Programming fundamentals with Python

Session 2

Pepe García jgarciah@faculty.ie.edu

2020-04-20

Plan for today

- Learn a bit about CLI tools

Plan for today

- Learn a bit about CLI tools
- Learn about version control systems

Plan for today

- Learn a bit about CLI tools
- Learn about version control systems
- Introduction to Git

Command line

The command line allows users navigating the computer and managing it in the same way a **Graphical User Interface** can.

```
$ ls -l


total 0
drwx------+  5 pepe  staff   160 Nov 11 02:22 Desktop
drwx------+  3 pepe  staff    96 Nov 10 21:07 Documents
drwx------+ 12 pepe  staff   384 Nov 11 16:32 Downloads
drwx------@ 66 pepe  staff  2112 Nov 11 15:54 Library
drwx------+  4 pepe  staff   128 Nov 11 09:18 Movies
drwx------+  3 pepe  staff    96 Nov 11 02:22 Music
drwx------+  4 pepe  staff   128 Nov 10 23:50 Pictures
```

Command line

Disclaimer

In this slide set, every time you see a **$** at the beginning of the line it means that that's a command to be written in the terminal.

Command line

Disclaimer

In this slide set, every time you see a **$** at the beginning of the line it means that that's a command to be written in the terminal.

Disclaimer 2

If you're on Mac, we will use the **Terminal** for today's session, if you're on Windows, please open **Git Bash**.

Listing files

We can **list files** in a folder using the **ls** command (**dir** command on Windows)

```
$ ls
```

```
Desktop   Documents Downloads Library   Movies    Music     Pictures  Public    opt
```

Changing directories

We change directories using **cd**.

```
$ ls

Desktop   Documents Downloads Library   Movies    Music     Pictures  Public    opt

$ cd Desktop
```

Changing directories

We can go to *upper* directories using **cd ..**

```
$ ls

Desktop   Documents Downloads Library   Movies    Music    Pictures  Public    opt

$ cd Desktop

$ cd ..

$ ls

Desktop   Documents Downloads Library   Movies    Music    Pictures  Public    opt
```

Getting current directory

We can see where we are with the **pwd** command (**echo %cd%** in Windows)

```
$ pwd
/Users/pepe

$ cd Desktop

$ pwd
/Users/pepe/Desktop
```

pwd stands for print working directory

Creating directories

One can create directories using the **`mkdir`** command:

```
$ pwd
/Users/pepe

$ mkdir hello_dolly

$ cd hello_dolly

$ pwd
/Users/pepe/hello_dolly
```

Version control

Version control is the process of handling programs, versions, changes, and differences in files.

With **version control** systems we can see:

- **Who** made changes

The version control system we're going to use in this course is **git**.

Version control

Version control is the process of handling programs, versions, changes, and differences in files.

With **version control** systems we can see:

- **Who** made changes
- To **which files**

The version control system we're going to use in this course is **git**.

Version control

Version control is the process of handling programs, versions, changes, and differences in files.

With **version control** systems we can see:

- **Who** made changes
- To **which files**
- **When** did they do it

The version control system we're going to use in this course is **git**.

Version control

Version control is the process of handling programs, versions, changes, and differences in files.

With **version control** systems we can see:

- **Who** made changes
- To **which files**
- **When** did they do it
- **Why** did they do it

The version control system we're going to use in this course is **git**.

Installing Git

If you don't have it installed, you can get it from **https://git-scm.com/downloads**

Installing Git

Disclaimer

If you're on Mac, we will use the **Terminal** for today's session, if you're on Windows, please open **Git Bash**.

Git concepts

Git terminology can be very broad, but we'll focus on the parts that matter

Working directory

The **working directory** is the folder in which our code will be. The contents of this folder will be controlled by **git**.

Staging area

Whenever we're happy about the state of a file, we move it to the **staging area**. In the **staging area** we save files that are ready to be saved.

Local repository

The **local repository** is the place in which we store all the changes made to all the files of our projects, over time.

Creating our first repository

Practice (5 mins)

- Create a folder called **my-first-repo** in your desktop
- Navigate to it using the terminal (**cd**)
- Open **spyder**, create a python file and save it in **my-first-repo** folder
- In the terminal, initialize the repository with **git init**

Git operations

We can always see the status of our repository:

```
$ git status

On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

file.py

nothing added to commit but untracked files present (use "git add" to track)
```

Git operations

We can use **git add file.py** to add the file to the staging area, in which we store the files ready to be commited.

```
$ git add file.py

$ git status

On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

new file:   file.py
```

Git operations

When there is a meaningful change we want to save, we use **git commit** to save it to our local repository.

We use **git commit -m “message”** and try to use a meaningful description of the changes we just made.

```
$ git commit -m "add file.py to git"
[master (root-commit) 123cd8b] add file.py to git
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.py
```

Git concepts

Git operations

One of the most powerful features of **git** is handling changes. Let's add this function to our **file.py**.

```python
def func(a, b):
    return a + b
```

Git operations

And let's see the changes now! **git diff**

Git will show the lines we added with a **+ sign** before, and those we removed with a **- sign**

```
$ git diff


diff --git a/file.py b/file.py
index e69de29..c09bd0e 100644
--- a/file.py
+++ b/file.py
@@ -0,0 +1,3 @@
+
+def func(a, b):
+    return a + b
(END)
```

Commit the last changes

Practice

Now, let's commit our latest changes

Git operations

Other of the cool features of **git** is watching the history of our repository. With **`git log`** we will see a log of all the changes that happened to our repository.

```
$ git log


commit 123cd8b45ae31065cdd7cf0ecd8ce83b444886db (HEAD -> master, origin/mas
Author: Pepe García <pepe@pepegar.com>
Date:   Mon Nov 11 23:55:49 2019 +0100


add file.py to git
```

Github classroom

At this point, everybody should have an account on Github. If you haven’t yet created one, please go and create one (you can follow the instructions in the second link of the bibliography).

Github classroom

Github classroom is the software we will use in this term to handle, submit, and review assignments. You will receive links like this one, and you'll need to accept the assignments:

https://classroom.github.com/a/csu9qbqV

Bibliography

Images and inspiration drawn from

How to teach Git

Codecademy - get started with Git and Github

Learn git concepts, not commands