Programming course

Lecture 1: Introduction

Diego de Sousa Rodrigues
diego.desousarodrigues@sciencespo.fr

SciencesPo Paris

Contact

- e-mail : diego.desousarodrigues@sciencespo.fr

Today's Agenda

- Logistics

Today's Agenda

- Logistics
- Why Economists Must Talk About Computing.

Today's Agenda

- Logistics
- Why Economists Must Talk About Computing.
- Computers : Who are they ?

Today's Agenda

- Logistics
- Why Economists Must Talk About Computing.
- Computers : Who are they ?
- Choose a programming language

Today's Agenda

- Logistics
- Why Economists Must Talk About Computing.
- Computers : Who are they ?
- Choose a programming language
- Roadmap

Logistics

- Meetings : Every Friday except **04.03** : 8am-10am.
- Communication : Slack
- Materials : all online to be shared through Slack or e-mail

Logistics

- Meetings : Every Friday except **04.03** : 8am-10am.
- Communication : Slack
- Materials : all online to be shared through Slack or e-mail
- Grades : it will be based on a final project to be done in team members

What and why ?

Why are you here ? ⇒ because programming is extremely useful in economics !

- dynamics models
- heterogeneous agent models
- machine learning (prediction, data collecting)
- econometrics

What and why ?

Why are you here ? ⇒ because programming is extremely useful in economics !

- dynamics models
- heterogeneous agent models
- machine learning (prediction, data collecting)
- econometrics

What are you going to learn ?

- Understand how does a computer think and work
- How to use the terminal ? How to use Git ?
- How to correctly write programs (in R and Julia)

How should you learn ?

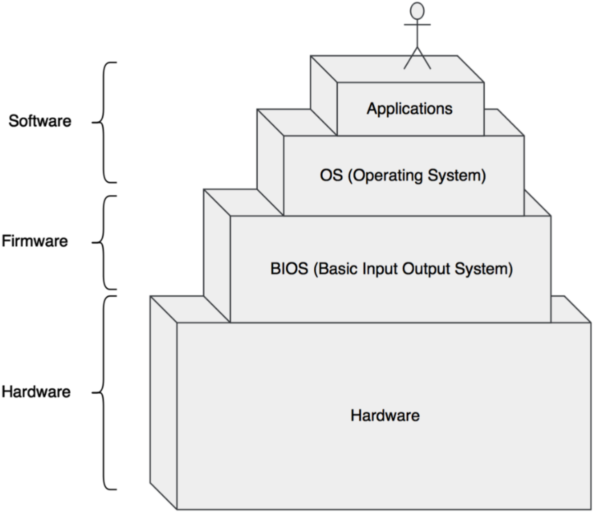
Doing programming is the only way to learn programming

How should you learn ?

Doing programming is the only way to learn programming

- We will try to make the class as interactive as possible with many exercices
- Weakly homeworks
- Final project by teams of 2-3 students

Computers : Who are they ?



The Software (I)

Interact with your computer

- The software : programs and other operating information used by a computer
- A program : a list of instructions given to the computer
- Operating system (OS) : a collection of programs in charge of the most basic tasks :

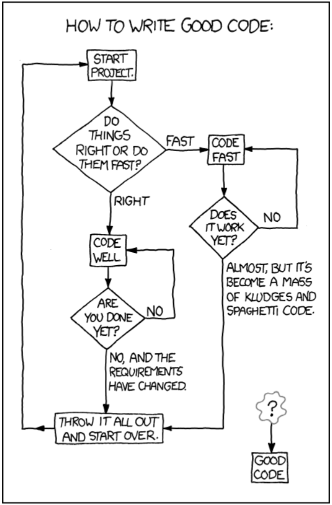
The Software (I)

Interact with your computer

- The software : programs and other operating information used by a computer
- A program : a list of instructions given to the computer
- Operating system (OS) : a collection of programs in charge of the most basic tasks :
  - provide an interface
  - manages the CPU
  - manages memory usage
  - provide security

The Software (II)

How to write a program ?


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL
ARE YOU DONE YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

The Software (II)

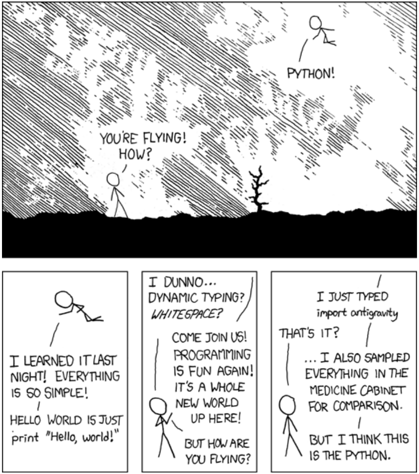
- Write re-usable code
- Write efficient code
- Write code with no bug
- Write nice and documented code

The Software (III)

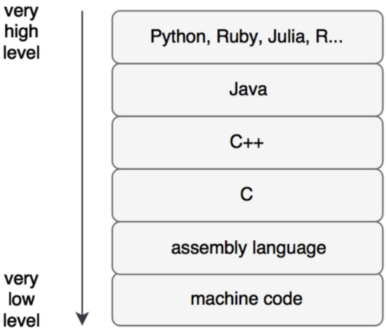
Best coding practices

1. Define CLEARLY what you want to do
2. Write a pseudo code i.e. a draft in mathematics or with words
3. Write AND comment the code
4. Test the code
5. Find the bugs (there always will be)
6. Re-start from 3.

Programming Languages


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Programming Languages (I)



Programming Languages (II)

High vs. Low

- Language : source code that will be transformed into binary code that can be interpreted by the CPU
- Translation is operated by a compiler

Programming Languages (II)

High vs. Low

- Language : source code that will be transformed into binary code that can be interpreted by the CPU
- Translation is operated by a compiler
- High vs. Low level : how far are you from the computer language (the machine code) $\approx$ How much translation do you need ?

Programming Languages (II)

Compiled vs. Interpreted

- Compiled languages (Fortran, C, C++) : the source code is compiled into machine code ex ante (by yourself !)
- Interpreted languages (Python, R, Matlab) : the interpreter executes the instruction directly by translating the code into routines that are already compiled in machine code

Programming Languages (II)

Compiled vs. Interpreted

- Compiled languages (Fortran, C, C++) : the source code is compiled into machine code ex ante (by yourself !)
- Interpreted languages (Python, R, Matlab) : the interpreter executes the instruction directly by translating the code into routines that are already compiled in machine code
- Julia (Compiled Just In Time) : speed of compiled + easiness of interpreted ; But you'll have to wait for the the end of the course !

Programming Languages (III)

Open source vs. proprietary

- Open-source (C++, R, Python) : free to use, huge community, everybody contribute to the shared knowledge and can develop libraries (+)
- Proprietary (Stata, Matlab) : you must pay and use the entire software (language + interface), developers from the company are paid to sustain the source code

Programming Languages

Less is more ?

- Java : `protected int i; int i = 1;`

Programming Languages

Less is more ?

- Java : `protected int i; int i = 1;`
- R : `i = 1`

Programming Languages

Less is more ?

- Java : `protected int i; int i = 1;`
- R : `i = 1`
- Julia (and Python) : `i = 1`

... but Julia can be 250x faster than R !

Programming Languages

Aruoba and Fernandez-Villaverde (2015)

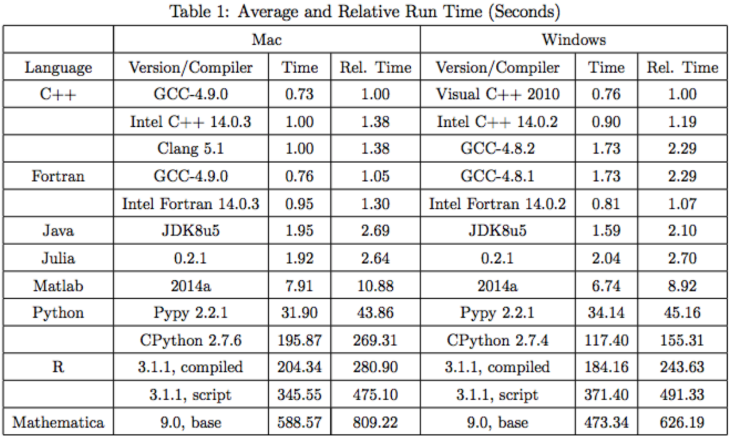


Table 1: Average and Relative Run Time (Seconds)

	Mac			Windows		
Language	Version/Compiler	Time	Rel. Time	Version/Compiler	Time	Rel. Time
C++	GCC-4.9.0	0.73	1.00	Visual C++ 2010	0.76	1.00
	Intel C++ 14.0.3	1.00	1.38	Intel C++ 14.0.2	0.90	1.19
	Clang 5.1	1.00	1.38	GCC-4.8.2	1.73	2.29
Fortran	GCC-4.9.0	0.76	1.05	GCC-4.8.1	1.73	2.29
	Intel Fortran 14.0.3	0.95	1.30	Intel Fortran 14.0.2	0.81	1.07
Java	JDK8u5	1.95	2.69	JDK8u5	1.59	2.10
Julia	0.2.1	1.92	2.64	0.2.1	2.04	2.70
Matlab	2014a	7.91	10.88	2014a	6.74	8.92
Python	Pypy 2.2.1	31.90	43.86	Pypy 2.2.1	34.14	45.16
	CPython 2.7.6	195.87	269.31	CPython 2.7.4	117.40	155.31
R	3.1.1, compiled	204.34	280.90	3.1.1, compiled	184.16	243.63
	3.1.1, script	345.55	475.10	3.1.1, script	371.40	491.33
Mathematica	9.0, base	588.57	809.22	9.0, base	473.34	626.19

Programming Languages

This course

- We will learn R : open-source, interpreted and user-friendly !
- Moreover, R is fantastic for people working with data.

Your first code (I)

Example : the Euler equation

- The Euler equation :

$$u'(C) = \beta \cdot \mathbb{E}_t[R_{t+1} u'(C)]$$

- Find $\hat{C}$ s.t. $F(\hat{C}) = 0$ where :

$$F(\hat{C}) = u'(\hat{C}) - \beta \cdot \mathbb{E}_t[R_{t+1} u'(\hat{C})]$$

Your first code (II)

Example : the Euler equation

- Find $\hat{C}$ s.t. $F(\hat{C}) = 0$ where :

$$F(\hat{C}) = u'(\hat{C}) - \beta \cdot \mathbb{E}_t[R_{t+1} u'(\hat{C})]$$

- Transpose $F(\hat{C})$ into a code line :

```
F(C) = u′(C) - beta * R * u′(C)
```

- and then, find the $\hat{C}$ s.t $\mathrm{F}(\hat{C}) = 0$

```
find-root(F)
```

Your first code (II)

Example : the Euler equation

- Finally, comment your code :

```
F(C) = u′(C) - beta * R * u′(C) # the Euler equation
find-root(F) # solve for the consumption level s.t. F(C) = 0
```

Roadmap

- (Part #1) :
  - UNIX-shell or *How to directly interact with your computer through the terminal ?*
  - Git and GitHub *How to write programs in team and deal with version control ?*

Roadmap

- (Part #1) :
  - UNIX-shell or *How to directly interact with your computer through the terminal ?*
  - Git and GitHub *How to write programs in team and deal with version control ?*
- Part #2 :
  - Programming in R, i.e. learning basics of programming by doing programming
  - starting with Julia !

Roadmap

- (Part #1) :
  - UNIX-shell or *How to directly interact with your computer through the terminal ?*
  - Git and GitHub *How to write programs in team and deal with version control ?*
- Part #2 :
  - Programming in R, i.e. learning basics of programming by doing programming
  - starting with Julia !
- Needed for next lecture : computer with a terminal