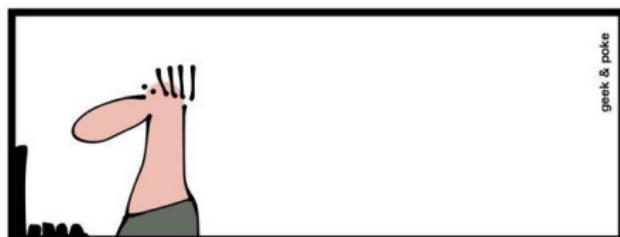
# Merging two different versions of a file



# git blame

```
0380d51c6 (Keisuke Fujii) 2018-09-28 08:54:29 +0200 13
0380d51c6 (Keisuke Fujii) 2018-09-28 08:54:29 +0200 14 # Used as a sentinel value to indicate a all dimensions
0380d51c6 (Keisuke Fujii) 2018-09-28 08:54:29 +0200 15 ALL_DIMS = ReprObject('<all-dims>')
adcc4836c (Stephan Hoyer) 2014-04-30 01:56:35 -0700 16
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 17
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 18 class ImplementsArrayReduce(object):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 19     @classmethod
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 20         def _reduce_method(cls, func, include_skipna, numeric_only):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 21             if include_skipna:
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 22                 def wrapped_func(self, dim=None, axis=None, skipna=None,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 23                     **kwargs):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 24                     return self.reduce(func, dim, axis,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 25                         skipna=skipna, allow_lazy=True, **kwargs)
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 26             else:
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 27                 def wrapped_func(self, dim=None, axis=None, # type: ignore
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 28                     **kwargs):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 29                     return self.reduce(func, dim, axis,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 30                         allow_lazy=True, **kwargs)
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 31             return wrapped_func
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 32
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 33     _reduce_extra_args_docstring = dedent("""\
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 34         dim : str or sequence of str, optional
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 35             Dimension(s) over which to apply '{name}'.
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 36         axis : int or sequence of int, optional
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 37             Axis(es) over which to apply '{name}'. Only one of the 'dim'
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 38             and 'axis' arguments can be supplied. If neither are supplied, then
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 39             '{name}' is calculated over axes.'")
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 40
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 41     _cum_extra_args_docstring = dedent("""\
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 42         dim : str or sequence of str, optional
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 43             Dimension over which to apply '{name}'.
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 44         axis : int or sequence of int, optional
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 45             Axis over which to apply '{name}'. Only one of the 'dim'
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 46             and 'axis' arguments can be supplied.'")
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 47
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 48 class ImplementsDatasetReduce(object):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 49     @classmethod
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 50         def _reduce_method(cls, func, include_skipna, numeric_only):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 51             if include_skipna:
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 52                 def wrapped_func(self, dim=None, skipna=None,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 53                     **kwargs):
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 54                     return self.reduce(func, dim, skipna=skipna,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 55                         numeric_only=numeric_only, allow_lazy=True,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 56                         **kwargs)
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 57             else:
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 58                 def wrapped_func(self, dim=None, **kwargs): # type: ignore
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 59                     return self.reduce(func, dim,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 60                         numeric_only=numeric_only, allow_lazy=True,
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 61                         **kwargs)
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 62             return wrapped_func
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 63
18c7b9dbf (Stephan Hoyer) 2014-06-22 01:07:03 -0700 64
```

# git blame



GIT BLAME

# git blame



GIT BLAME

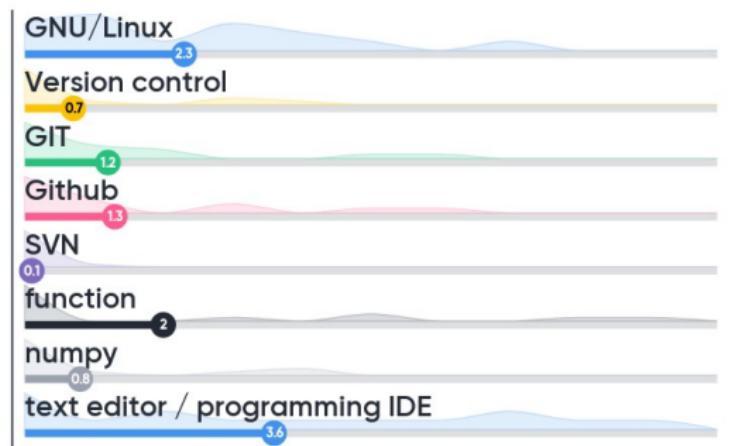
Github more popular than GIT



# Github more popular than GIT

Mentimeter

Rate these terms (0: What is this? 10: I know how to use this)



15

# Github

Some things worth noting about the [Github web-interface](#):

- ▶ the README.md should give a quick intro or more complete documentation
- ▶ stars, watchers and forks and when the last commit are good indicators for quality
- ▶ by the way: if you want, watch our repository to get mail notifications if some body posts a question in the issue tracker ([more about watching repositories](#))
- ▶ repository settings: permissions, enable/disable features (bug tracker, wiki)
- ▶ license says how you are allowed to use the code
- ▶ Github actions allow to run code (build jobs) on Github's servers after each commit ([example](#))

# Github

Some things worth noting about the [Github web-interface](#):

- ▶ the README.md should give a quick intro or more complete documentation
- ▶ stars, watchers and forks and when the last commit are good indicators for quality
- ▶ by the way: if you want, watch our repository to get mail notifications if some body posts a question in the issue tracker ([more about watching repositories](#))
- ▶ repository settings: permissions, enable/disable features (bug tracker, wiki)
- ▶ license says how you are allowed to use the code
- ▶ Github actions allow to run code (build jobs) on Github's servers after each commit ([example](#))

Example user page:

<https://github.com/gvanrossum>

# GIT: Why?

- ▶ GIT as solution for versioning files

# GIT: Why?

- ▶ GIT as solution for versioning files
  - ▶ changes ("commits") can be reverted also individually
  - ▶ major versions (branches) can be maintained individually, changes (e.g. bug fixes) applied
  - ▶ splitting work into small parts, helps to keep an overview of changes, but also to find bugs (git blame, git bisect)

# GIT: Why?

- ▶ GIT as solution for versioning files
  - ▶ changes ("commits") can be reverted also individually
  - ▶ major versions (branches) can be maintained individually, changes (e.g. bug fixes) applied
  - ▶ splitting work into small parts, helps to keep an overview of changes, but also to find bugs (git blame, git bisect)
- ▶ GIT as solution for collaboration

# GIT: Why?

- ▶ GIT as solution for versioning files
  - ▶ changes ("commits") can be reverted also individually
  - ▶ major versions (branches) can be maintained individually, changes (e.g. bug fixes) applied
  - ▶ splitting work into small parts, helps to keep an overview of changes, but also to find bugs (git blame, git bisect)
- ▶ GIT as solution for collaboration
  - ▶ multiple developers can work on code at the same time independently and then merge changes into one version
  - ▶ work is split into readable and consistent parts, which allows code review by other developers

# GIT: Why?

- ▶ GIT as solution for versioning files
  - ▶ changes ("commits") can be reverted also individually
  - ▶ major versions (branches) can be maintained individually, changes (e.g. bug fixes) applied
  - ▶ splitting work into small parts, helps to keep an overview of changes, but also to find bugs (git blame, git bisect)
- ▶ GIT as solution for collaboration
  - ▶ multiple developers can work on code at the same time independently and then merge changes into one version
  - ▶ work is split into readable and consistent parts, which allows code review by other developers
- ▶ GIT for distributing software/code

# GIT: Why?

- ▶ GIT as solution for versioning files
  - ▶ changes ("commits") can be reverted also individually
  - ▶ major versions (branches) can be maintained individually, changes (e.g. bug fixes) applied
  - ▶ splitting work into small parts, helps to keep an overview of changes, but also to find bugs (git blame, git bisect)
- ▶ GIT as solution for collaboration
  - ▶ multiple developers can work on code at the same time independently and then merge changes into one version
  - ▶ work is split into readable and consistent parts, which allows code review by other developers
- ▶ GIT for distributing software/code
  - ▶ Github is the main platform for hosting opensource code
  - ▶ libraries and applications can often be installed directly from Github
  - ▶ Github issues, Pull requests and stars allow interaction with the community

## GIT: What?

- ▶ versions are connected via a tree-like graph, each node is a commit (=one version of the entire project)

# GIT: What?

- ▶ versions are connected via a tree-like graph, each node is a commit (=one version of the entire project)
- ▶ think of the commit as change ("patch") to the previous version, especially when writing the commit message

# GIT: What?

- ▶ versions are connected via a tree-like graph, each node is a commit (=one version of the entire project)
- ▶ think of the commit as change ("patch") to the previous version, especially when writing the commit message
- ▶ there are operations on the tree of commits: branch, rebase, cherry-pick, merge, copy commits to/from Github (= push/pull)

## GIT: How?

- ▶ command line (that's how we will do it)
- ▶ GUIs (e.g. gitk), IDE integration
- ▶ API (eg. Python or commit sha1 on first page of slides)

# GIT is difficult... :(



# GIT terminology

- ▶ version control system: a tool like GIT (alternatives: hg, SVN, ...)
- ▶ repository: a folder with files under version control, the complete history is stored in the subfolder .git
- ▶ commit: one version of the repository or equivalently the change made in this version
- ▶ commit message: a short message describing what has been changed in the commit and if not obvious, why it was a good idea to change it
- ▶ working directory (or working tree): the folder where files can be edited (don't confuse it with the commit history, which is tree-like!)

# Github: things to know

- ▶ fork: copy a GIT repository from a Github account to your own account, typically in order to get write permissions in the repository
- ▶ pull request: ask the original owner to copy commits from the forked repository back to the original repository
- ▶ repository collaborators: to allow others to write to your repository, invite them as collaborator via the Github webpage of the repository:  
Settings/Manage access/Invite a collaborator

# GIT commands

- ▶ Copy a GIT repository from Github to your computer:  
`git clone https://github.com/lumbric/git-games/`
- ▶ Add a new file to the repository:  
`git add path/to/file`
- ▶ Commit a file, i.e. record the current version for the GIT history:  
`git commit path/to/file`
- ▶ Which changes/files have not yet been committed:  
`git status`
- ▶ Copy new commits from local machine to Github:  
`git push`
- ▶ Copy new commits from Github to local machine:  
`git pull`

## GIT display changes

- ▶ gitk has a nice tree viewer, but looks a bit old fashioned
- ▶ Github has a nice web interface, but no tree viewer
- ▶ on the command line use `git show` or `git diff`

# Discuss questions collected as homework

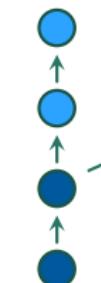
Questions in etherpad:

<https://yourpart.eu/p/lecture-scientific-computing>

# Bonus: Branches, merges and conflicts

Getting started  
with branching

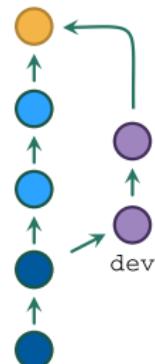
... a commit



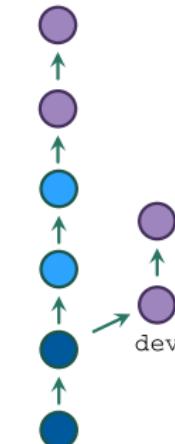
Three options to **merge** the changes from dev into master

- 1) A merge commit
- 2) Rebase
- 3) Squash and merge

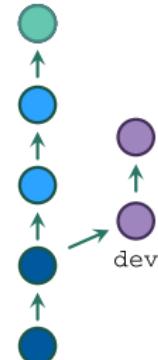
1) A merge commit



2) Rebase



3) Squash and merge



## Bonus: Merge conflicts

- ▶ You run `git pull` or `git merge` or `git rebase` and then see the error:  
`CONFLICT (content): Merge conflict in example.py`  
`Automatic merge failed; fix conflicts and then commit the result.`

## Bonus: Merge conflicts

- ▶ You run `git pull` or `git merge` or `git rebase` and then see the error:  
CONFLICT (content): Merge conflict in `example.py`  
Automatic merge failed; fix conflicts and then commit the result.
- ▶ Look for lines like these in `example.py`:  

```
<<<<< HEAD
some change
=====
some other change
>>>>> other
```
- ▶ edit the file, delete the markers and the parts you don't need

## Bonus: Merge conflicts

- ▶ You run `git pull` or `git merge` or `git rebase` and then see the error:  
CONFLICT (content): Merge conflict in `example.py`  
Automatic merge failed; fix conflicts and then commit the result.

- ▶ Look for lines like these in `example.py`:

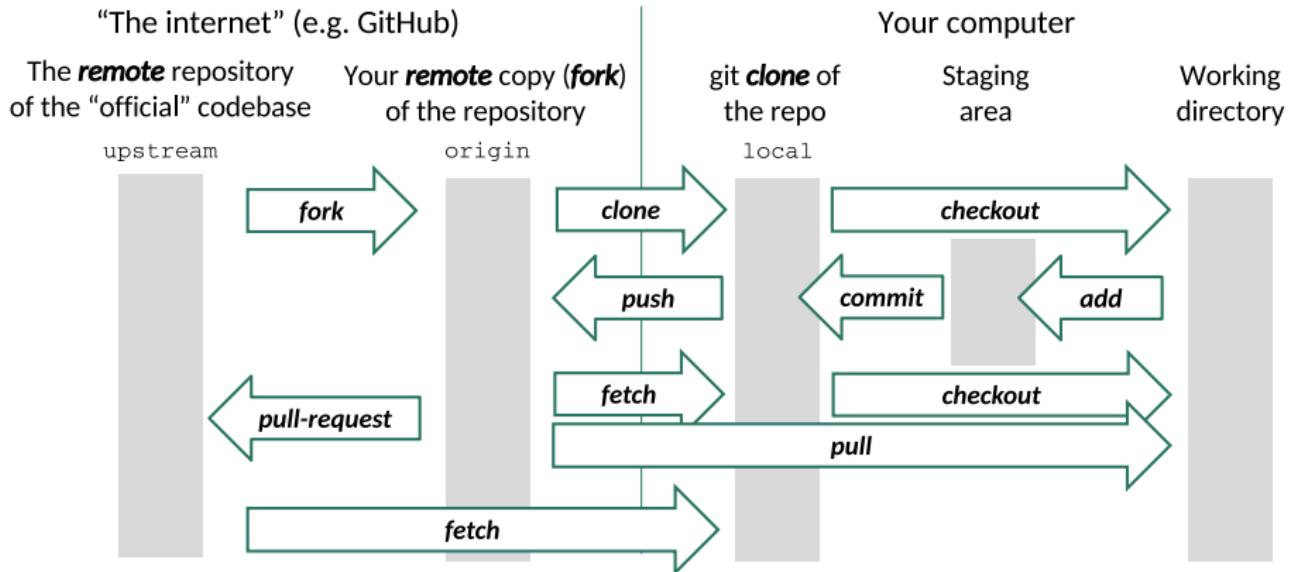
```
<<<<< HEAD  
some change  
=====
```

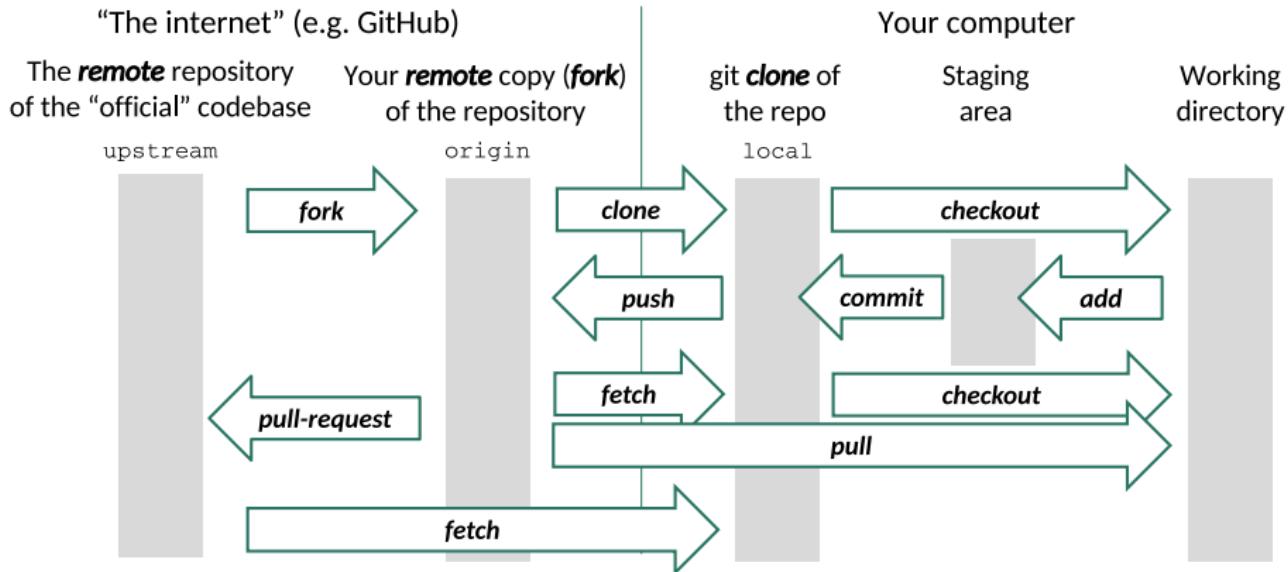
```
some other change  
>>>>> other
```

- ▶ edit the file, delete the markers and the parts you don't need
- ▶ run `git add example.py` and `git commit`

# Bonus: GIT workflow



# Bonus: GIT workflow



Source: Slides for Open Source Energy System Modeling by Daniel Huppmann, CC-BY 4.0

## Bonus: More topics

Also good to know, but probably beyond the scope of this lecture today...

- ▶ what is the GIT sha1 / hash?
- ▶ licenses and open source
- ▶ the `.gitignore` file
- ▶ different ways of reverting, see also [ohshitgit.com](http://ohshitgit.com)
- ▶ what does `git add` actually do?
- ▶ how does a pull request work exactly?

## Exercise

# Exercise

- ▶ Go to <https://github.com/lumbric/git-games/>
- ▶ read the instructions in README.md and choose a game (e.g. tic-tac-toe)
- ▶ follow the instructions in the README.md of the game you choose, e.g. Tic-tac-toe

## Homework assignment

# Homework assignment

- ▶ decide one maintainer for each group
- ▶ maintainer only: fork the exercise repository  
<https://github.com/inwe-boku/homework-scientific-computing>
- ▶ add other group members as collaborators to the forked repository in Github
- ▶ one group member only: create a hello world program or some other simple file with code or text file, add and commit it and push it
- ▶ other group members: change something (e.g. add another print statement) and commit the changes and push them to Github
- ▶ bonus points if you manage to get a merge-conflict on purpose and resolve it correctly