

# Programming fundamentals with Python

## Session 3

Pepe García [jgarciah@faculty.ie.edu](mailto:jgarciah@faculty.ie.edu)

2020-04-20

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

Let's **create a repository on Github!**

Let's **create a repository on Github!**

- Log into your github.com account


Let's **create a repository on Github!**

- Log into your github.com account
- Create a new repository

# Github - create a new repository

Owner

Repository name \*

 pepegar ▾ / my-first-repo ✓

Great repository names are short and memorable. Need inspiration? How about [super-happiness](#)?

Description (optional)



**Public**

Anyone can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.


Skip this step if you're importing an existing repository.



**Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

**Grant your Marketplace apps access to this repository**

You are subscribed to 2 Marketplace apps



**Travis CI**

Test and deploy with confidence



**Mergify**

Pull requests automation service

Create repository

# Github - linking our local repo with Github

Now that we have created this repo, it's identified with a unique URL, we'll use the **https** url to link our local and remote repositories.

## Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** `https://github.com/popogor/my-first-repo.git`



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

# Github - linking our local repo with Github

In order to link our local repo with Github, we will need to **set a new remote** in our local repository:

```
$ git remote add origin https://github.com/popogor/my-first-repo.git
```

This will tell **git** where is the **remote repository** to which we will upload our changes.



# Github - pushing code!

Now that we have everything set, we're ready to **push** our code to Github!

```
$ git push origin master
```

```
Enumerating objects: 3, done.
```

```
Counting objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 608 bytes | 608.00 KiB/s, done.
```

```
Total 3 (delta 0), reused 0 (delta 0)
```

```
To https://github.com/popogor/my-first-repo.git
```

```
* [new branch]      master -> master
```

**git push** sends changes from the local repository to the remote one. It's the way we have to *upload* code to github.

## Interlude - branches

As you've seen, the latest parameter that we passed to push is **master**. That's the name of the default **branch** in git.

Branches are a way of being able to work in parallel features on the same repository. We're not going to use them in this course, but if you're interested, there's a link in the bibliography.

The reverse of **pushing** is **pulling**, and that will allow us to bring all changes **from the remote repository to our local repository**.

```
$ git pull origin master
```

# Solving conflicts

*Conflicts* occur naturally when coding. Mostly when we do collaborate with others.

Let's introduce a conflict and fix it ourselves!

# Solving conflicts

First, in our local copy of **my-first-repo**, let's change the function we had to:

```
def func(a, b):  
    return a - b
```

A simple change, just modify it so it subtracts instead of adding.

# Solving conflicts

And then **git add** and **git commit** it.

```
$ git add file.py
```

```
$ git commit -m "change function and make it subtract"
```

```
[master ae46fc3] change function and make it subtract  
1 file changed, 1 insertion(+), 1 deletion(-)
```

# Solving conflicts

Now let's simulate the changes someone else would make in github.

# Solving conflicts

Then, in our local repository, let's **git pull**

```
$ git pull origin master
```

```
remote: Enumerating objects: 5, done.
```

```
remote: Counting objects: 100% (5/5), done.
```

```
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
Unpacking objects: 100% (3/3), done.
```

```
From https://github.com/popogor/my-first-repo
```

```
* branch                master      -> FETCH_HEAD
```

```
4c659b6..e441a78 master    -> origin/master
```

```
Auto-merging file.py
```

```
CONFLICT (content): Merge conflict in file.py
```

```
Automatic merge failed; fix conflicts and then commit the result.
```



# Solving conflicts

First of all, we need to see the conflicts

```
$ git diff
```

```
diff --cc file.py
```

```
index 64ad20f,84e4d51..0000000
```

```
--- a/file.py
```

```
+++ b/file.py
```

```
@@@ -1,3 -1,3 +1,7 @@@
```

```
def func(a, b):
```

```
++<<<<<< HEAD
```

```
+     return a - b
```

```
++=====
```

```
+     return a * b
```

```
++>>>>>> e441a78ff5f91b986f0da3afddbb7a7a01ee1859
```

# Solving conflicts

As you've seen in the last **git diff**, there are incompatible changes in our function. We need to make a decision, let's say that we want to keep the multiplication.

We finish the process by doing **git add** and **git commit** after solving the conflict, telling git we're happy with the result.

```
$ git add file.py
```

```
$ git commit -m "merged conflict in file.py"  
[master 1abfd41] merged conflict in file.py
```

# Solving conflicts

```
$ git log
```

```
commit 1abfd4151a6d44e3268c59b56065730676e545db (HEAD -> master)
```

```
Merge: ae46fc3 e441a78
```

```
Author: Pepe García <pepe@pepegar.com>
```

```
Date: Tue Nov 12 01:55:00 2019 +0100
```

```
merged conflict in file.py
```

```
commit e441a78ff5f91b986f0da3afddbb7a7a01ee1859 (origin/master)
```

```
Author: popogor <46658846+popogor@users.noreply.github.com>
```

```
Date: Tue Nov 12 01:38:13 2019 +0100
```

```
change function and make it multiply
```

```
commit ae46fc37dbbf344f3ee4e5d189bb0714a543dd0f
```

**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>

# Cloning a project

```
$ git clone https://github.com/pepegar/my-first-repository.git
```

```
Cloning into 'my-first-repository'...
```

```
remote: Enumerating objects: 16, done.
```

```
remote: Total 16 (delta 0), reused 0 (delta 0), pack-reused 16
```

```
Unpacking objects: 100% (16/16), done.
```

we use **git clone** to copy a repository to our local computer.

Images and inspiration drawn from

- **How to teach Git**
- **Codecademy - get started with Git and Github**
- **Learn git concepts, not commands**
- **Using branches**
- **Switch git branch**

My old set of slides:

- **<https://slides.com/pepegar/pfp-9>**