Programming fundamentals with Python

Version Control - Git

Pepe García jgarciah@faculty.ie.edu

SCHOOL OF SCIENCE & TECHNOLOGY

Plan for today

In this session we will:

- learn about version control and why we need
- learn about git

Version control

Why do we need version control software?

Have you ever found yourselves with a bunch of copies of a file (an assignment maybe?) that you save to not loose what you've created?

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**.

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 **VS Code**, 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 committe

file.py

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

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)
```

In the picture we can see we added three lines

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 -> maste
Author: Pepe García <pepe@pepegar.com>
Date:   Mon Nov 11 23:55:49 2019 +0100

add file.py to git
```

Github

Now, let's create an account in Github!

Go to github.com and create an account. (if you've one already, that's OK)

Bibliography

Images and inspiration drawn from

How to teach Git

Learn git concepts, not commands