Computer Programming
and
Database Systems

Master in Data Science and Business Analytics
yr. 2019/20

Instructor
Carlo Lucibello
carlo.lucibello@unibocconi.it

Teaching Assistant
Fabrizio Pittorino

Practical Info

- **All resources and communications on Piazza.**
  - Lessons' slides and exercises uploaded before each lesson. Solution to exercises will be provided in the form of Jupyter Notebooks.
  - Contact me If you don't receive an invitation within the next few days on your *studbocconi.it* account.
  - You can also link your personal email to your Piazza account for convenience
  - You can automatically forward your *studbocconi.it* incoming email to your personal account to make sure you don't miss important mail

- **Final grade** is 80% from **general exam** and 20% from **group assignment** (likely on managing databases and doing simple data analysis, using Spark or MongoDB).

- **Course content** roughly divided in (see the Syllabus):
  - **Python Programming**: from basic language constructs (just an overview) to advanced features and scientific libraries
  - **Database Systems**: Pandas, relational databases, SQL, Spark, NoSQL (MongoDB).

Programs and Programming Languages

A **computer program** is a set of instructions encoded in a binary (0/1s) format which can be “understood” by a machine.

The **execution** of a program boils down to fetching (binary) data from some address in memory, perform some arithmetic/boolean operation, store the result, perform the new operation, ….

Programming Languages allow Humans to create computer programs.
Compared to spoken languages, the grammar of computer languages may be less expressive, their domain more restricted, but they are always **unambiguous**.

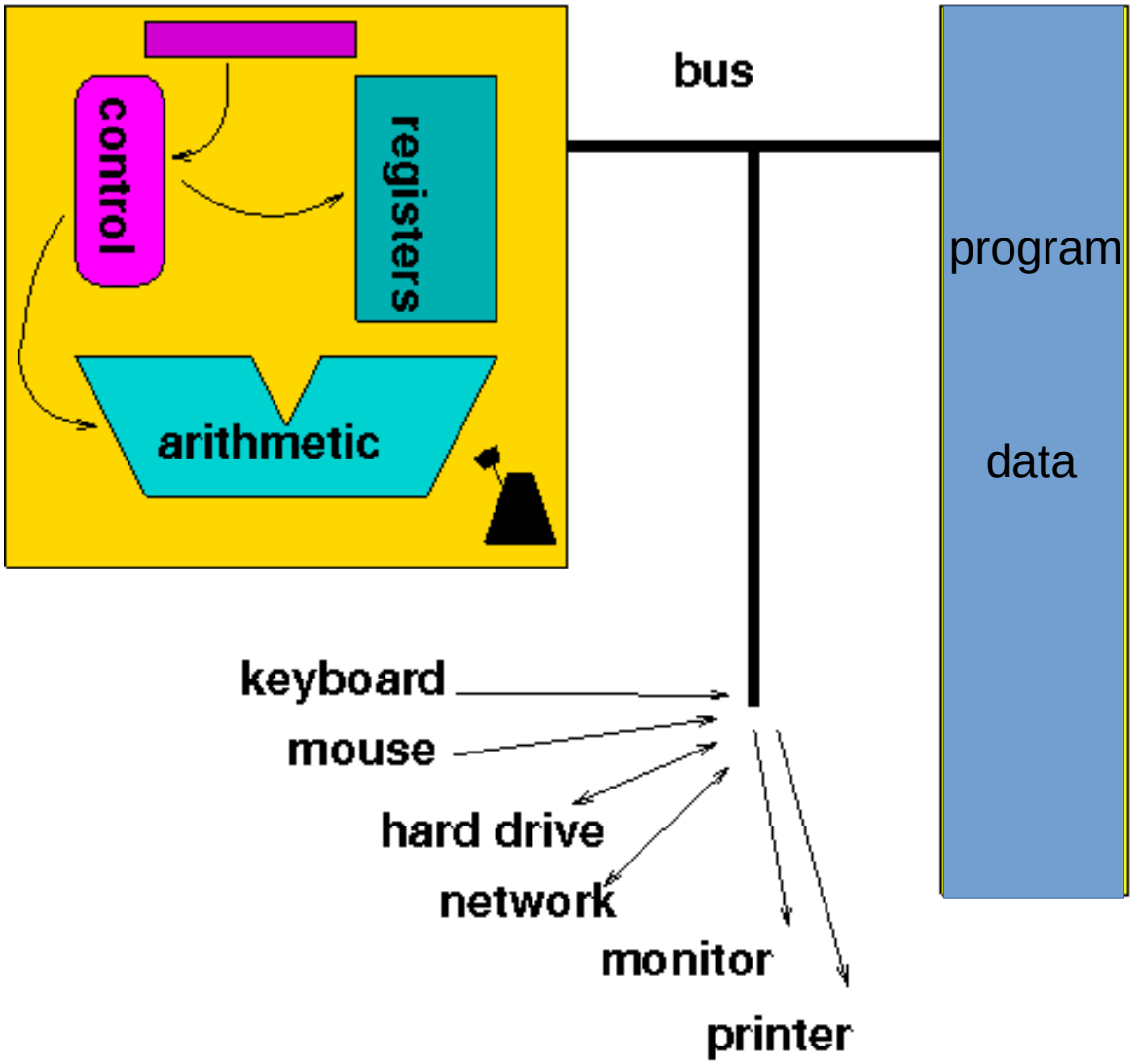
For **compiled languages** (C, C++, Fortran, …) the **source code** (one or more text files) is first compiled by a program called **compiler** to a set of machine instructions (the program) and then the program can be executed.

For **interpreted languages** (Python!) the situation is a bit different:
in a single step the source code, or statements coming from the shell, is sent to a program called **interpreter** which takes care of both the translation and the execution.

Computer Architectures

Central Processing Unity (CPU)

Random Access Memory (RAM)



The program and the data, usually dwelling on a hard drive, are copied to the RAM at computation time (faster access and comunication with the CPU).

For the inner working watch: https://www.youtube.com/watch?v=cNN_tTXABUA

Python

Python was created as an hobby project during Christmas holidays by Guido van Rossum and first released in 1991.

It has recently seen a huge rise in popularity as the predominant language in data science. R, Matlab and Julia are possible contenders.

Advantages:
- General purpose
- Easy to learn, easy to use
- Beautiful?
- Expressivity
- Interactivity
- Stability
- Easy to extend with C/C++
- Portability
- Huge community, libraries, ….

Disavantages:
- Slooooow, but:
  - Most performance critical parts in libraries are written in C
  - Some recent progress with decorators for just in time compilation (Numba, PyPy, …)
- Weak typing can lead to hard to predict and buggy behaviour

Setting up a programming environment

Unless you are an experienced programmer and you want to stick to your workflow, we recommend the following:

Download and install the **Anaconda** distribution.
After installation, you may want to update some packages (e.g. jupyterlab) in the Environments tab.
Anaconda is available for most Operating Systems (Linux / Mac OS/ Windows)

It comes with:

- A Python 3 interpreter (the “thing” which actually actually runs your code)
- IPython (a python shell on steroids)
- Jupyter Notebooks
- Basic data science libraries (numpy, scipy, matplotlib,…)
- The conda package manager
- JupyterLab (a integrated environment centered around notebooks, still a bit experimental but recommended for the course)
- Visual Studio Code (optional install). Is a lean but fully fledged IDE (Integrated Development Environment), recommended as your main non-python centric editor.

Jupyter / JupyterLab

Jupyter notebooks are useful for coding small project, for reports and for exploratory data analysis.

JupyterLab is the evolution of Jupyter notebooks. It aims to be an Integrated Development and Data Analysis Enviroment on the line of Matlab and RStudio.

Make sure you have JupyterLab >= 1.0.0 installed.

--- Live Demo: lesson1.ipynb ---

IDEs

Notebooks are cool for scripting and exploratory data science.

For larger projects you may want to consider factorizing your code into modules and using an Integrated Development Environment (IDE).

My advise is to use **Visual Studio Code**, which is distributed with Anaconda and supports most programming languages (I use it for Python, Julia and C++).

PyCharm is another good choice, altough it is python-specific.

If you feel very brave go for Vim.

If you are on the minimalistic side, any text editor (à la Notepad) and a Python interpreter will get the job done.

The Open Source movement

Most libraries you will learn to use during this course (numpy, scipy, matplolib, pandas…), tools (jupyter, visual studio code,…) and python itself are **Open Source** projects.

In the data science / machine learning world large part of the codebase is open source. For instance, most deep learning frameworks are open source and backed up by tech giants:

- TensorFlow (Google)
- PyTorch (Facebook)
- CNTK (Microsoft)

“In open source, we feel strongly that to really do something well, you have to get a lot of people involved.” Linus Torvalds

“Open platforms historically undergo a lot of scrutiny, but there are a lot of advantages to having an open source platform from a security standpoint.” Sundar Pichai (Google CEO)

Profit comes from services, knowledge and data.

You can follow, file issues, and eventually contribute to the development of many projects on **GitHub** (recently acquired by Microsoft).