List of ideas for the project for CS101 at MEC

Here is a list of ideas for the project. This document is still in progress, and need to be conclude!

- We need 6 or more different project ideas.
- Each question paper will be given to a certain number of team. One team by section will have a paper, with 4 or 5 peoples by team.
- We will first ask them to create teams as they want (inside each sections, A1-B4), and then we can randomly create/complete groups if needed.

Goal and things to be cautious about

- Do our best to **avoid / detect cheating** (between 2 teams with the same paper, or from Internet).
- Ensure that they can ALL participate (the CS lab is open everyday until 8pm, they can use their own laptop).
- Be sure that they learn useful skills, and use **open-source and free tools** (they will have to submit through Moodle and email, by uploading a 7z/zip archive, containing a certain numbers of documents following the standard of open-source projects, see below)
- Be sure that **they all take part of the project**. During the last week of April (27-04 01-05) we will have a 5-minute meeting with each of the teams, in order to ask subtle questions about concepts and tricky parts of their code (e.g. two questions to each member). To be faire, we need to inform them about that detail.

What do they have to produce and give back

1. They have to submit a `zip` archive named `SectionSSS__TeamYYY__Project1_CS101_2015.zip`, with `SSS` being the section (A1 to B4) `YYY` being the number of the team (from 1 to 6, which is also the project/paper number).
2. That `.zip` archive has to be sent by email to (`CS101at`crans.org, or Arya's email, or .. ?) or uploaded on Moodle. **ONLY ONE SUBMISSION BY TEAM IS POSSIBLE.** They will have to agree on which version they decide to send, and who takes care of it (team coordination).

Content of the zip file:

A complete example of what they have to produce could be available on Moodle? For a « dummy » project.

- one file named `AUTHORS` with a list of students in your team, with one name, roll number and email by line, **like this**:

  Super Man, 14XJ00327, super.man@gmail.càm ... (3 or 4 more students here) ... Bat Man, 14XJ00365, bruce@batman.com
- one file named `LICENSE`, containing the entire copy of the Open-Source license under which your project can be released (even if nothing will be publicly released). This license will have to be chosen in one of these lists: `OpenSource.org/licenses`, or `GNU.org/licenses`. I recommend one of these: the MIT license, the GNU GPLv3 license, the BSD license, or the WFTPL license.
- one file named `README.md` or `README` explaining **in a few lines** what is the project about,

- one file named `INSTALL` explaining briefly how to use the project. You have to list the modules/packages required by your program (if you used any), and any special manipulations required to use and test your program(s),
- one optional file name `TODO`, that can contain a list of point that still need to be worked on. Can be used to assess how advanced the team was on their subject. And if they tried some extra method but have not been able to complete the job, this document will state what would be required to conclude the extra ideas.
- one file named `SectionSSS__TeamYYY__Project1_CS101_2015.pdf` (in the **PDF format**, not Word `.doc.` or `.docx`) containing a **report of their work process**. More details about that report document are below,
- and finally, a list of Python programs (as many as you want, at least one), that solve the problem(s) we gave you.

About the report

We need to specify a little bit what this report should contain. The goal of such document is for you to be trained on technical communication, oriented on these three points:

- state what was your task(s), what approaches you considered and tried to complete,
- optionally, the report can contain remarks of any kind about the project (was it interesting? can we improve it for next year?)

Outline ? The suggested outline is the following:

1. introduction (one paragraph), about the general domain and subject, the task, and the solution that you implemented,
2. first part giving details about the task(s), what model did you chose, what convention did you follow. The mathematical and scientific notations has to be explained here. Reference books and website can be given here (or at the end of the report),
3. details about your Python programs, which one does what, and how. You also have to specify how to use each program (if it is a command line program, is there arguments to give to the script? if not, is there something to change inside the program?). You also have to clearly say what external modules you used. If you program requires any external module that is not included in Anaconda, you need to include a link to the official webpage from where that module(s) can be downloaded (freely).

Tool to write your report ? The suggested way to write this report is either:

- the more common solution: to use Microsoft Word or any word processing software, and to convert to document to PDF before sending it,
- or to start using
  $LaTeX$, the standard for technical and scientific documents. An excellent tool your team could use is the free website `ShareLaTeX.com`, that can allow your team to collaboratively write your report.

Style to follow for your programs:

Conventions

Your team has to follow the Python typing conventions, as we do in lectures, labs and exams from January.

Your programs will be automatically read by the pylint software, than you can download freely and use easily within Spyder (on Windows, type `conda install pylint` or `pip install pylint` on a DOS console, when being connected to Internet).

A too important number of syntax or style warnings or errors in one of your Python program might reduce your final mark for the project.

`@author` in the doctstring of each program

As Spyder is doing it automatically, we ask you to include a line like this one in each of the docstring (the big string between triple quotes `« »"`) in the beginning of each Python program:

```
@author: Team 1 Section A3
```

CS101 Project

When ?

In March/April, 4 weeks.

Ideas (4 to 8, 6 will be good)

Some project ideas:

Numerical integration (1D, 2D and more) Implementing Riemann's rectangle method, basic Monte-Carlo method, and then triangle method, the Simson method etc. Monte-Carlo method can be used to compute integral in 2D (surface) and 3D (volume) and more (cf. Vijay's mail).

Bonus: graphical representation of the methods (plotting the rectangles, the triangles etc under the curve).

Constraints optmization A program to automatically produce the future time tables? Cf. Abhijit's mail.

Matrix operations

- Matrix (as list of lists) operations: sum, subtraction, multiplications, integer power (two approaches, basic one in $O(n^4)$ or optimized one $M^{(2n)} = (M^n)^2$ and $M^{(2n+1)} = M(M^n)^2$ in $O(n^3 \log(n))$). Then the Gram-Schmidt process (can be used to compute the rank), the Gauss elimination method applied to computing the inverse (M^{-1} when defined) and the determinant.

Bonus can be to implement this with OOP: Matrices as a class, with proper overloading of Python's operations (`+`, `-`, , , `/`, etc), by wrapping the functions inside special method `__add__`, `__pow__` etc. The goal is to be able to use matrices as any Python numbers: `M1 + (M2 - M3)**4` for instance.

1D optimization and differentiation (with plotting)

- Finding the zero of a nice function with different approaches (dichotomy/binary search, Newton's method, method of the secant, etc). **Bonus:** graphical representation. Application to computing a square root of any number, and a n^{textth} root.

- Numerical derivation (1D) and gradient (2D, 3D) ? **Bonus:** graphical representation

2D optimization and differentiation (with plotting) Introduction to convex optimization in 2 dimension: gradient descent and more advanced methods.

Numerically compute a gradient, and implement the basic gradient descent method. Extra work can be to implement better methods (conjugated gradient descent, optimal steps gradient descent) etc.

Bonus: graphical representation, in 2D it is possible to show the gradient descent (one picture). Like this example.

DE and DE systems. Solving and plotting solutions to Differential Equations, and DE systems, with basic methods (Euler method, or the matrix one) or the good module.

- TODO: include suggestions from Vijay.
- FIXME which one? SymPy?

About cryptography About prime numbers ? Computing prime numbers, many approaches (efficiency is important), and some applications to cryptography.

Simple implementation of a Cesar cypher protocol and hacking the Cesar cypher, and maybe a basic RSA protocol (with small keys) ?

> More ideas from MA101, PH101, EE101, MA102 CB101, EE102 ? Or from the huge list of problems from `projecteuler.net` (by selecting only easy and short ones – but not too easy!).