POLITECNICO
DI TORINO

Data Science Lab

Introduction to Python

DataBase and Data Mining Group

Andrea Pasini, Elena Baralis

Summary

- **Python engine**
  - Basic components and setup
- **Python language**
  - Data types, object oriented programming
- **Numpy library**
  - Computation with multi-dimensional arrays
- **Pandas library**
  - Tabular data and data preprocessing
- **Scikit-Learn library**
  - Machine learning and data science tools

Introduction to Python

- Python language
  - Clean and concise syntax
    - No semi-colons to end instructions
    - No braces to define if clauses and for loops
    - No need to specify variable types
    - …

Java

```
List<String> l = new LinkedList<>();
for (int i=0; i<10; i++) {
    l.add(i);
}
```

Python

```
l = []
for i in range(0,10):
    l.append(i)
```

Introduction to Python

- Python is an **interpreted** language
  - Code is not compiled to machine language
  - However the source code is compiled to an intermediate level, called **bytecode**
  - For this reason, to run Python programs, you need an **interpreter** that is able to execute the bytecode

Introduction to Python

PoliTo DBMG

- Sequence of operations executed by the interpreter

Python code (.py)

```
l = []
for i in range(0,10):
    l.append(i)
```

→ Syntax checker and translator

↓

Bytecode (.pyc)

↓

Python Virtual Machine (PVM)

Introduction to Python

- A common Python 3 setup on a **Linux** System
- Typically in the usr/bin folder:
  - "**python3**" executable: run Python programs
  - "**pip3**" executable: install Python packages
  - "**ipython3**" executable: run programs line by line
  - "**jupyter**" executable: run a jupyter notebook

Introduction to Python

- Executing a Python program

my_script.py

```
print("Hello")
```

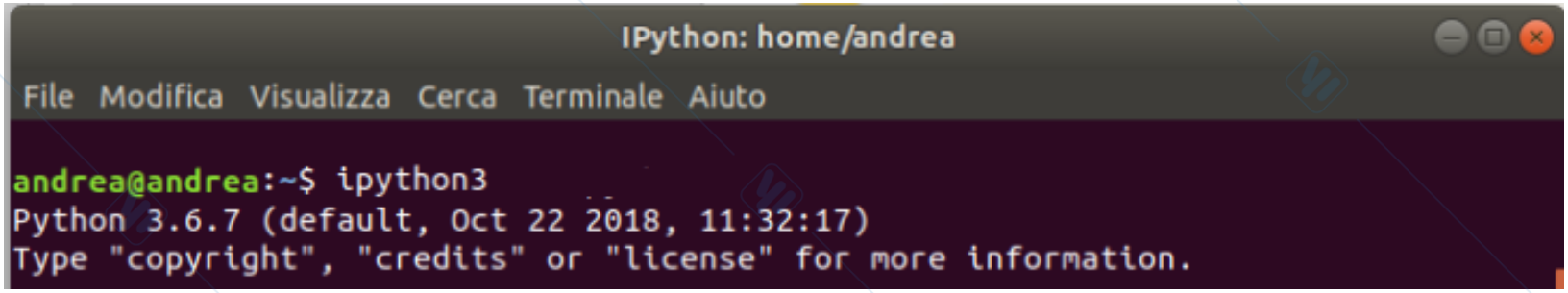
~/Documents/MyScript

- Type in your terminal:
  - cd ~/Documents/MyScript
  - python3 my_script.py

Introduction to Python

PoliTo DBMG

- Running Python line by line with iPython
- Type in your terminal:
  - ipython3 (or ipython, depending on your installation)



Introduction to Python

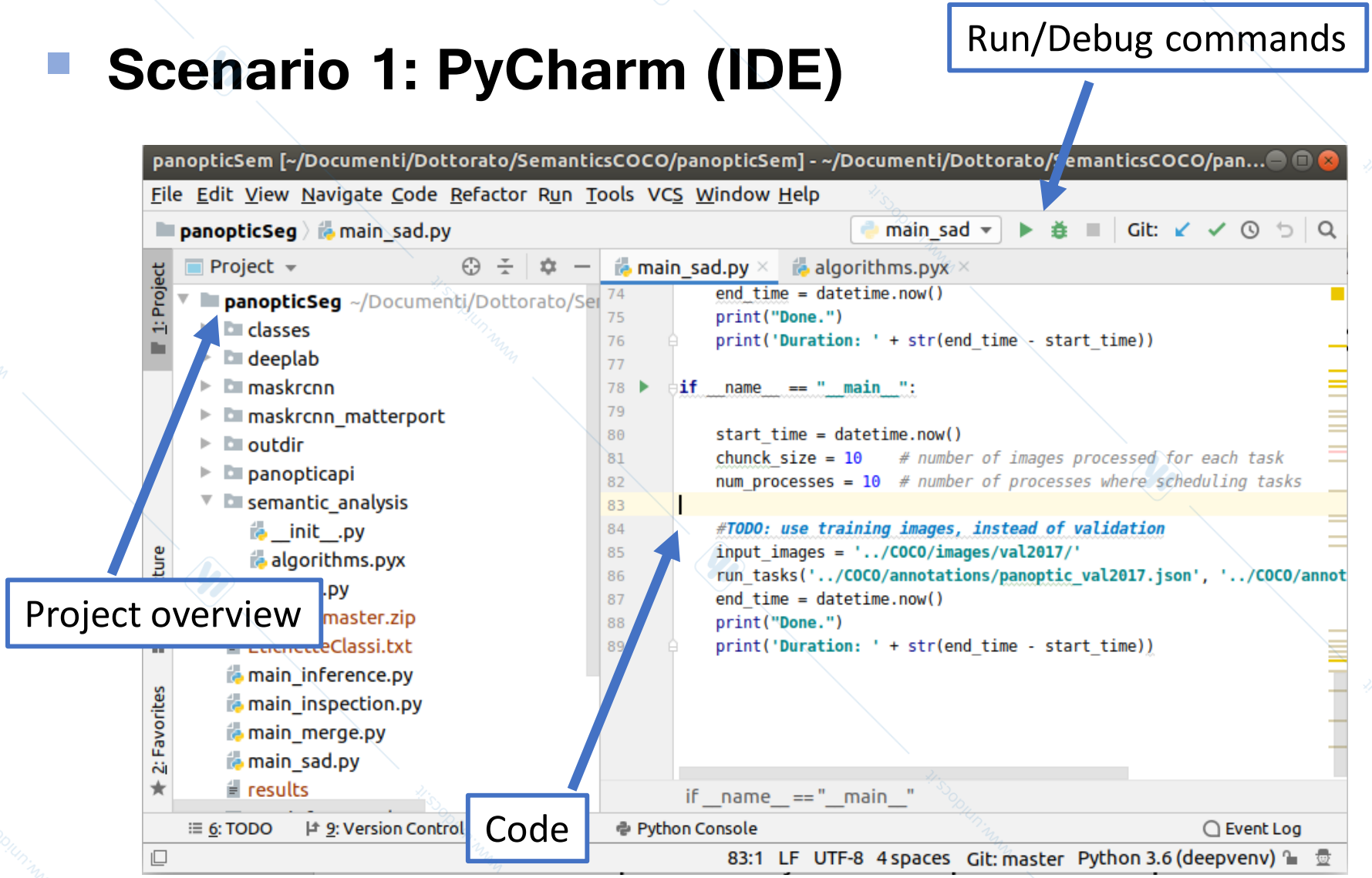
PoliTo DBMG

- Write your program line by line to see the results step by step…

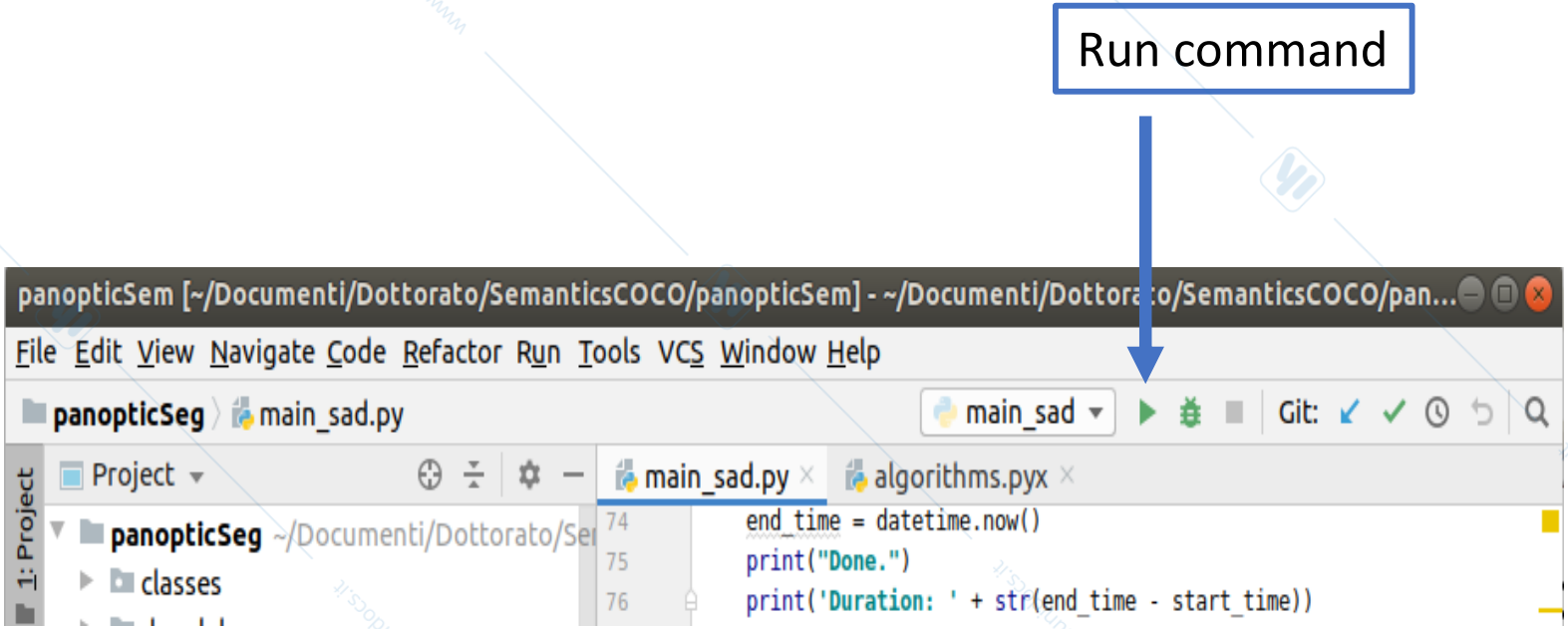
```
IPython: home/andrea
File  Modifica  Visualizza  Cerca  Terminale  Aiuto

andrea@andrea:~$ ipython3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
Type "copyright", "credits" or "license" for more information.

IPython 5.5.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: mystring = "hello"

In [2]: print(mystring)
hello

In [3]:
```

- **Python** and **iPython** programs are the core for executing scripts, but…
- There are two typical scenarios:
  1. Develop your Python **project** with an **IDE**
     - Example: Visual Studio Code, PyCharm
     - **Debug** and **run** your code inside the IDE
  2. Develop and test a Python **script** with **Jupyter notebook**
     - Inspect **step by step** the results
     - Keep the history of the output of the script

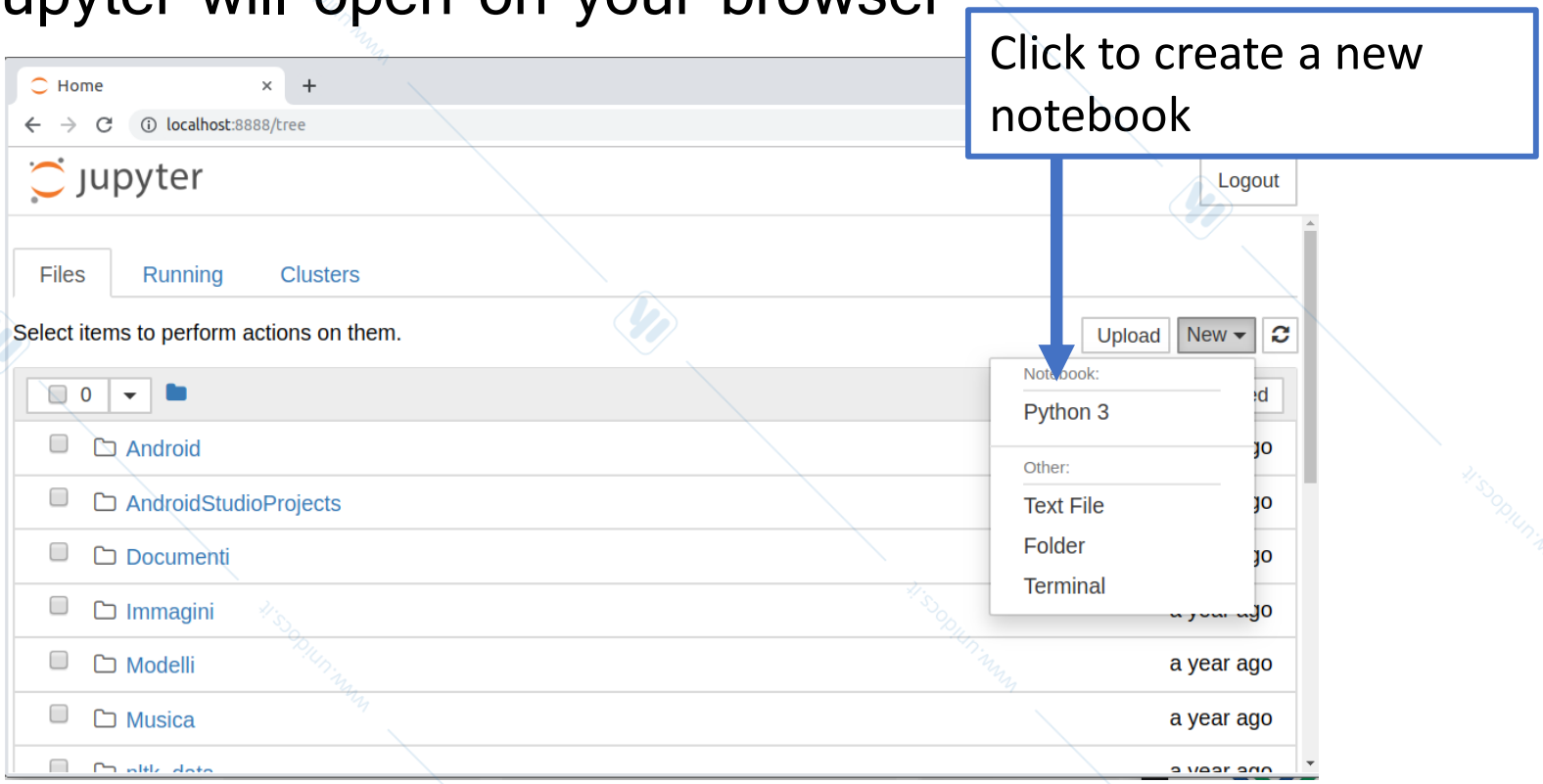
Scenario 1: PyCharm (IDE)



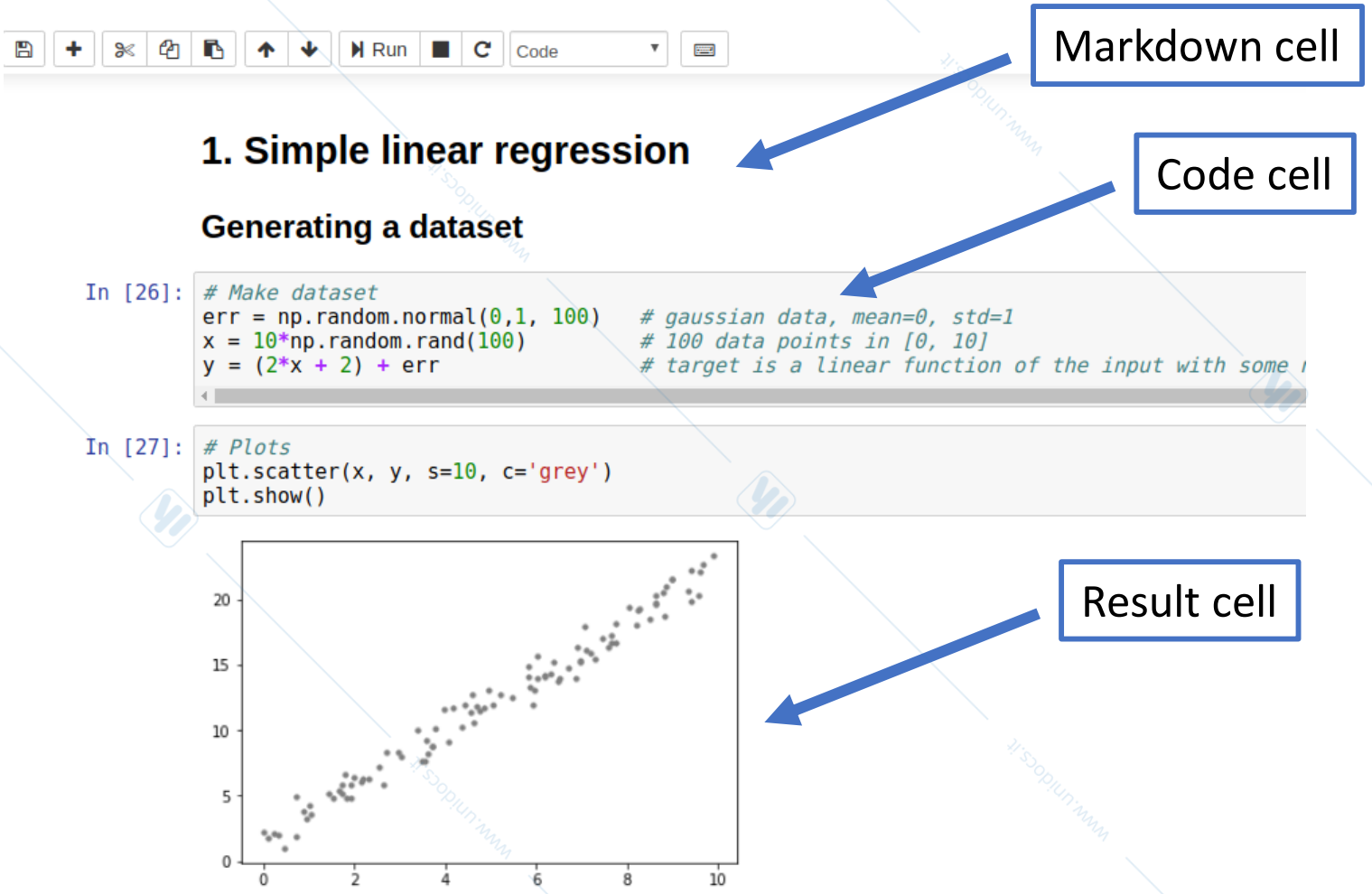
- **Scenario 1: PyCharm (IDE)**
  - When you click on the run button, the IDE will automatically call the "**python**" command to execute your script



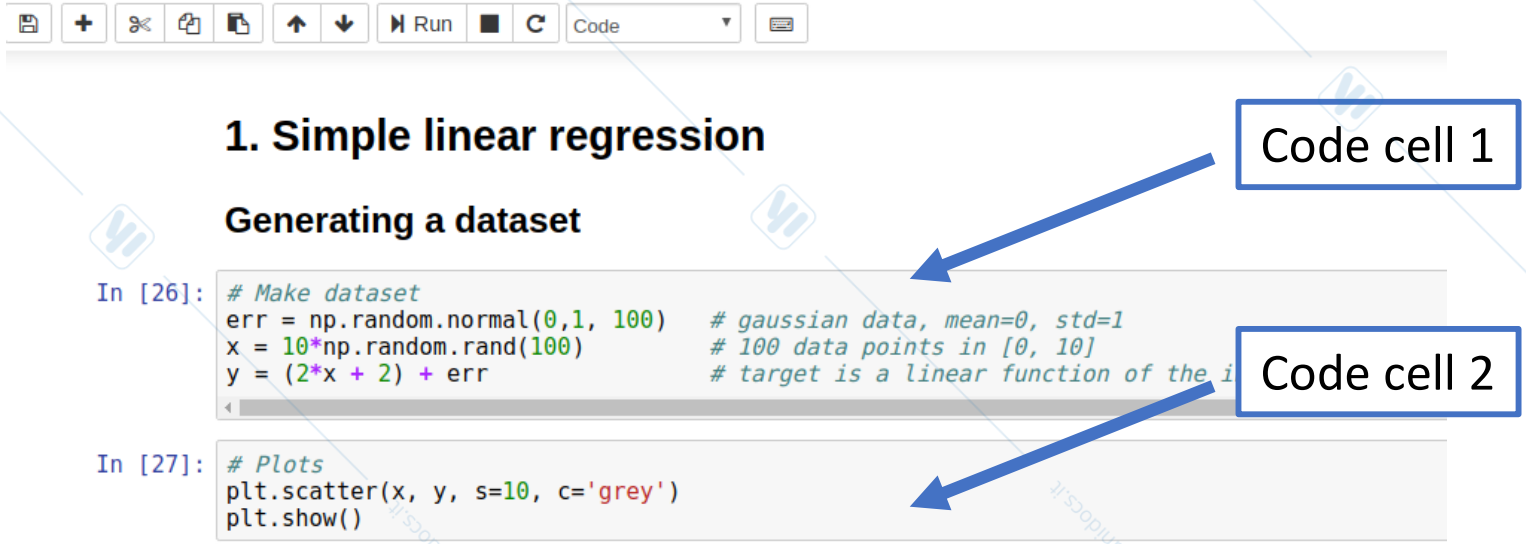
- **Scenario 2: Jupyter notebook**
  - Type in your terminal
    - jupyter notebook
  - Jupyter will open on your browser



Scenario 2: Jupyter notebook



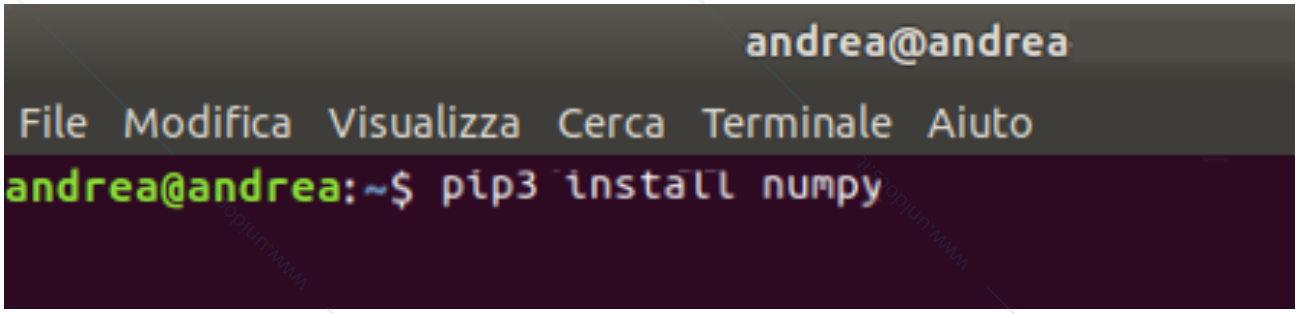
- **Scenario 2: Jupyter notebook**
  - Based on **iPython** command
  - Each code **cell** can be executed **separately** by pressing CTR + ENTER



IDE vs Jupyter notebook

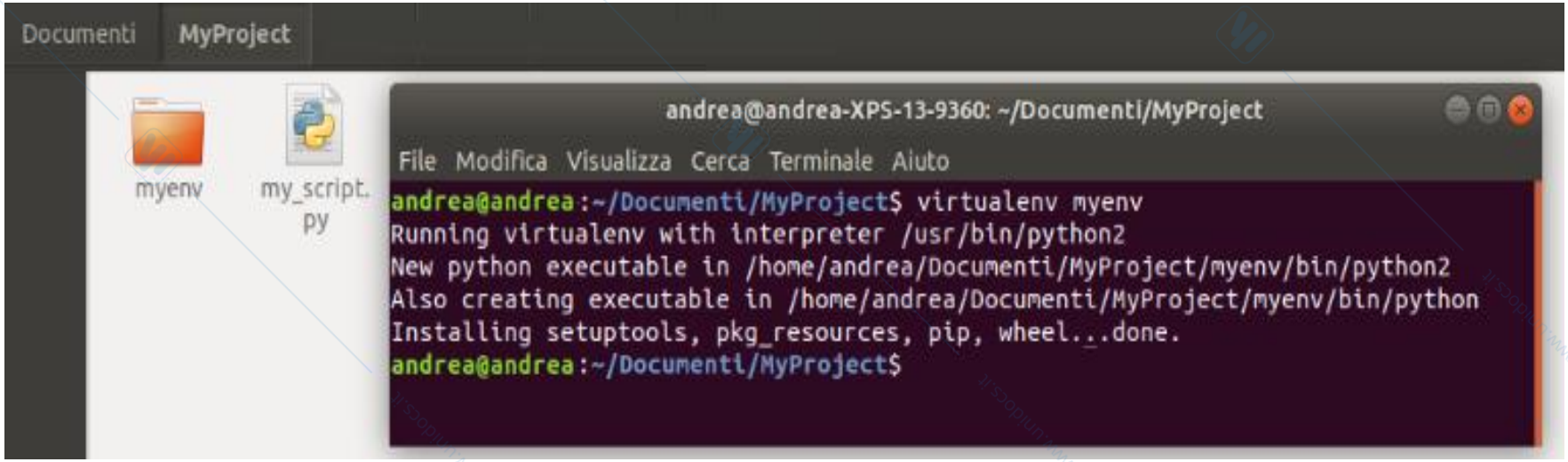
- **IDE**
  - For more **complex** projects (many files)
  - More powerful debug commands
  - More powerful code editing tools
- **Jupyter notebook**
  - For simple scripts and prototypes
  - Great **visualization** tool
    - Example: **report** with Python code and text for explanations

- **Installing libraries**
  - Python language is provided with many useful libraries:
    - Numpy, Pandas, Matplotlib, Scikit-learn, SciPy, …
  - To use any of them you first have to install it with the **pip** command:
    - pip3 install numpy
    - pip3 install pandas



- **Virtual environments**
  - The pip command will associate the libraries to your **default Python installation**
  - A more powerful way of managing libraries is to use a Python **environment (virtualenv)**
    - Designed when you want to design **different projects** that use different libraries and **configurations (e.g. versions)**
      - Each projects is associated to a virtual environment

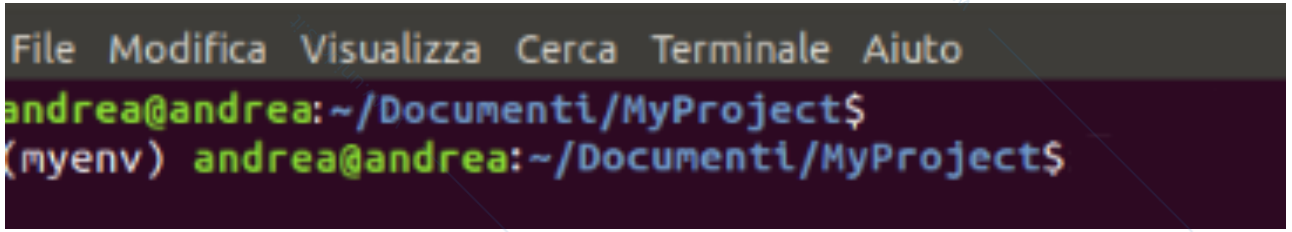
- **Virtual environments**
  - To create a new environment:
    - cd ~/Documents/My_project
    - virtualenv myenv
  - It will create a new environment in your project folder



- **Virtual environments**
  - To **activate** the created environment:
    - cd ~/Documents/My_project
    - source myenv/bin/activate

```
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
andrea@andrea:~/Documenti/MyProject$
(myenv) andrea@andrea:~/Documenti/MyProject$
```

- **Virtual environments**
  - After activation you can use the terminal to work within the environment



  - Install libraries to the current environment
    - pip3 install my_library
  - Execute a script/notebook within the environment
    - python3 my_script.py
    - jupyter notebook