

# Geometric & Graphics Programming Lab: Lecture 6

Alberto Paoluzzi

October 21, 2016

- 1 Workshop N.2
- 2 Jupyter notebook required
- 3 Minimal git/github instructions

# Workshop N.2

# Parametric (spatial) building frame in reinforced concrete

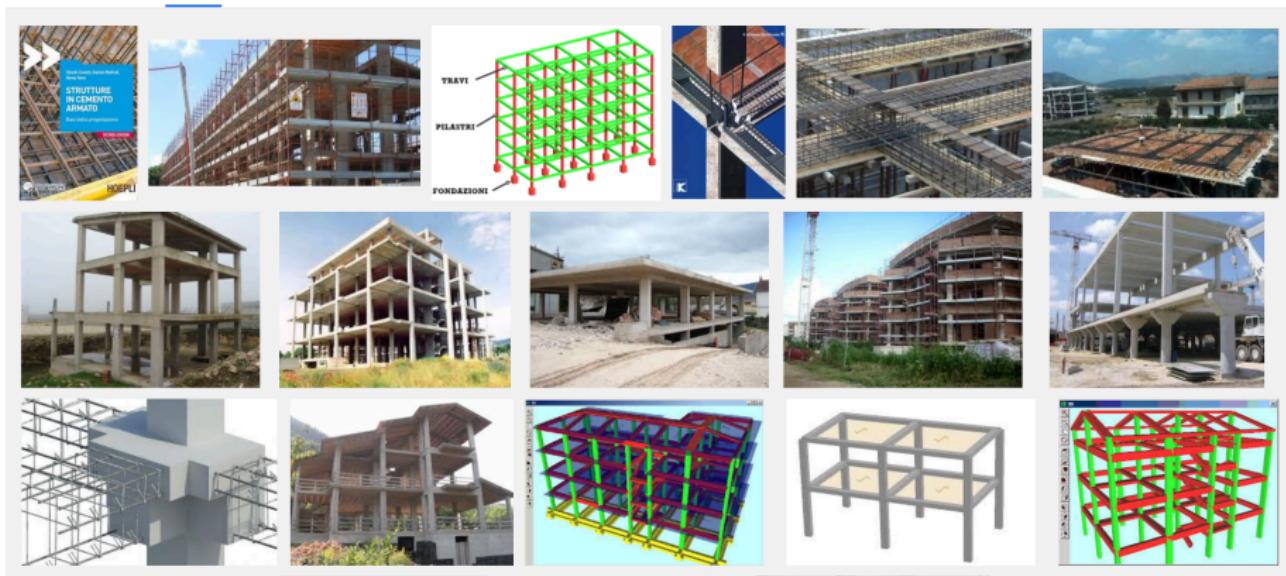


Figure 1:

# Look at some examples

- Strutture in cemento armato
- Reinforced concrete structures

!

# Requirements (1/2)

- Define a single Python function, named `ggpl_bone_structure`, with a single input parameter `file_name`.
- The function must return a **3D value** of **type HPC** (the geometric type of the `pyplasm` library), representing the bone structure of a (parametric) reinforced concrete building.
- The function must read a **comma separated value file** (suffix `.csv`), named `frame_data_xxxxxxx.csv`, where `xxxxxx` is the student's ID code.

# Requirements (2/2)

- The `frame_data_xxxxxxx.csv` file must include on **odd lines**:
  - the 3D vector positioning the local origin of the next frame with respect to the local origin of the previous one.
  - The first such vector sets the origin of the first frame w.r.t. the basis of the whole building.
- The `frame_data_xxxxxxx.csv` file must include on **even lines**:
  - the actual parameters of a planar concrete frame, according to the scheme of formal parameters required by the `ggpl` function developed in [workshop n.1](#) by the student itself.

# Hints to solution

- Once read the data file and assembled the **STRUCT** of planar frames, create the **connecting beams** using the **QUOTE** primitive
- Provide **only orthogonal connections**. Every pillar **must have** either 2 or 3 beams incident on it.
- Use **pyplasm** primitives **QUOTE** (to produce **1D cell complexes**) and **PROD** for **Cartesian product** of cell complexes (**HPC** values)

# Style specs (1/2)

- produce a **notebook** file, of type `.ipynb` (The ipynb file extension is associated with the [IPython notebook](#) and/or [Jupyter](#), a rich architecture for interactive computing written in Python and available for various platforms.)

# Style specs (1/2)

- produce a **single** Python function, with **ONE** parameter of **str** type
- output: a **single HPC** value
- use **meaningfull identifiers** (variables and parameters)
- use **camelCase** ids
- add **Python docstrings** (google for it)
- produce a **single** notebook file, named **workshop\_02.ipynb**
- file path: **your\_repo/2016-10-14/workshop\_02.py**

Jupyter notebook required

# Notebook tutorial

## Notebook Basics

# Minimal git/github instructions

# Minimal git/github instructions (1/2)

create your local repository

```
$ mkdir development
$ cd development
$ git clone https://github.com/your-account/gmpl
$ cd gmpl
$ mkdir 2016-10-14
$ cd 2016-10-14
$ touch workshop_01.py
```

# Minimal git/github instructions (2/2)

commit your work

```
$ git add -A .
$ git commit -m "add a short note to commit"
$ git push origin master
```