Introduction to the CS101 course at MEC

Folders

See these subfolders online for more documents:

- YouTube videos for the first part of the CS101 lectures (by Professor Sanjay Dhande, founder director of Mahindra Ecole Centrale).
- Exams: exam papers and complete solutions,
- demos: about 60 interesting use of Python for various scientific applications,
- hackathon: assignments and solutions for the two hackathons I organized at MEC in Feb. and March 2015,
- feedback: evaluation of the course, by the students (after the course),
- labs: assignments and complete solutions for the practical sessions (labs),
- projects: for the projects (see mec-cs101-integrals or mec-cs101-matrices for detailed solutions of my two projects).
- slides: complete slides (PDF or PowerPoint) for all the CS101 lectures.
- solutions: solutions to various website or books about learning Python.

License

This is a really quick introduction to computer programming, CS and Python for the CS101 course.

- Any feedback, question, remark or improvement is of course welcome!
- This document is also (publicly) available on-line at `http://perso.crans.org/besson/cs101`,
- as a PDF at `http://perso.crans.org/besson/cs101/index.pdf`,
- or as a simpler web-page at `http://perso.crans.org/besson/cs101/index.htm`.

---

1. Reading this document will give you a *short* overview of what you will learn at Mahindra École Centrale (MEC) during the CS101 course and why you will learn it. This course will be given in the second semester of **2014-2015** (from January 2015 to May 2015).
2. A lot of resources are available (websites, videos, on-line courses, problems, textbooks etc). You should try to spend some time reading, viewing or working on these, *by your own*: **before** the beginning of the course but also *during* the course.
3. During the December 2014 break, you will have **4 weeks without any class**, and working on that CS101 course is *an excellent way for you to prepare the second semester*, to *stay focus on learning*, and maybe to have some fun!

---

1) How to (quickly) start to learn computer programming (with Python)

1. Read carefully **this document**, from here to the last page. Using a soft copy (from a computer or a smartphone) is better, as this document contains lots of hyperlinks.
2. From `continuum.io/downloads`, **download** Anaconda and install[1] the latest version of **Python** on your own computer (if you have one, it is probably running on Mac OS X, Windows or Linux). You can also try one of the on-line Python interpreters (no need to install anything).
3. Go **read the main reference websites** and documents (main links are given below).
4. Start to get used to the user interface of the EDI we will used for the course: Spyder.
5. Practice with the first examples from `introtopython.org/hello_world.html`.
6. (Optional) If needed, read these FAQ: general Python FAQ, programming FAQ, or Python on Windows FAQ, or ask us any question by email at `CS101@crans.org`.
7. **Keep practicing** by going through the `IntroToPython.org` mini-lessons, watching `PythonLearn.com` videos, or reading `Python.org` tutorials. The most important thing is to **write code by yourself**!

Note[1]:

- Python version 2.7 (with all the required packages and some good additional softwares) can easily be downloaded and installed in two clicks with the **Anaconda** installer, from `continuum.io/downloads` (chose the **2.x** installer).
- If needed, one of these could help: `IntroToPython.org/programming_environment.html`, or `Developers.Google.com/edu/python/set-up`, or `PythonLearn.com/install.php`.
- (Optional) You can also download Python by itself at `Python.org/downloads`, or use a heavy but more complete installer. If Anaconda fails, please try PythonXY (at `code.google.com/p/pythonxy`) or Canopy (at `store.enthought.com/downloads`) or WinPython (at `WinPython.sourceforge.net`). They are all free, available for Windows, Mac OS X and Linux.

2) Quick explanation of what is computer programming

Computer programming (often shortened to *programming*) is a process that leads from an original formulation of a computing problem to executable programs.

It involves activities such as analysis, understanding, thinking, and generically solving such problems resulting in an algorithm, verification of requirements of the algorithm including its correctness and its resource consumption, implementation (commonly referred to as coding) of the algorithm in a target programming language.

Computer programming and software engineering are nowadays one of the most popular activities for newly trained engineers, and lots of you will probably do your first internships in an IT company!

2.1) Main properties a computer program should satisfy

Whatever the approach to development may be, the final program must satisfy some fundamental properties. The following properties are among the most relevant[1]:

1. Usability: the ergonomics of a program, *ie* the ease with which a person can use the program for its intended purpose or in some cases even unanticipated purposes.
2. Reliability: how often the results of a program are correct.

[1] This short explanation comes mainly from Wikipédia. The linked articles give more details!

3. Readability: to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.
4. Robustness: how well a program anticipates problems due to errors (not bugs).
5. Efficiency or performance: the amount of system resources a program consumes (processor time, memory space, slow devices such as disks, network bandwidth etc): *the less, the better.*
6. Portability: the range of computer hardware and operating system platforms on which the source code of a program can be compiled/interpreted and run.
7. Maintainability: the ease with which a program can be modified by its developers in order to make improvements or customizations, fix bugs and security holes, or adapt it to new environments.

2.2) Why choosing Python?

Because it is free, excellent on the 6 firsts of these 7 key aspects, and because of how easy it is to learn and use it in comparison to the other popular languages (such as C or Java), **we at MEC have decided to use Python for the CS 101 and 202 courses** (the two first CS courses in the common syllabus of our 4 branches). Other languages will be used in the future for some other courses.

First example That small 2-line program below will be Python thanking us for choosing it, by producing the output: `« I am so happy that you chose me! » -- Python 2.7.6` !

```
# Your first Python program (a comment starts with # and is not be executed)
from sys import version      # We import one constant value from the sys module
# Python is thanking us for choosing it, we print a message with the function print
print("« I am so happy that you chose me! » -- Python" + version[0:5])
```

3) Quick explanation[2] of *what is Python*

Python is an *easy-to-learn*, *powerful* programming language, and *one of the most popular in the world* from the last 15 years. Python is a *widely used general-purpose*, *high-level* programming language.

It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for beginners, but also for scripting and rapid application development in many areas and on most platforms. The Python interpreter and the extensive standard library are *freely* available in source or binary form for all major platforms from the Python website, `Python.org`. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as `C++` or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library, the result of years of hard work by a huge community scattered around the world.

[2] This short explanation comes mainly from the Python.org Tutorial and the Wikipédia page about Python. Feel free to follow your curiosity and go read more!

4) Some motivation for learning programming with Python

- During your +11 or +12, or during your former education, you may have learned[3] or started to learn C, C++, or Java. If you did, you probably remember how *slow* and *painful* is the usual write/compile/test/re-compile cycle.

- But having no previous experience[4] in programming is not an issue for this course, because we will start from the really basics.
- For example, if you do much work on computers, eventually you find that there is some task you would like to automate. You may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you would like to write a small custom database, or a specialized GUI application, or a simple game.

In any case, **Python** is just the language for you. This page (`docs.Python.org/2/tutorial/appetite`) gives a more complete overview of the many reasons to learn programming, and especially to chose Python!

4.1) Python 2 or 3?

We chose Python 2 (version 2.7) rather than Python 3.4, (for many reasons). One lecture at the end of the course might be used to highlight the main differences between the two versions, but no worry, they are **very similar**! We will also quickly explain how to easily port a Python 2 code to Python 3.

[3] Programmers should read `wiki.Python.org/moin/BeginnersGuide/Programmers`.
[4] True beginners should read `wiki.Python.org/moin/BeginnersGuide/NonProgrammers`.

5) On-line main references

These websites are the **most useful resources for learning Python**, and we will use them a lot.

- `IntroToPython.org` is a series of excellent on-line lectures, and it will be *our first reference* for teaching you the Python part of CS101. Try to start reading it right away! It comes with many examples, and dozens of exercises which will be used for the programming labs.
- `PythonLearn.com` will be the other main reference (chapters 0 to 11). That course comes as Youtube videos (watch the first one?), complete slides, a notebook, and lots of examples.
- `Python.org` is the official website of reference for the Python language.

---

6) Some details about CS101

This **CS101 course** aims to teach all of our MEC students about the basics of programming computers and algorithmic, using Python 2.

6.1) First contact with programming!

This course is specifically designed to be your first programming course using the popular Python programming language. For a lot of you, this will be your very first contact to computer programming.

The outline of the course is designed to start from the fundamental bases and to go slowly to an intermediate level in algorithmic, problem solving using a computer program, and Python. We will use simple examples, some data analysis and some questions from chemistry, physics, maths and engineering courses as programming exercises through the course. And you will have lots of time during the CS labs to practice programming and to think on your own (but always with the help of one or more tutors and lab assistants).

Understanding *how to process data* is valuable for everyone regardless of your career. This course might kindle an interest in more advanced programming courses or courses in web design and development (this will be covered for the CSE branch students in 3$^{\text{rd}}$ and 4$^{\text{th}}$ years), or just provide some basic skills to help you when you are faced with a bunch of data that you need to analyze.

You can do the first programming assignments for the class using your personal computer or smartphone (or even a web-browser). **All required softwares** and on-line resources for the course **are free** (and open-source). It is **highly recommended to install Python** on your machine. Regardless if you use a Mac, Windows or UNIX/Linux computer, Python is available for you!

6.2) « *And what if I do not have a laptop neither a smart-phone?* »

Well, the programming lab sessions will happen in the computer lab, which is open for you *a lot*. For viewing the on-line lectures, practicing for assignments and projects and working on your own, it is recommended to have your own laptop, but you can manage without it. We will do our best to help you in case you do not have your own machine. Purchasing a laptop will help but it is not mandatory.

6.3) Thinking like a *computer scientist*?

The goal of this course is also to teach you to *think like a computer scientist*. This way of thinking combines some of the best features of mathematics, engineering, and natural science.

- As *mathematicians*, computer scientists use formal languages to denote ideas and computations,
- As *engineers*, they design things, assembling components into systems and evaluating trade-offs among alternatives,

- As *scientists*, they observe the behavior of complex systems, form hypotheses, and test predictions.

The single most important skill for a computer scientist is **problem solving**, which means the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately. As it turns out, the process of learning to program is an excellent opportunity to practice problem solving skills. On one level, you will be learning to program, a useful skill *by itself*. On another level, you will use programming as a means to an end. As we go along, that end will become clearer, especially with the programming project you will have to do in March and April 2015.

6.4) Technical details

This small paragraph gives some logistic details, for you to know how this CS101 course will be conducted. Some of these details can still change.

Team for CS101

The CS101 course will be half time of lectures, and half time of labs, and many professors will be there for both lecturing and conducting the labs:

1. **Four professors for lectures and labs:** Founder Director of MEC Prof. Sanjay G. Dhande will give the first lectures, and then Prof. Aryakumar Bhattacharya, Lilian Besson and Dr. Vipin Kizheppatt will take care of the lectures and the labs.
2. **Two professors for labs:** Profs. Prof. P Kondaiah, Vivek Bajpai (and one new faculty) will help to conduct labs, and be assisting you for the programming projects on a regular basis.
3. **Four labs assistants:** Aakash Deep, Nagendar, Sai Srinivas and Ashish Roy will help us to keep the two computer labs in order, and to help conducting labs and projects.

Other aspects

1. **Number of hours** (by week): this course will use 2 hours of lectures and 2 hours of computer lab, all being mandatory. Attendance will be taken, and too many absences will be punished.
2. **Rooms:** lectures will be in one of the lecture theater (LT 2 and 4), and labs will happen in the computer lab room or the drawing (AutoCad) lab.
3. **Evaluation:** 50% will be for the written part on concept of programming and algorithmic, and 50% will be on programming. For the conceptual half: 10% for the two written Mid-Term Exams (mid of February, end of March), 30% for the Final Exam (in May). For the programming other half: 10% will be the first (small) project to do alone, 15% on all the labs (from January to April on various aspects), and 20% on the second project to do in small groups. (*This could change.*)
4. **Moodle** will mainly be used by us to upload some lectures notes or slides and the lab exercises and assignments. *You will have to use it* to give us back algorithmic homeworks and programming projects, on a regular basis.
5. **Language:** English, as always, will be the only speaking and writing language used in this course. (if you are curious, you can try to write a Python program in Hindi or in Telugu at `wiki.Python.org/moin/TeluguLanguage`, but that's pretty much it)

7) Textbooks of reference

7.1) At the MEC library or stationary shop?

These two books are already available at the MEC library, and will be available at the MEC stationary shop. If you are impatient and want to order them by yourself, feel free to do it, either by the stationary shop or via `flipkart.com`!

- The small green book called « *Python in easy steps* » (by Mike McGrath) gives a good introduction to the basic concepts of programming (with Python *v2.7*). 5 copies are already there, and 30 will be available. Buying it is recommended (at the stationary shop for Rs. 225 or via flipkart.com).
- The world-famous *Bible* of algorithms and data structures, « *Introduction to algorithms* » by Cormen, Leiserson, Rivest, and Stein, has already *30* copies waiting for you at the library! This book will be extremely useful for *all the CS courses*, especially CS 101 and 202. For every CSE student, we recommend you to purchase one (India-specific) version: the last (3rd) edition will be available at the stationary shop for Rs. 995, (or via flipkart.com Rs. at 784, or here). Solutions to almost all problems are available on-line.

7.2) More more on-line documentation?

These books are **freely** available, on-line (as soft copy). It is possible to download them *once* in order to read them from your (off-line) laptop or smart-phone *anytime you want*. All these resources will be available off-line on every computer of the CS lab, in order to allow you to consult them whenever you want.

- « *Python for Informatics: Exploring Information* », the `PythonLearn.com/book.php` notebook (only chapter 0 to 11).
- The « How to » guides on the Python website are in-depth documents on *advanced* topics.
- And there is also the Python official documentation on `docs.Python.org` (and this Android app or these iOS apps, to read the Python documentation off-line on your smart-phone). This documentation can also be downloaded in order to use it off-line.

---

8) Others resources

This last part gives a **lot more resources** and links to help you learn algorithmic, programming and Python.

8.1) More on-line courses or references

- Some extra tutorials or lectures: `ZetCode.com/lang/python`, or `GitHub.com/yoavram/CS1001.py`, or `www.LearnPython.org`, or `Sthurlow.com/python`.
- « *How to Think like a Computer Scientist: Interactive Edition* » is the interactive version of the now famous eponymous book (also available as a PDF).
- « *Problem Solving with Algorithms and Data Structures using Python* », a good and complete on-line course on algorithmic, data structures and problem solving with Python.
- Two courses on `www.Coursera.org`: InteractivePython is designed to be a fun introduction to the basics of programming in Python, and PythonLearn is the Coursera version of `PythonLearn.com`.

8.2) More books on Algorithms and Data Structures?

Python and concrete programming will be the main part of the CS101 course, but **you will also learn about algorithms and data structures**, and it will be an important part of the the course and main component of evaluations. These books are already available at the MEC library, and will be available at the MEC stationary shop.

- The small but good *« Algorithms + Data Structures = Programs »*, by N. Wirth, costs Rs. 250,
- The famous *« Data Structures and Algorithms »*, by Aho, Hopcroft and Ullman, costs Rs. 750.

8.3) Which software will we use to write our Python programs?

As far as now, we decided to use the Spyder EDI[5] for this course.

In CS 101 labs, we will be able to help you if you use the same environment as everyone, of course it will be harder if you choose something else. We will try our best, but for both you and us, it will really be simpler if you also use Spyder. Spyder will be used (on Windows) as the editor and software to write, execute, analyze and debug your Python programs.

Spyder has many advantages: it is free and open-source, easy to install on many platforms (Windows, Mac OS X and Linux), and it is easy to begin with. But it is also powerful enough to do everything we need for CS101, including for real projects as you will have to deal with for the second group project.

(Optional) An other IDE?

But if you want to try another *Integrated Development Environment* (IDE), **on your own**:

- You can choose among the most popular ones, there is Komodo Edit, PyCharm, Wing IDE 101, or Geany on Linux.
- We will use the excellent and multi-platform **IPython** console (ie. command line environment), which is included in Anaconda. And DreamPie on Windows or bPython on Linux are good also.
- ... But you can also prefer to use a powerful (but more generic) **text editor**: the most popular are emacs and vim (both multi-platforms), Notepad++ for Windows and TextMate for Mac OS X. My personal favorite has been Sublime Text 3 since a year or so (with Anaconda plug-in). (Learn why a text editor can be more powerful than a word processor? on Wikipédia)

8.4) On-line *interpreters*

These free web-services brings all the simplicity, conciseness and greatness of Python right into your web browser! You can use any of these to quickly test a piece of code, or practice programming with your laptop or smart-phone from anywhere, without having to install Python!

1. `REPL.it/languages/Python` is visually the best one, and comes with some basic examples.
2. `Trinket.io/python` aims to « *put interactive Python anywhere on the web* ».
3. `PythonTutor.com` is a *visually helpful debugger for Python*. You write some simple code on the left, and then the website is showing you, *step by step*, what is happening when the code is executing. It will help you a lot at the beginning.
4. `Live.SymPy.org` is a *math-oriented* on-line interpreter, which use the powerful SymPy project, a Python library for symbolic mathematics[6], a full-featured computer algebra system (CAS).
5. `Skulpt.org` and `CodeSkulptor.org` are simple but good on-line interpreters, with lot of examples.
6. Finally, the official website provides `console.Python.org`.

[5] Read that page `https://pythonhosted.org/spyder/installation.html` to know how to install in on your laptop. The recommended way is to use **Anaconda**.

[6] By the way, you can also use `www.SymPyGamma.com` as a powerful *calculator* (for maths, physics or anything)!

8.5) Python on your smart-phone!

It is also possible to install an application on any (recent enough) smart-phone, allowing you to use Python on your device even off-line!

- These 2 free Android applications bring an almost complete Python environment to your Android smart-phone: QPython or PyCo.
- These *non-free* iOS applications do the same for your iOS smart-phone: `www.PythonForiOS.com`, or one from that list or that one for 2$. (A link for a *free* Python 2.7 iOS app will be appreciated!)

8.6) Fun websites to practice or learn Python programming

These websites *could* help you, but **we will not use them**. Feel free to visit them, and you can use them if you want, **on your own**.

- `code.org/learn` and « *The Hour of Code* » are a great collection of tutorials for kids, but can be an easy and fun way for you to start (watch this 1-minute promotional video: `youtu.be/dU1xS07N-FA`).
- `www.CodeAcademy.com` is a brilliant website to help you learn Python. As they said: « *Learn to program in Python, a powerful language used by sites like YouTube and Dropbox* ». This course is estimated to take less than 13 hours, starts from the basis, and covers almost everything we want to cover in our CS101 course.
- `HourOfPython.com` is short but interesting. Two small tutorials: « *From Blocks to Code* » which starts with visually helpful blocks to go to the code underneath and learn some basics of Python, and a « *Visual Introduction to Python* » which aims to give a visual introduction to code using Python and some moving *turtles*!

These other websites are not focused on Python, but can be interesting for you, **in the future.**

- The famous `KhanAcademy.org` has a good intro to CS course (with Javascript instead of Python).
- `www.CodeAvengers.com` is a website designed to easily learn HTML, CSS or Javascript, in order to create webpages, games or apps. This aspect of programming will **not be covered** by CS101, but will be by CS412 course (CSE branch only). If you are curious about that aspect, we would recommend you *to wait for the summer break!* (three months: May, June and July 2015).

8.7) Some powerfull Python library that makes you want to write cool projects right away

- To create Graphical User Interface (GUI), Python natively provides the **Tk module**, but can also use the world-champion **QT framework** (with PyQT), the Linux-oriented **GTK+** (with PyGTK), the multi-platform Kivy (running on Android) or wxPython all are excellent frameworks in Python for building softwares and games with a GUI. In one week in April, if possible, we will probably cover the bases of GUI design, by using Tk.
- The **MatPlotLib** project is an excellent scientific plotting tool for Python. With **NumPy** and **SciPy**, it allows to use Python in a way very similar to MatLab. A very good introduction to scientific computing using Python is these lectures notes on `jrjohansson.github.io`.
- The **SageMath** project provide a common scientific interface based on Python and hundreds on free and open-source modules, shipped together to provide the best environment to do *abstract* mathematics computation in Python.
- **SciKit-Learn** and StatsModels are two excellent toolboxes for Machine Learning and Data Mining, implementing the state-of-art algorithms in an *efficient and fully documented way.*
- Similarly, the Pandas project is a full-featured toolbox for data analysis.

- Python also comes ready for the web, with Django, or the Sphinx documentation tool.
- **PyGame** is a pretty cool and complete framework to write 2D games in Python.

---

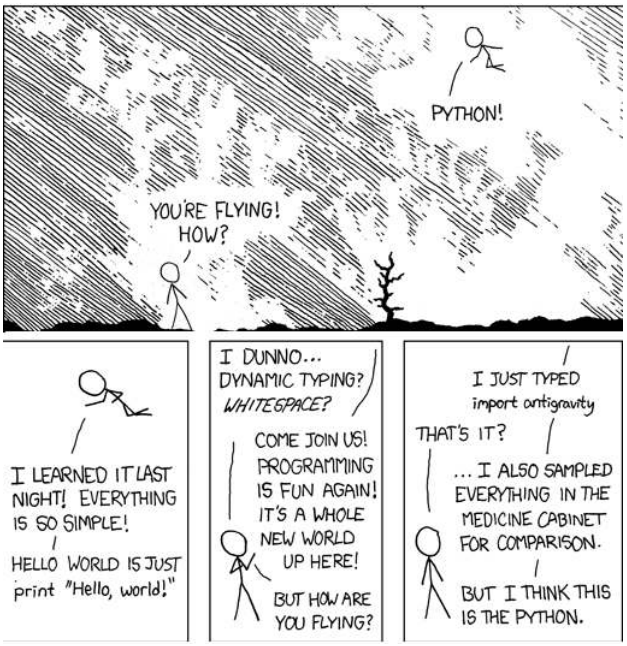
9) Credits and License

9.1) Credits

This document has be written by Lilian Besson for the Computer Science 101 course at Mahindra École Centrale, in November 2014.

Feel free to contact us for any reason, by email at `CS101@crans.org` or with that (anonymous) form at `perso.crans.org/besson/contact/en`.

9.1) License



Python comic from `xkcd.com/353`.