Medical Images
Course 2022-2023

Lab session 2
Gitlab
Python classes

Dani Tost

Summary

Last week we reviewed python. Today we'll introduce the concept of python classes

We also introduced gitlab. Today we'll learn by practice with it.

Remember that we'll use and specifically our teaching instance:
https://gitlab-gie.cs.upc.edu/

Part I

Python classes

Python classes

Python is an object-oriented programming language. All objects in python are organized into **classes.** An object of a class is called an **instance** of the class. It has **attributes** and **methods**.

Example 1:

```
>>> l = [2, 5, -1, -8, 7]
>>> l.sort()
>>> l
[-8, -1, 2, 5, 7]
>>> type(l)
<class 'list'>
>>> len(l)
5
```

`sort` is a **method** of class `list`

A `list` is a **class**
`l` is an **instance** to class `list`
The class `list` supports the **function** `len` and the **indexing** mechanism `l[i]`

Python classes

Example 2:

```
>>> from datetime import date
>>> d = date(2022, 9, 27)
>>> d.day
27
>>> d.month
9
>>> d.year
2022
>>> type(d)
<class 'datetime.date'>
```

`day`, `month` and `year` are attributes of `d`
`d` is an instance to class `date`

Python classes

All objects in python are organized into **classes.** An object of a class is an **instance** of the class. It has **attributes** and **methods**.

Instanciation (creation of an instance, a variable of a class)

```
from module import classname
instance_name = classname(parameters)
```

Access to an attribute

```
instance_name.attribute_name
```

Assignment of a value to an attribute

```
instance_name.atribute_name = value
```

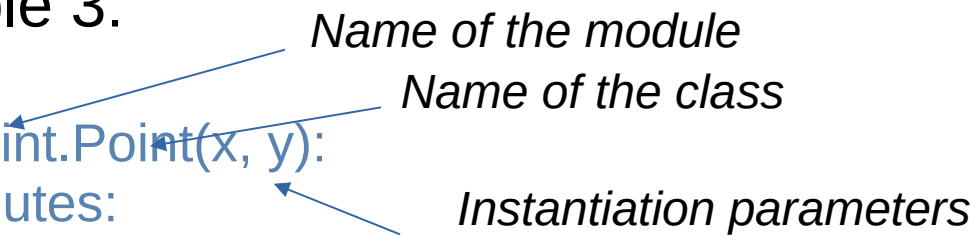
Invocation of a method:

```
instance_name.method_name(parameters)
```

Specification of a class

The specification of a class describes how to use and how to implement it.

Example 3:




```
class point.Point(x, y):
    Attributes:
        x: the x coordinate
        y: the y coordinate
    Methods:
        dist(q)
            Returns the distance between the point and q
This class supports the function str() and the comparison
==.
```

Usage:

```
>>> from geom import Point
>>> p = Point(8, 3)
>>> p2 = Point(1, 0)
>>> p.dist(p2)
7.6157731058639
```

Download point.py and try to use it. Create instances, access to attributes, invoke the method `dist`, apply the function `str`. Check that two different instances with the same attributes values are considered as equal (with operator `==`). Use the doctest tests_point.txt.

Example

Example 4: the module numpy provide different mathematical classes. The class `numpy.ndarray` represents n-dimensional arrays and it can be used to represent raster images

numpy.ndarray

class **numpy.ndarray**(*shape, dtype=float, buffer=None, offset=0, strides=None, order=None*) [source]

An array object represents a multidimensional, homogeneous array of fixed-size items.

Attributes

dtype : *dtype object*

Data-type of the array's elements.

size : *int*

Number of elements in the array.

Methods

all([axis, out, keepdims, where])	Returns True if all elements evaluate to True.
any([axis, out, keepdims, where])	Returns True if any of the elements of *a* evaluate to True.

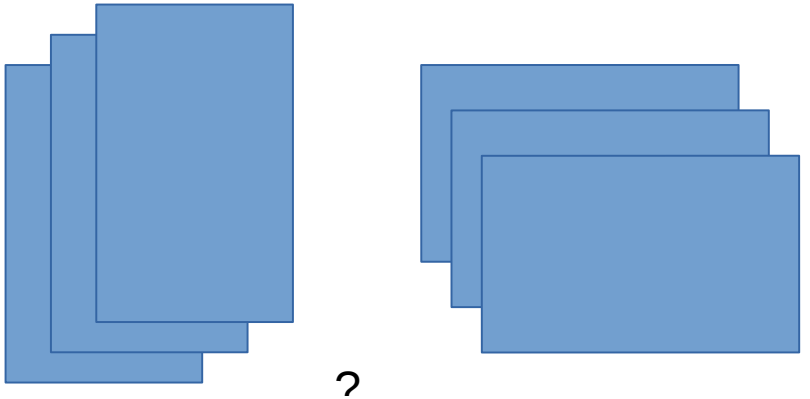
Read the **specification** in the documentation.
You should be able to know how to **create an instance**, access and modify **attributes** and i**nvoke methods.**

Example

Example 4: the class `numpy.ndrray` to represent n-dimensional arrays

```
>>> import numpy as np
>>> ima = np.ndarray(shape=(100, 200, 3), dtype=np.uint8)
>>> ima.shape
(100, 200, 3)
>>> ima.size
60000
>>> ima.ndim
3
>>> ima.fill(8)
>>> ima[0,0,0]
8
>>> ima[10:50, 10:190]=[255, 0, 0]
>>> ima[50:90, 10:190]=[255, 125, 125]
```

ima is a ndarray of 100 x 200x 3 and 1 byte per value. Which is the width and which is the height? How many channels does it have? How do you imagine it?



?

https://numpy.org/devdocs/reference/generated/numpy.ndarray.html?highlight=numpy%20ndarray#numpy.ndarray

https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html

Example

Example 4: to visualize or render the array we'll use the library matplotlib. We'll create a figure `(fig)` composed by 1 subplot `(ax)` in which we will show the array without axes.

```
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots(1, 1)
>>> ax.axis('off')
>>> ax.imshow(ima)
<matplotlib.image.AxesImage object at 0x7ff414210ee0>
>>> plt.show()
```

200 = number of columns = width

100 = number of rows = height

Implementation of classes

During the course you'll have to implement or modify the implementation of classes. To do so, you have to read the **specification** and create the code to make the implemented class behave as described in the specification (same **attributes** and **methods**, same functionality).

The simplified structure of a class definition is:

```
class name:

    def __init__(self, parameters):
        ....
    def method1(self, parameters):
          ....
```

Some methods start and end with __ . They are **special** methods, invoked in a different way. The method **`__init__`** is invoked when creating an instance. It creates and initializes the attributes.

The headers of the methods have a first parameter «**`self`**» that represents the instance with which the method is invoked.

Example

```
class point.Point(x, y)
    Attributes:
       x : the x cartesian coordinate
       y : the y cartesian coordinate
   Methods
      dist(p2)
         Returns the distance to point p2
   Supports the function str that returns the string '(x, y)'
   Supports the comparison == and !=.
```

This is the **specification.** It's text.

To implement it, we need to create a module (file) called `point.py`. Why this name? Check in the specification.

We will create the class `Point` inside this module

Example

Look at the file `point.py`. Do you understand? We need a way to refer to the instances that we will create and on which we will invoke methods. We refer to them as **self.**

Thus, the headers of all the methods have a first parameter «**`self`**» that represents the instance with which the method is invoked

```
from math import sqrt
class Point:
    def __init__(self, xx, yy):
        self.x = xx
        self.y = yy
    def dist(self, q):
        return sqrt((self.x-q.x)**2 + (self.y -q.y)**2)
```

```
>>> from point import Point
>>> p1 = Point(3, 4)

def __init__(self, x, y)
```

```
>>> p1.dist(p2)

def dist(self, q):
```

Example

What does «supports the function» mean?

We can do:

```
>>> from point import Point
>>> p1 = Point(2, 3)
>>> p2 = Point(2, 3)
>>> p1 == p2
True
>>> p1 != p2
False
>>> str(p1)
'(2, 3)'
>>>
```

```
class Point:
    def __init__(self, xx, yy):
        self.x = xx
        self.y = yy

    def dist(self, q):
        return sqrt((self.x-q.x)**2+(self.y-q.y)**2)

    def __str__(self):
        return '({}, {})'.format(self.x, self.y)

    def __eq__(self, q):
        return self.x == q.x and self.y == q.y
```

For each function supported by a class we need to implement the corresponding **special** method.

Some special methods

Method	Use
`a.__getitem__ (expr)`	`a[expr]`
`a.__setitem__(expr, val)`	`a[expr] = val`
`a.__len__()`	`len(a)`
`a.__str__()`	`str(a)`
`a.__add__(b)`	`a + b`
`a.__sub__(b)`	`a – b`
`a.__contains__(x)`	`x in a`
`a.__le__(b)` (lt/eq/ne/gt/ge)	`a <=b` (<,==, !=, >, >=)
`a.__iter__()`	`iter(a)`

Class methods

Class methods are defined at the class level, not at the instance level. They have many uses. In the course, we'll use class methods as alternative ways of creating instances.

To invoke a class method:

```
from module import classname
instance_name = classname.classmethod(parameters)
```

Example:

```
>>> from datetime import date
>>> d1 = date(2024, 7, 12)
>>> d2 = date.today()
```

Use of `__init__`

Use of class method `today`

Class methods

See this new specification of class point

class point.Point(x, y):

Class methods:

from_polar(radius, angle)

returns an instance from its polar coordinates with angle in radiants

from_dict(dpoint)

Returns an instance from a dictionary {‘x’:xcoord, ‘y’:ycoord}

Attributes:

x: the x coordinate

y: the y coordinate

Methods:

dist(q)

Returns the distance between the point and q

to_dict()

Returns a dictionary: {‘x’: x-coordinate, ‘y’:y-coordinate}

Usage:

```
>>> from point import Point
>>> p = Point(2, 3)
>>> str(p)
'(2, 3)'
>>> q = Point.from_dict({'x':5, 'y':8})
>>> str(q)
'(5, 8)'
>>> from math import pi
>>> r = Point.from_polar(1, pi/4)
>>> str(r)
'(0.7071067811865476, 0.7071067811865475)'
```

Class methods

Try to add the method to_dict and from_dict to the class. Test it with the new test file: tests-point-2.txt.

```python
from math import sqrt, cos, sin

class Point:
    def __init__(self, xx, yy):
        self.x = xx
        self.y = yy

    def dist(self, q):
        return sqrt((self.x-q.x)**2+(self.y-q.y)**2)

    def __str__(self):
        return '({}, {})'.format(self.x, self.y)

    def __eq__(self, q):
        return self.x == q.x and self.y == q.y

    def to_dict(self):
        return {'x':self.x, 'y': self.y}

    @classmethod
    def from_polar(cls, radius, angle):
        return cls(radius*cos(angle), radius*sin(angle))

    @classmethod
    def from_dict(cls, dict_coord):
        return cls(dict_coord['x'], dict_coord['y'])
```

← See the solution here

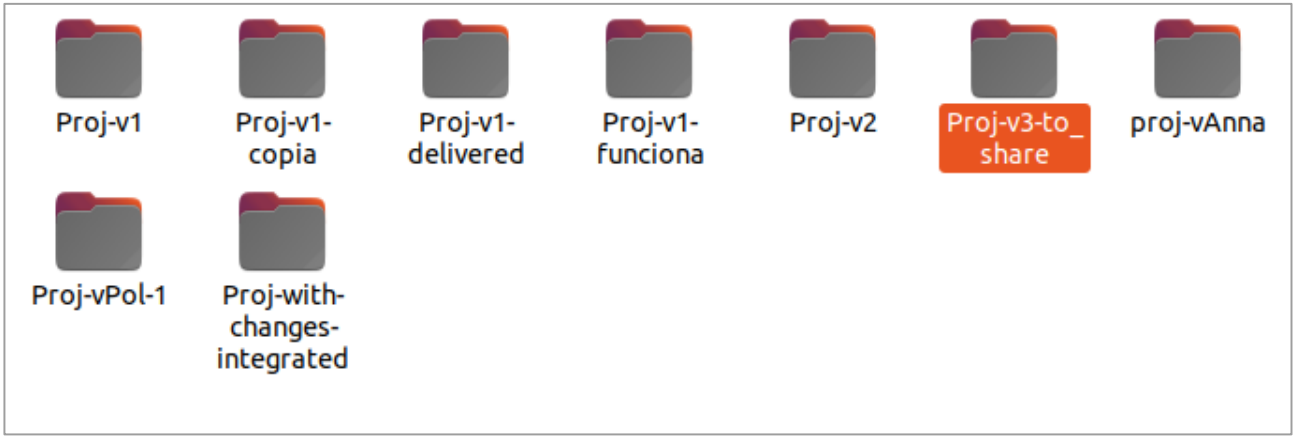
Part II

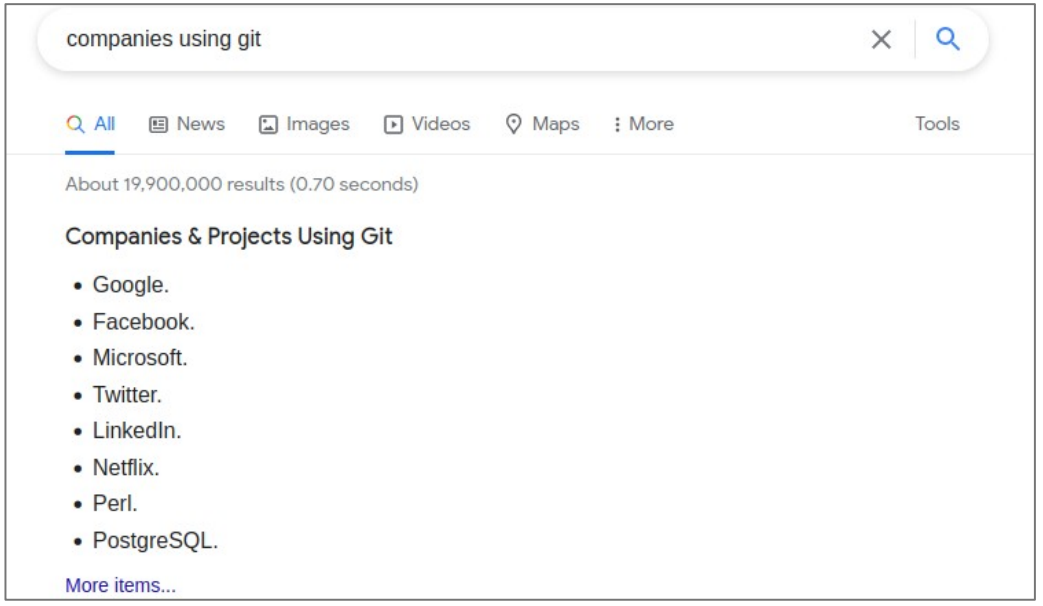
Git

Git

Why should we use Git?

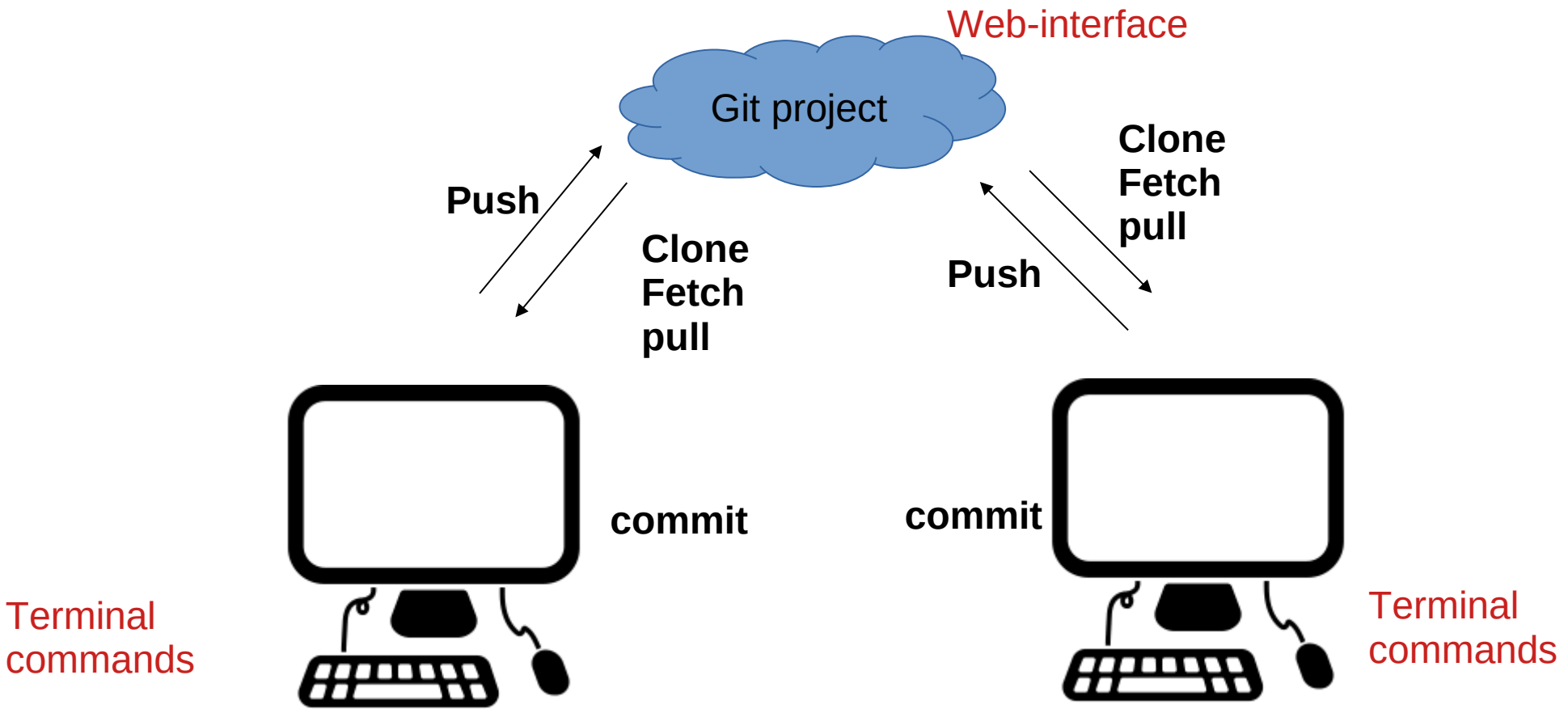
- To avoid this →
- Because it is used worldwide
- Because it is a very valued know-how
- Because it has a GNU license





Git workflow

Git stores the project in the cloud. Every contributor has a local version of the project that needs to be synchronized with the cloud version periodically (at each session at least).



https://docs.gitlab.com/

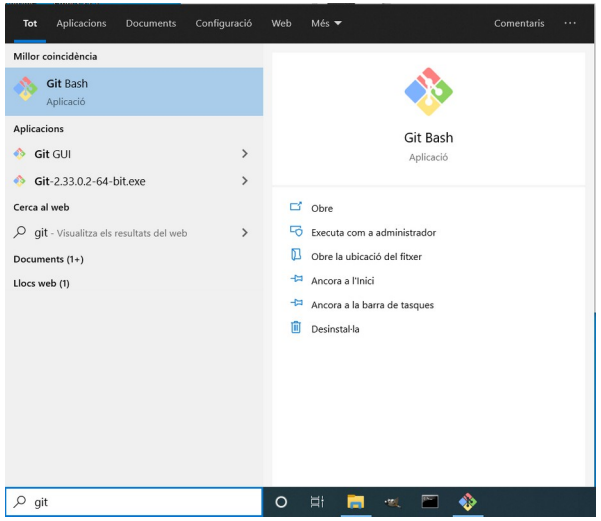
Git

Configuration

Once installed, configure your environment write down the following commands in a terminal:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

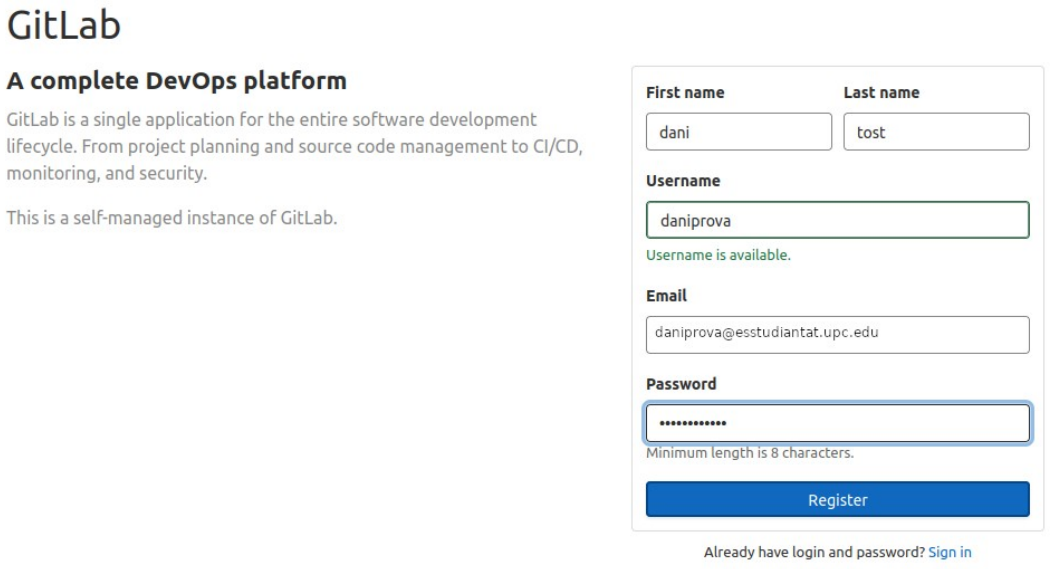
In windows, once git is installed, you'll have a **gitbash terminal** that is linux oriented. You can work in this bash to launch the commands in linux-style. Alternatively you can install a better terminal than cmd. The Mac-OS terminal is like a linux terminal.



Git projects

Git provides a GUI web-interface to manage the project. To access to it, depending on the level of privacy of the project, you may need a git account and have been granted the access.

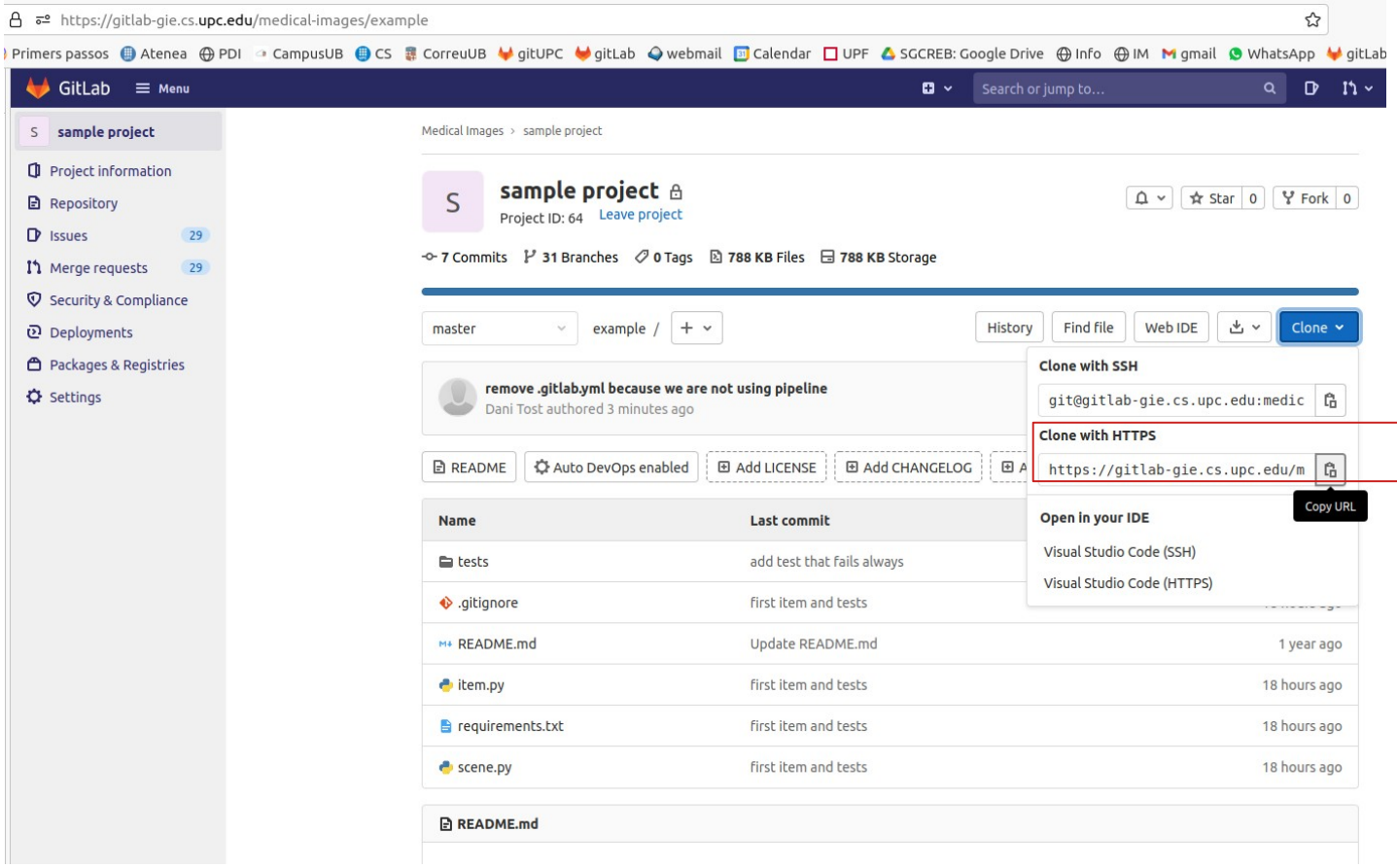
Let's follow the work flow with a sample project. In a web browser, go to its url.
You'll need to register to git with your **estudiant.upc.edu** address.



Now you should be able to login.

Git projects

You should see the example project:
https://gitlab-gie.cs.upc.edu/medical-images/example

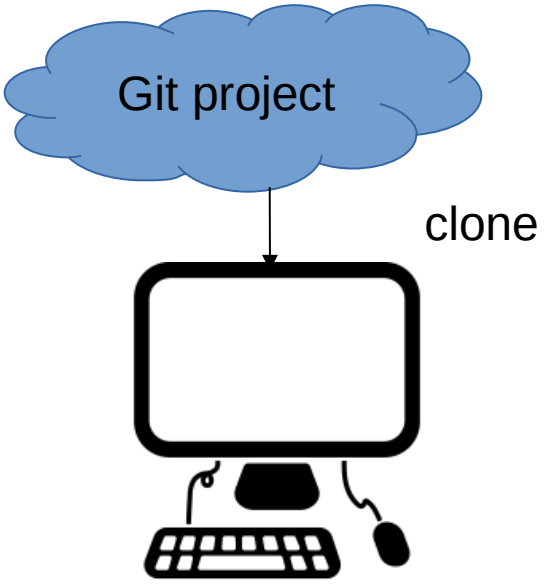


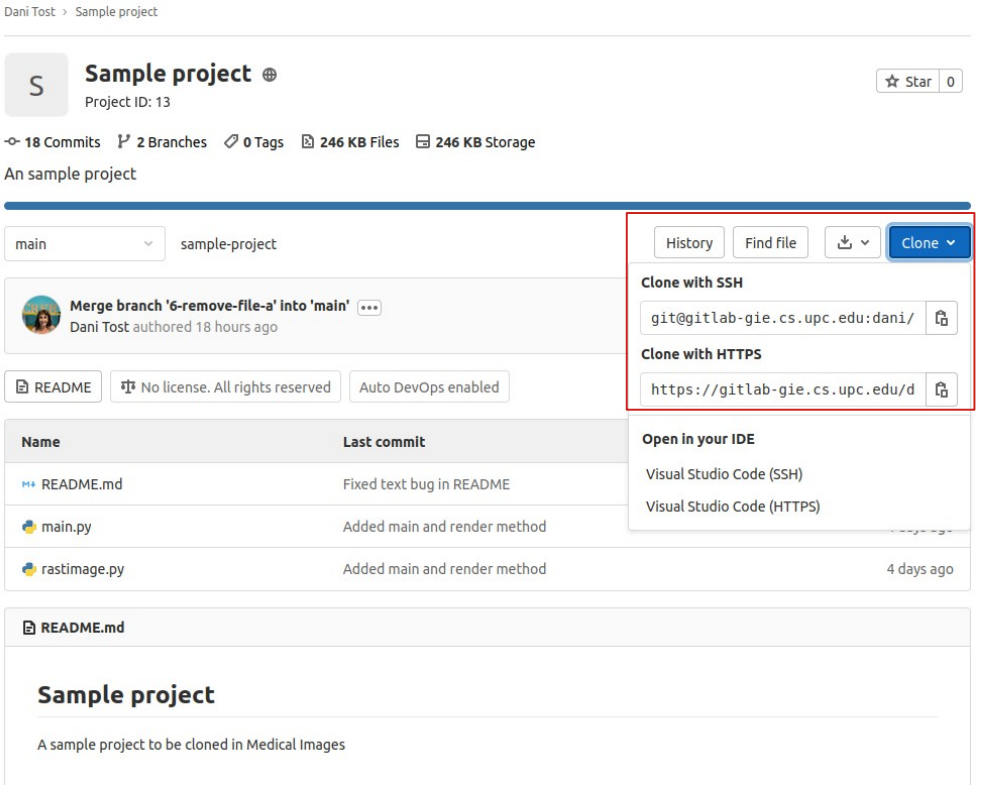
The url to clone the project

Check the issues. You'll probably have one assigned

Git workflow

Once the project is created, a contributor can **clone** it in his/her computer. You can do that using the protocol http or ssh (see https://docs.gitlab.com/ee/ssh/ for the second case)






```
$ mkdir IM
$ cd IM
$ git clone name_of_the_project
$ cd name_of the_project
  Your have all the files here
```

https://docs.gitlab.com/

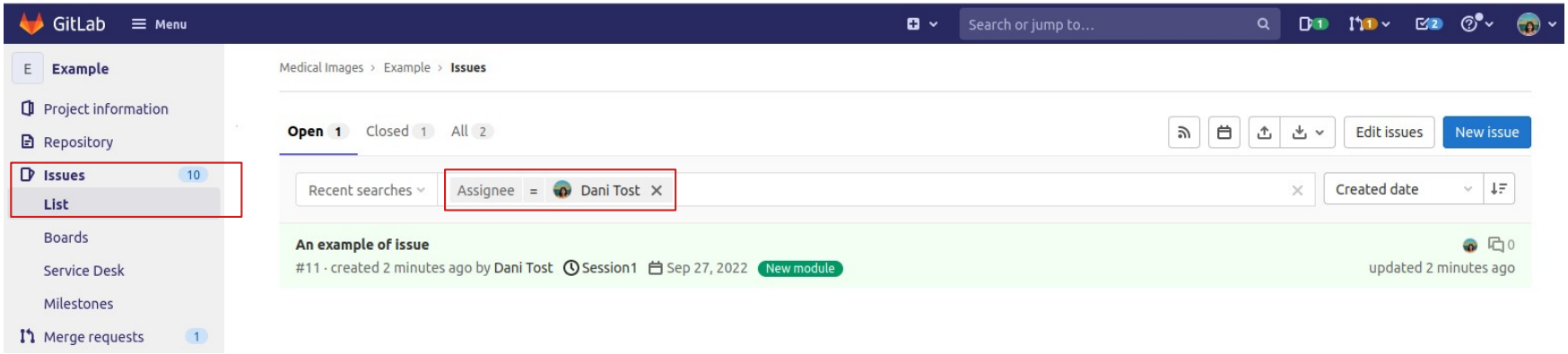
Git workflow

With the project cloned, you are ready to work.

You detect a need, an error or any other **issue** in the project or and you write it using the web UI. You specify the problem, the milestone, due date and so.
Be specific. This is a very important information!

I have already assigned an issue to 1 of the team members of each group. You should solve it together.

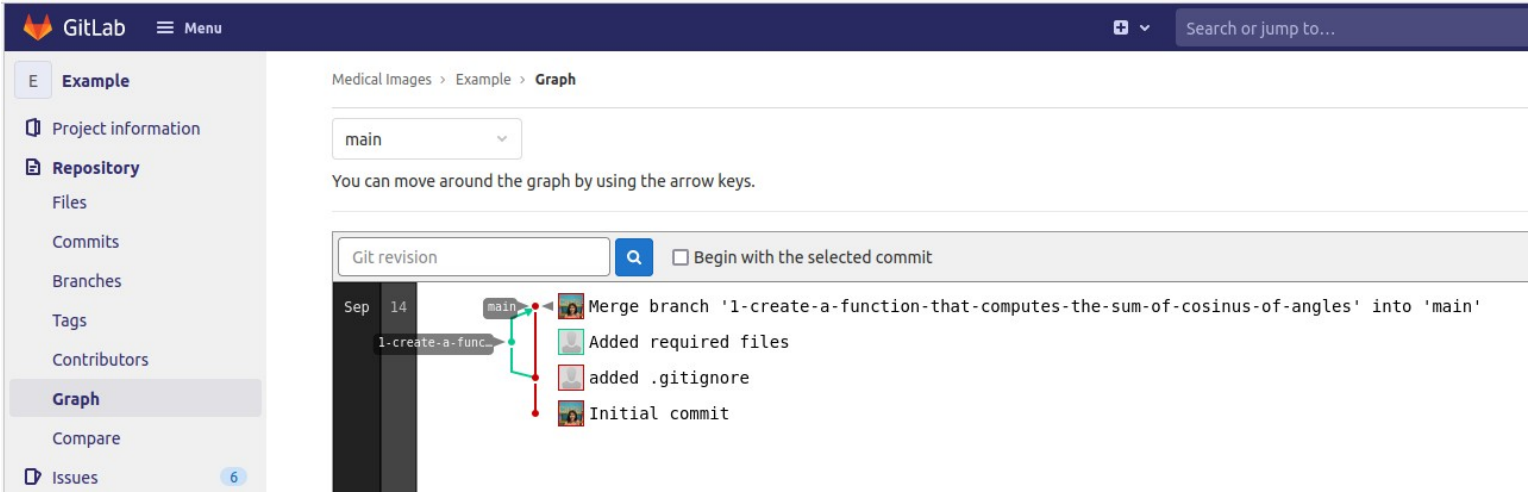
Check your assigned issues.



Git workflow

While you resolve an issue, you cannot bother other members of the team that are working in other issues. This is why you need to work in a separate branch of the code. When you are ready, you will merge it in the main (master) branch.

For instance here, I have solved issue 1 in a separate branch and when was satified I merged it into the main (master) branch



Git branches

To know in which branch you are:

```
$ git status
```

To change branch:

```
$ git checkout namebranch
```

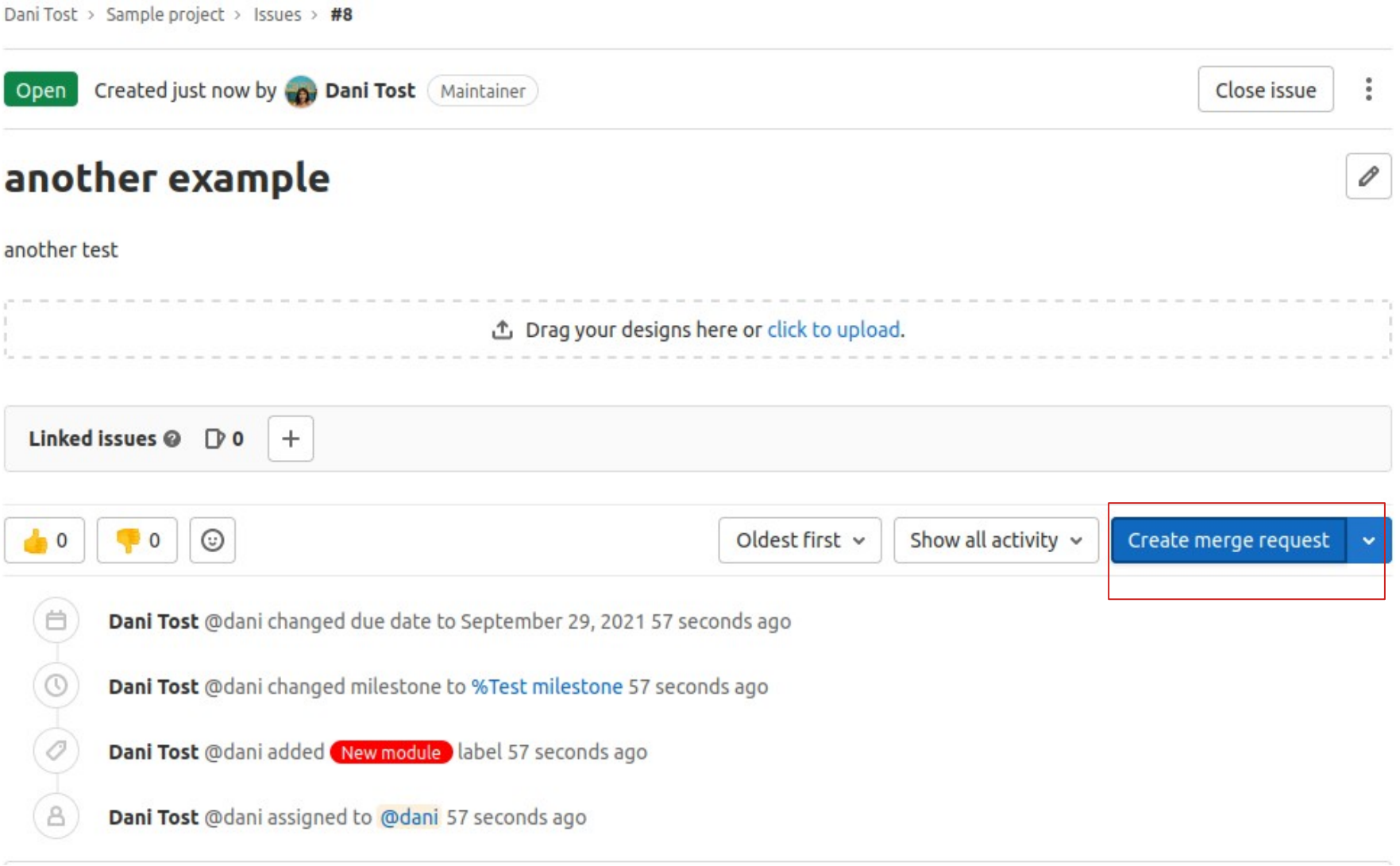
VIP

Before doeing anything make sure that you are in the correct branch!!!

```
$ git status
En la branca 1-create-a-function-that-computes-the-sum-of-cosinus-of-angles
La vostra branca està al dia amb «origin/1-create-a-function-that-computes-t
he-sum-of-cosinus-of-angles».

no hi ha res a cometre, l'arbre de treball està net
$ git checkout main
S'ha canviat a la branca «main»
La vostra branca està al dia amb «origin/main».
$ git status
En la branca main
La vostra branca està al dia amb «origin/main».

no hi ha res a cometre, l'arbre de treball està net
```

Git workflow

When you or another developer is ready to solve the **issue**, he/she creates a **Merge Request.** This will automatically create the branch in which you must work.



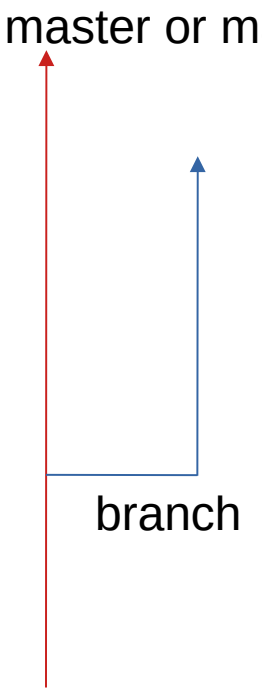
Git workflow

When you or another developer is ready to solve the **issue**, she creates a **Merge Request**.

Observe that the branch 8-another-example has been created, related to issue #8 its status is still draft.

Git workflow

Automatically, git will create a new **branch**. You will develop in this branch without bothering the rest of developers. The version in the master will keep stable and working.



In the terminal, in the sample-project directory do:

`$ git fetch` downloads the new branch and other files

```
De https://gitlab-gie.cs.upc.edu/dani/sample-project
 * [branca nova]     8-another-example -> origin/8-another-example
```

`$ git status` to know at which branch you are and the status of your files

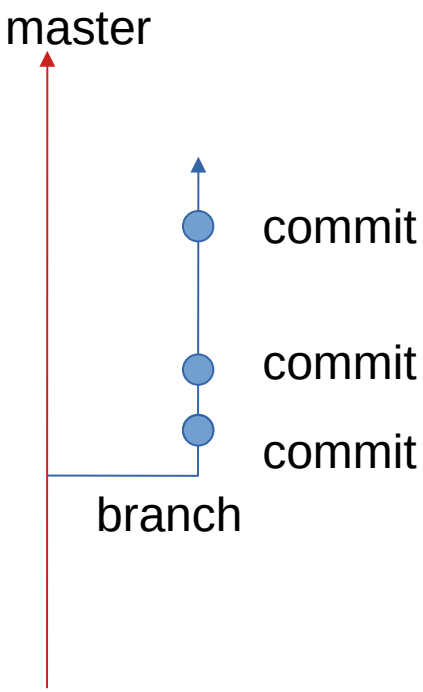
```
En la branca main
La vostra branca està al dia amb «origin/main».

no hi ha res a cometre, l'arbre de treball està net
```

`$ git checkout 8-another-example` to go to that branch

```
La branca «8-another-example» està configurada per a seguir la branca remota
S'ha canviat a la branca nova «8-another-example»
```

Git workflow

In the branch, we work. From time to time, when ready, and as often as possible, we **commit** the changes with a message. The **commit** is local. It is not uploaded to the cloud.




```
:$ git rm main.py
:$ git add point.py
:$ git status
La vostra branca està al dia

Canvis a cometre:
  (use "git restore --staged <file>..." to unstage)
        suprimit:         main.py
        fitxer nou:       point.py
        suprimit:         rastimage.py

Fitxers no seguits:
  (useu «git add <fitxer>...» per a incloure-ho en la comissió)
        point.py~
```

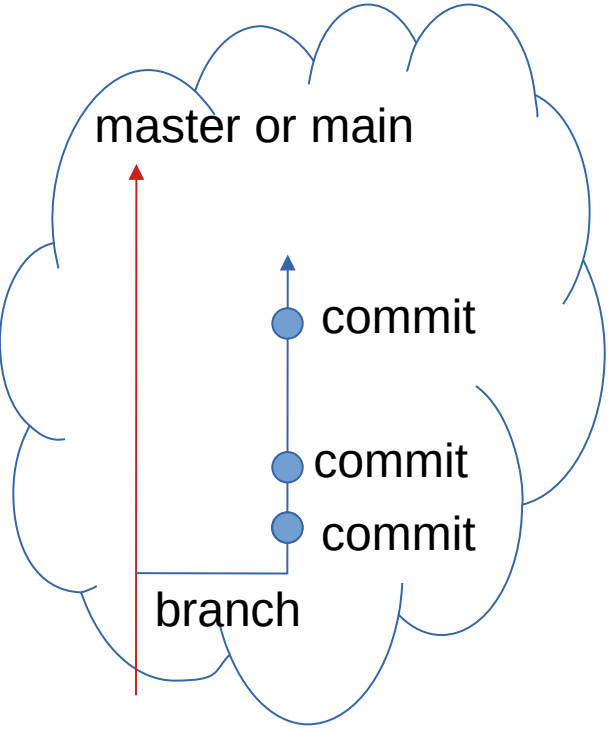
```
:$ git commit -m "this is an example"
[8-another-example 9a70a29] this is an example
 3 files changed, 28 insertions(+), 104 deletions(-)
 delete mode 100644 main.py
 create mode 100644 point.py
 delete mode 100755 rastimage.py
```

Edit a first version of the solution. Just the header of the function and pass. Try to commit and push. Modify and re-do the process.

```
$ git add file
$ git rm file
$ git commit –m «this is an example»
```

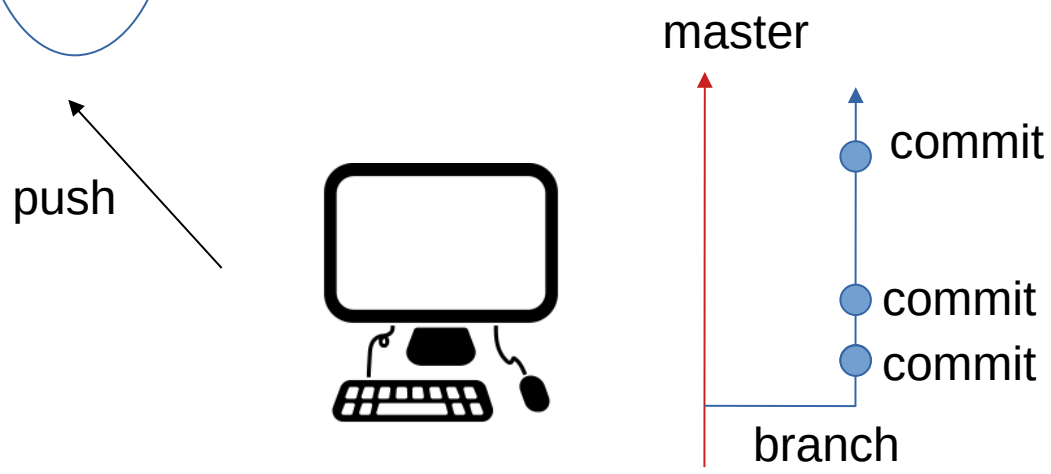
Git workflow

From time to time (at least once during esach work session) we **push** the changes to the cloud version.




```
$ git push
Username for 'https://gitlab-gie.cs.upc.edu': dani@cs.upc.edu
Password for 'https://dani@cs.upc.edu@gitlab-gie.cs.upc.edu':
S'estan enumerant els objectes: 4, fet.
S'estan comptant els objectes: 100% (4/4), fet.
Delta compression using up to 12 threads
S'estan comprimint els objectes: 100% (3/3), fet.
S'estan escrivint els objectes: 100% (3/3), 558 bytes | 558.00 KiB/s, fet.
Total 3 (diferències 0), reusat 0 (diferències 0)
remote:
remote: View merge request for 8-another-example:
remote:   https://gitlab-gie.cs.upc.edu/dani/sample-project/-/merge_requests/9
remote:
To https://gitlab-gie.cs.upc.edu/dani/sample-project.git
   ab7dc97..9a70a29  8-another-example -> 8-another-example
```

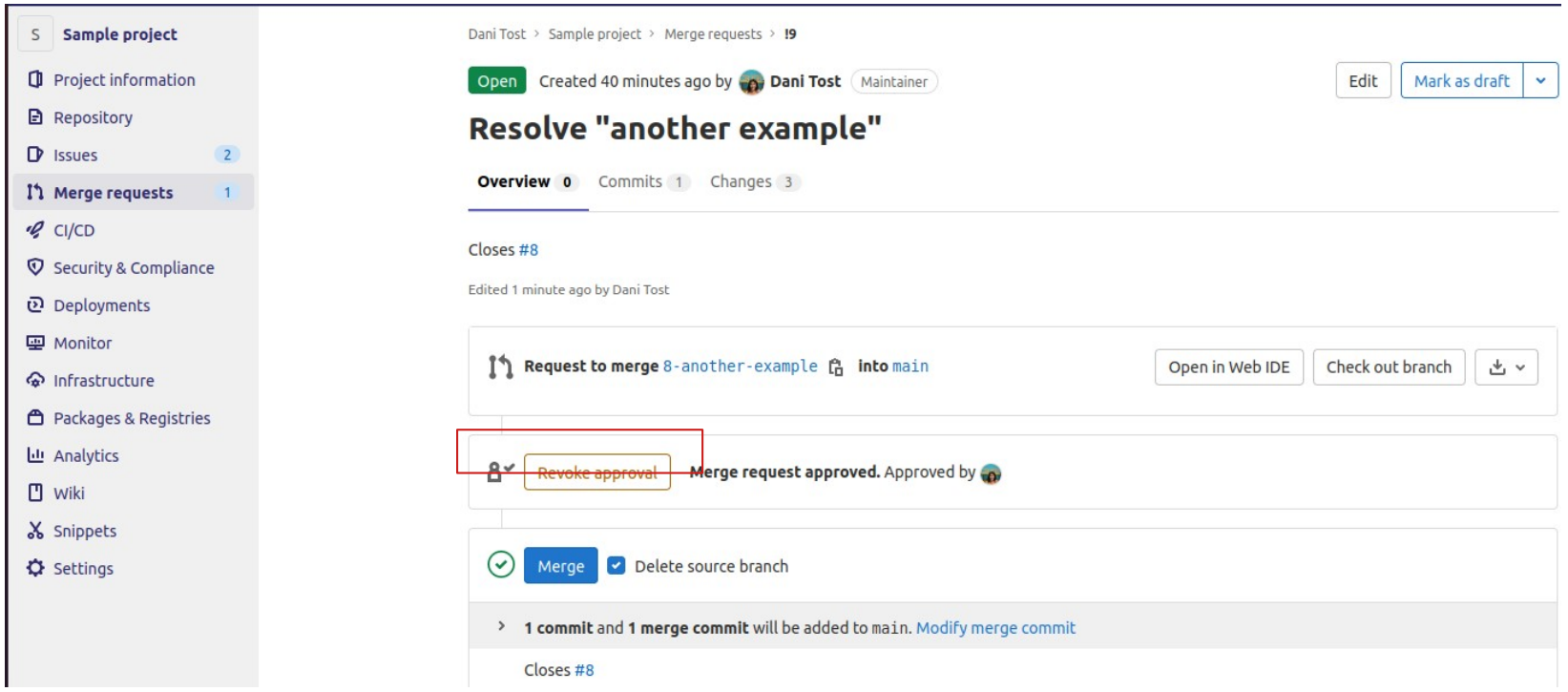
`$ git push`



Git workflow

When we are ready, we **merge** the branch to the master. We may ask another developer to **review** the branch before. During the merging **pipeline** we may allow some **tests** to be passed and prevent from merging erroneous files.

We do that in the web GUI. Aprove (if it is your role), remove the draft status and merge. In the lab work, I will the one who approves.



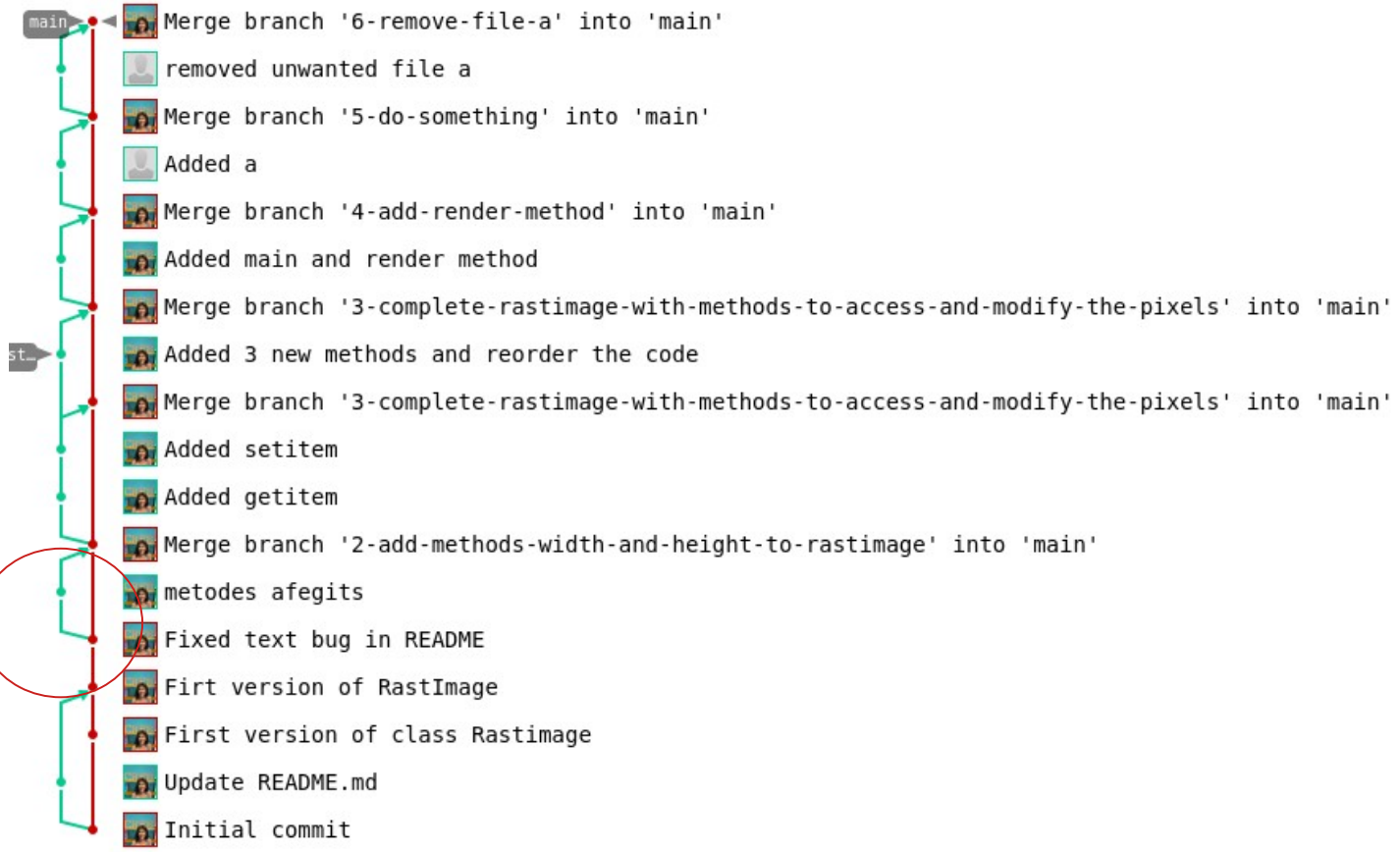
Git workflow

After having merged the branch, we checkout to the master/main and we pull the new version.

```
$ git status
En la branca 8-another-example
La vostra branca està al dia amb «origin/8-another-example».

$ git checkout main
S'ha canviat a la branca «main»
La vostra branca està 2 comissions per darrere de «origin/main», i pot avançar-se ràpidament.
  (useu «git pull» per a actualitzar la vostra branca local)

$ git pull
S'estan actualitzant ab7dc97..e1c1806
Fast-forward
 main.py      | 42 ------------------------------------------
 point.py     | 28 ++++++++++++++++++++++++++++
 rastimage.py | 62 ------------------------------------------------------------
 3 files changed, 28 insertions(+), 104 deletions(-)
 delete mode 100644 main.py
 create mode 100644 point.py
 delete mode 100755 rastimage.py
```

Example

.



The developer commits changes in her branch and merges with the master when ready.
Sometimes she keeps working in the branch and merges again after a while.

Git routine work procedure

- In a terminal, you **create** a project or **clone a repository**
- In the GIT web UI, you define **milestones** (.e.g. L1 and L2)
- In the GIT web UI, you create issues:
  - to solve bugs
  - to add new modules, classes, functionalities
- When you are ready to solve an issue. In the GIT web UI, you create a merge request that will automatically create a branch
- In the terminal, you **fetch** the new **branch**, **checkout** in it and work in it. Whenever you modify a file or create a new file, you **commit** the changes locally.
- In the terminal, you periodically push the changes to the cloud version. You can push unfinished versions, but avoid code giving syntaxis errors.
- When the issue is solved, in the GIT web UI, you ask to merge the branch with the master branch. I'll do it if it passeds the tests.
- In the terminal, you checkout to the master/main branch and **pull** the code.

Homework for next week

Practice with python and git: resolve your issue in the example project if you haven't done it yet.

During the week, you'll see that you have been assigned to a new project called image_viewer. As homeworks, clone it and analyze the class Rimage. We'll use it next week. You'll find its specification here.