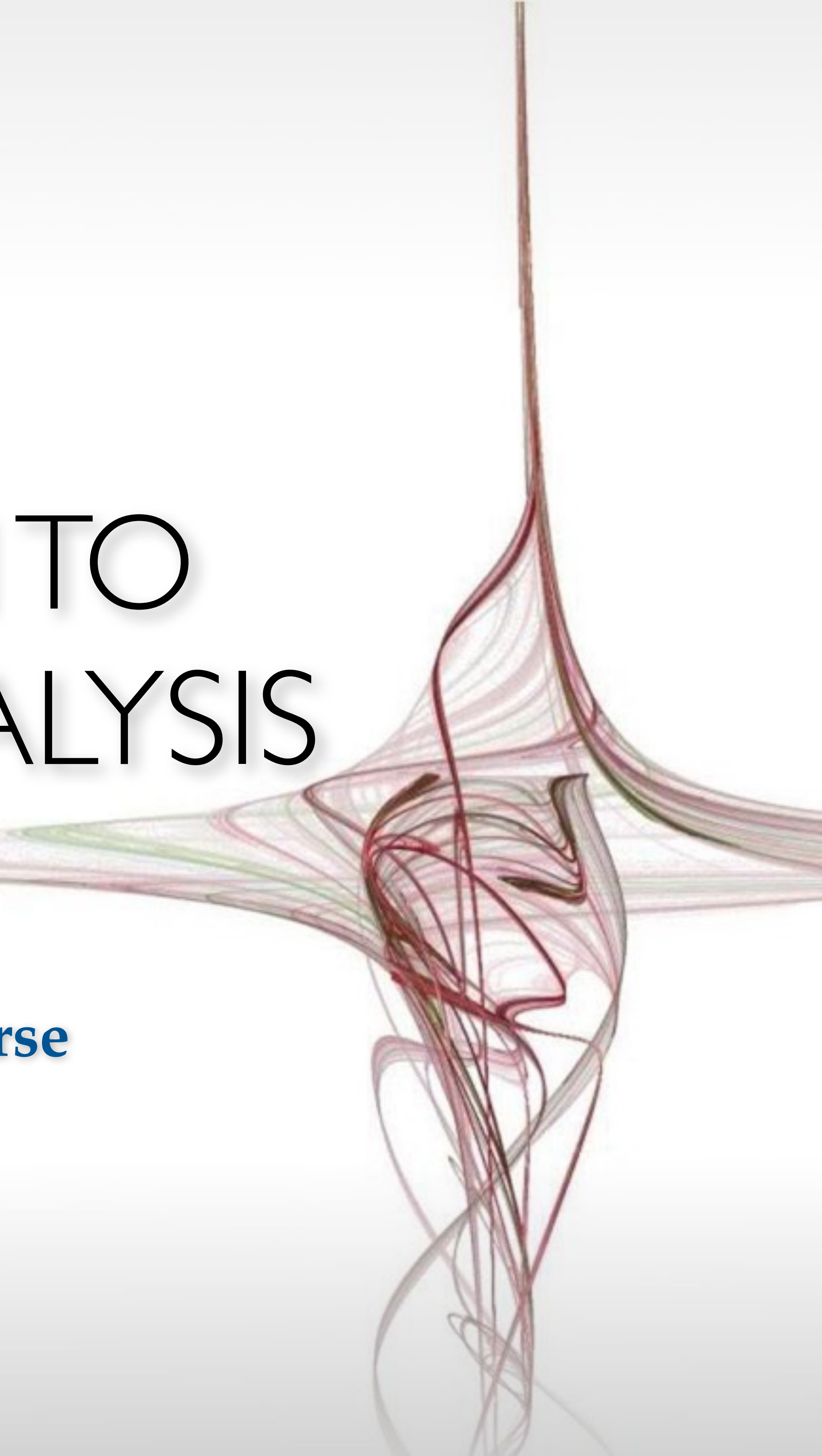
2013

INTRODUCTION TO NUMERICAL ANALYSIS



Lecture 0-1:
All you need to know about this course

Kai-Feng Chen
National Taiwan University

OVERVIEW

- This is a **quasi-laboratory course**, since no one can learn how to do numerical analysis only by listening to the lectures and take notes (*and only do the homework once a while!*).
- **PRACTICE** is extremely important:

  ⟹ You will never learn the calculus without doing lots of differential/integral exercises, right?
- **You are strongly recommended to bring your laptop to this lecture and practice during the lecture.**

  (hopefully the battery life of your laptop can run over 3 hours!)
- If you do not have a laptop, you are encouraged to work with your classmate who has laptop during the lecture.

A QUASI-LABORATORY LECTURE

One will never learn any musical instrument without real practice.

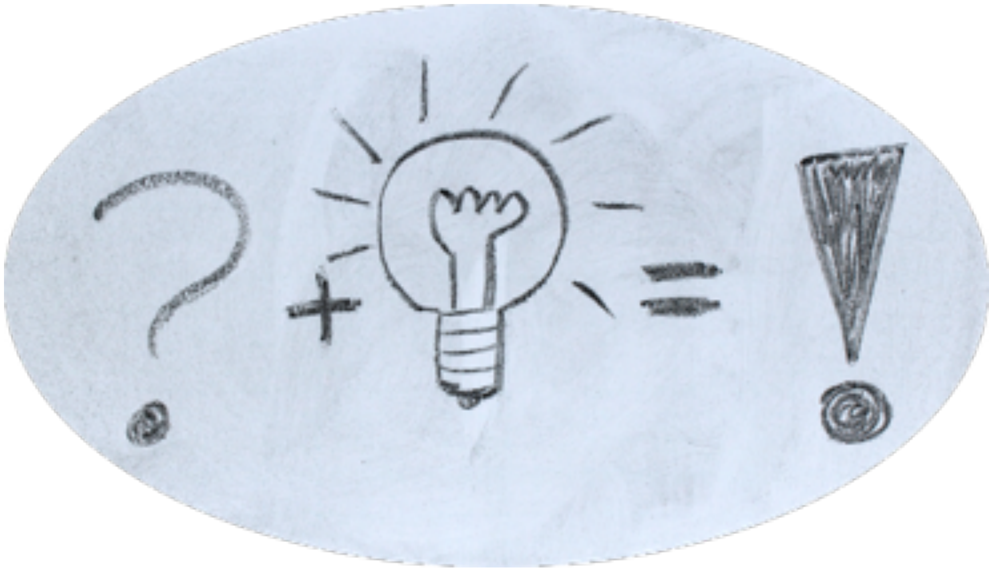
Simply watching a couple of great performances will never work!

A QUASI-LABORATORY LECTURE

- I will not just "*blah-blah*" throughout the whole 3 hours. Instead, 1/2~2/3 of the total time will be devoted to an introductory lecture with slides.
- **Rest of time will be used for practice/exercise/problem-solving, just like your laboratory courses!**
- There will be also some short "trial periods" during the main lecture, which allows you to try something easy.
- Please also stop me when you run into any difficulties or troubles throughout the whole lecture.

THE GOAL OF THIS COURSE

- Learn how to solve a problem with computers rather than with a pen and papers.
- Learn how to utilize the existing computing tools/functionalities, or build your own tool.
- Learn how to formulate a problem into a simple program that can give you an answer clearly and quickly.
- **And have fun with them! (most important!)**



WHAT ARE WE GOING TO DO?

- We will use **PYTHON** as the base language.
  (*well, python is probably the easiest computer language to learn and I would assume many of you already learned it from other course!*)
- We will discuss how to use python and the associated numerical/graphical libraries to solve scientific problems, which could be beneficial to your own physics (experimental/theoretical) studies in the near future.
- It does not mean you do not need to learn other computer languages (e.g. C++, fortran, R, Java, php, etc.) in the future for your own work. Hopefully you will get some more "taste of computing" in this semester.

FOR THE EXPERTS…

- I’m pretty sure some of you already well experienced in programming.
- Part of this course can be relatively easy for you in this case, and you can probably learn it by yourself without any difficulties.
- **If you are in this situation, I would recommend:**

  ⟹ Discuss with me and maybe we can do something beyond the scope (e.g. a more challenging project).

  ⟹ Become the mini-TA! Come to the class and act as a helper for your classmates (especially during the exercises period!).

SCIENTIFIC COMPUTING WITH PYTHON

- We will use **SciPy** (pronounced like "sign pie") packages in this lecture. See http://www.scipy.org for more information.

  ⇒ **SciPy library**:
  fundamental library for scientific computing.

  ⇒ **NumPy**:
  base N-dimensional array package.

  ⇒ **Matplotlib**:
  Comprehensive plot making.

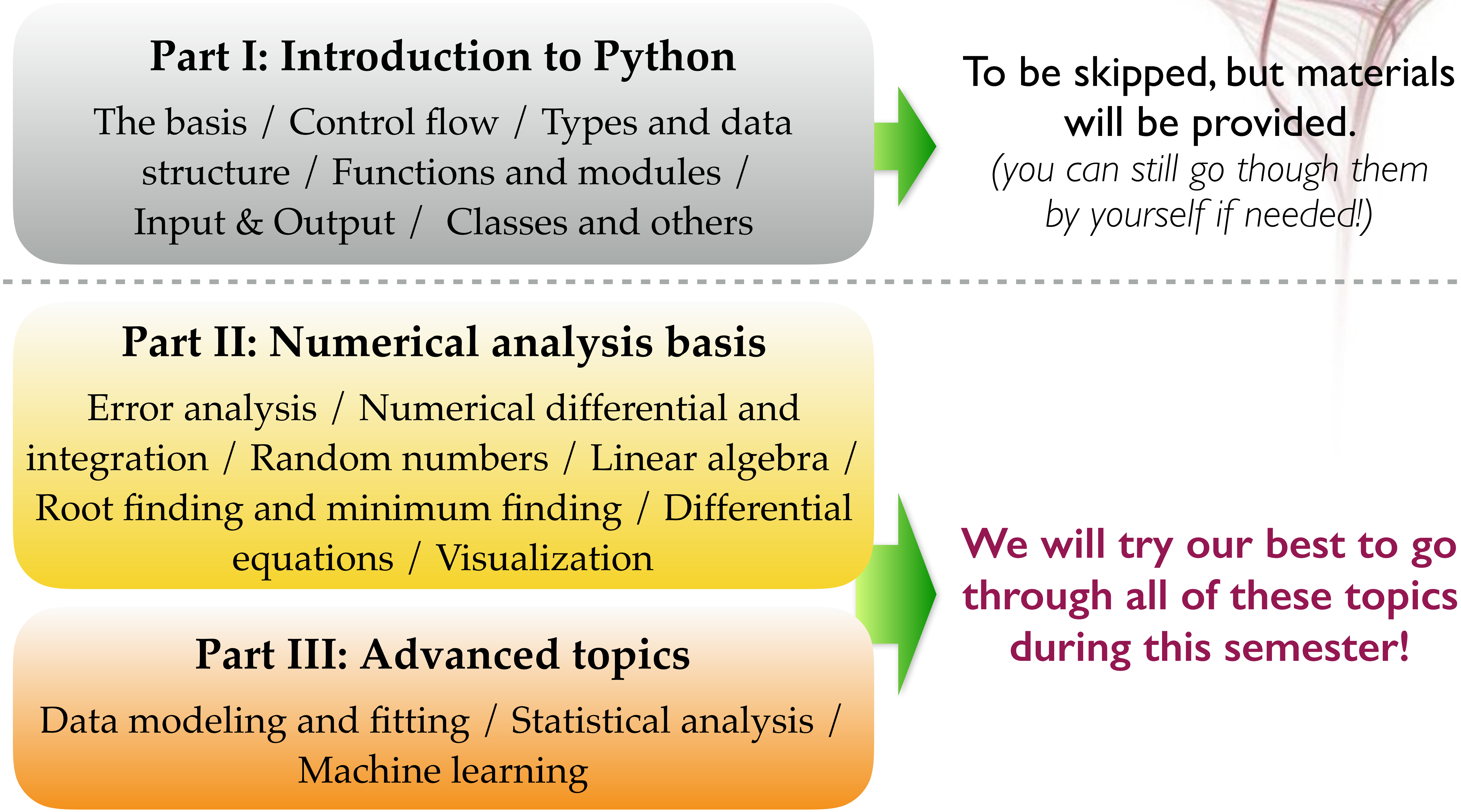
  ⇒ **Scikit-learn**:
  Machine learning tools.

- Several data managing related packages you might be interested in: **scikit-image** (image processing tools), **panda** (data structures).
- Also consider the **IPython** as a nice enhanced interactive console.

SCIENTIFIC COMPUTING WITH PYTHON (CONT.)

- Few other additional packages will be used during the lecture:
  - ⟹ **VPython**: http://vpython.org
    Easy creating 3D animations and visualizations. *Many of you may have used it before with your general physics lecture!*
  - ⟹ **iminuit**: http://iminuit.readthedocs.io
    Using the minimization engine Minuit under python. To be used in the lecture about data modeling and fitting.
  - ⟹ **Keras** & **TensorFlow**: https://keras.io
    Easy building your neural network! To be used in our (not-so-)deep-learning lecture.
- We will find some more information about these packages when we are going to use them!

OUTLINE



Part I: Introduction to Python

The basis / Control flow / Types and data structure / Functions and modules / Input & Output / Classes and others

To be skipped, but materials will be provided.
(you can still go though them by yourself if needed!)

Part II: Numerical analysis basis

Error analysis / Numerical differential and integration / Random numbers / Linear algebra / Root finding and minimum finding / Differential equations / Visualization

Part III: Advanced topics

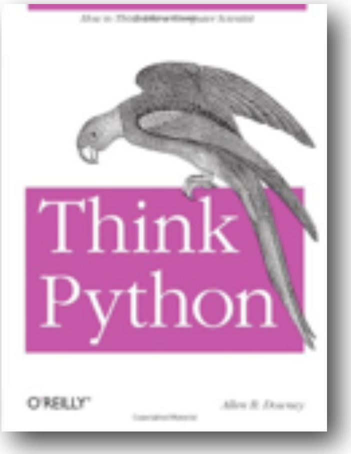
Data modeling and fitting / Statistical analysis / Machine learning

We will try our best to go through all of these topics during this semester!

TEXTBOOK & REFERENCES

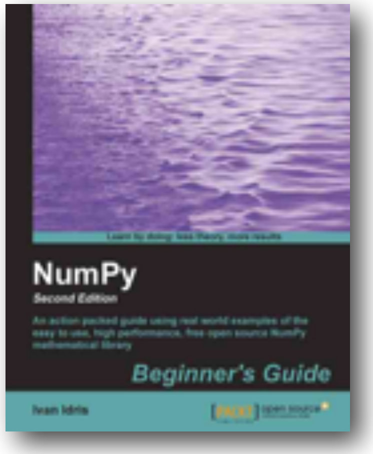
- For **python** itself, most of the information can be found online. Getting a printed textbook is not really required. A couple of nice online books/documents are available:
  - □ Python.org tutorial: https://docs.python.org/3.6/tutorial/index.html
  - □ Think python (✷*slides are based on this book*): http://www.greenteapress.com/thinkpython/html/index.html
  - □ A byte of python: http://swaroopch.com/notes/python/

Caveat: the documents/books may be prepared for python 2 or python 3. Please note they can be different!



TEXTBOOK & REFERENCES (II)

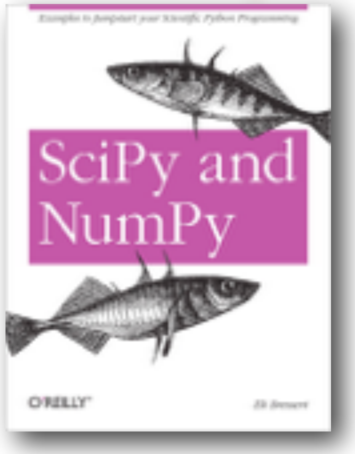
- For **SciPy (and NumPy)**, there are already some document available on the official website and some online e-books:
  - □ Official web document: http://docs.scipy.org/doc/
  - □ NumPy Beginner's Guide: http://it-ebooks.info/book/2847/
  - □ SciPy and NumPy book: http://it-ebooks.info/book/1280/



NumPy

Second Edition

Beginner's Guide



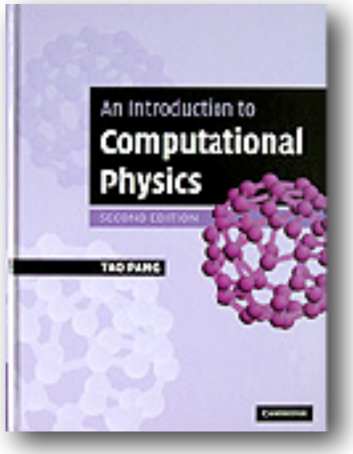
SciPy and NumPy

O'REILLY

In principle you can always find the help online, so it is not really required to have a printed book.

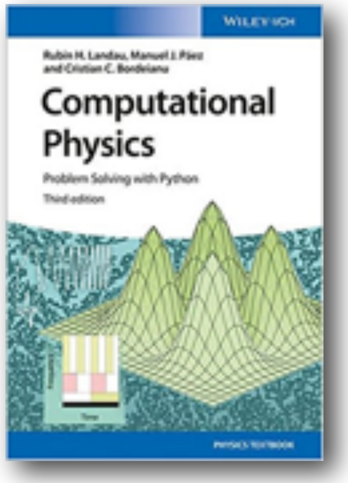
TEXTBOOK & REFERENCES (III)

- References for computational physics & algorithms (python and non-python):

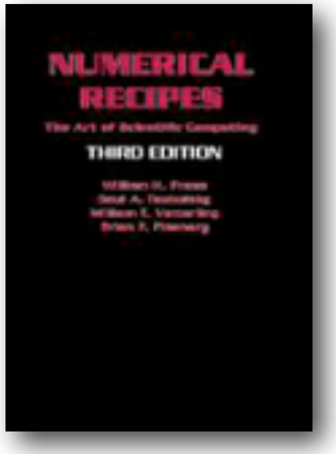


An Introduction to Computational Physics
by Tao Pang
2nd Edition (2006, 2012)

Introduction to Computation and Programming Using Python
by John V. Guttag (2016)



Computational Physics: Problem Solving with Python
by Rubin H. Landau et al.
3rd Edition (2015)



Numerical Recipes:
The Art of Scientific Computing
by William H. Press
3rd Edition (2007)
http://www.nr.com/

There are still many other computational physics or algorithm text book can be found on the market.

THE ULTIMATE REFERENCE

Whatever, Google tells you everything...

EVALUATION

- **Homework:**
  - □ Exercises will be assigned for most of the topics.
  - □ Please hand back (*upload*) the code within 2 weeks.
- **Quizzes:**
  - □ Will be assigned in April or May.
  - □ Time limit: 2 weeks.
  - □ *Googling the answer is allowed. Discussions are also allowed.*

Surely this sounds too relaxed!?

EVALUATION (II)

- **From the basic grading toward the final goal:**

If you fulfill all of the minimal requirements (homework & quizzes, no delayed hand back)

You have to collect 3 gold coins just like the Super Mario game!

EVALUATION (III)

How to collect the "golden coins":

□ Finish all of the homework assignments, all on-time!

□ Be the first 2 people uploading the answer to any one of the quizzes during the midterm week. Or hand-in the best (most elegant) answer!

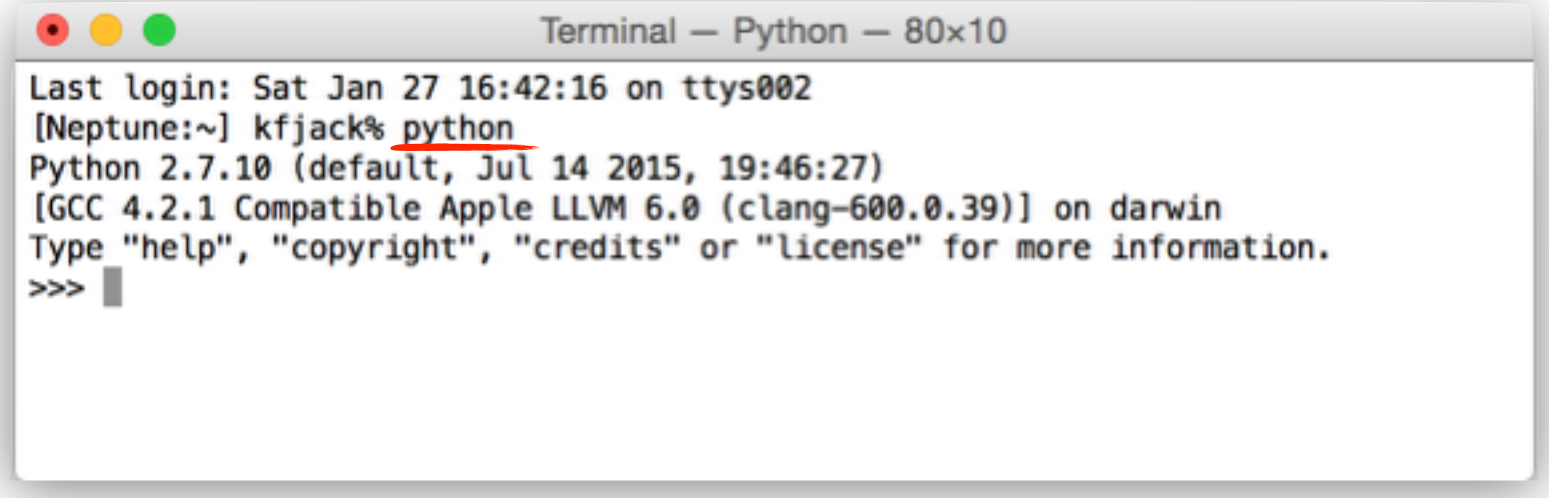
□ Tournament: entering the semi-final round.

□ Additional project presentation (*strongly recommended if you are already familiar with coding*).

By default everyone can get **B+~A+** easily.
Surely you will loose the grading if:
1) Delayed/No hand-back of the homework
2) No hand-back of the examines

GETTING START

- If you are using any unix-like operation system, such as **Linux** or **Mac OSX**, usually a python is pre-installed in your system:



```
Last login: Sat Jan 27 16:42:16 on ttys002
[Neptune:~] kfjack% python
Python 2.7.10 (default, Jul 14 2015, 19:46:27)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

You can simply start a terminal and type "python". Note the default version can be still python 2!

- For **Windows**, in principle you can download the python from the official download area: http://www.python.org/download/

But wait – this is not enough!

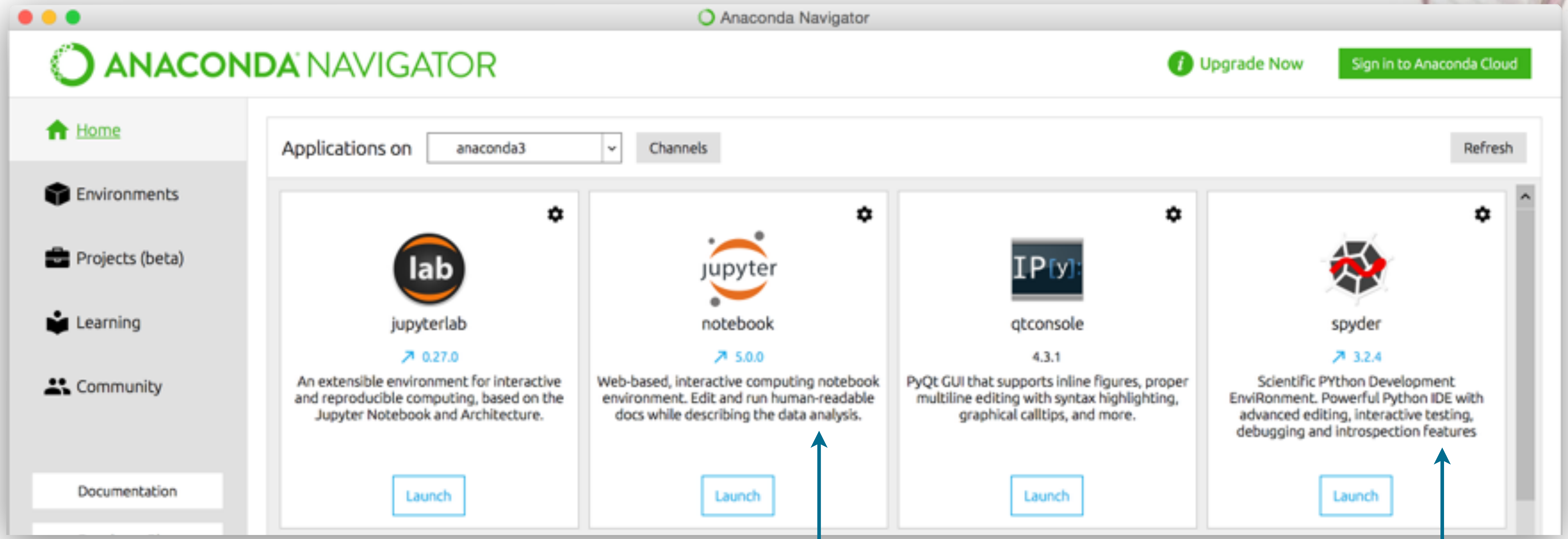
GETTING START (II)

- Since in this lecture we will use SciPy and NumPy, it will be much easier if you can install all of them together. There are some integrated package available:
  - □ Option #1: Get the "**Anaconda**"
    http://continuum.io/downloads.html
    Also simply download it and install. It requires a little bit more command line working experience but the support is good.
  - □ Option #2: Get the "**Canopy Express**" (free version):
    https://www.enthought.com/store/
    Just download it and install. It comes with all the needed packages already, together with a nice IDE ready to go.

Basically these two options support Windows, Linux, and Max OSX.

GETTING START (III)

- If you installed the **Anaconda**, this is what you will see:



jupyterlab
↗ 0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

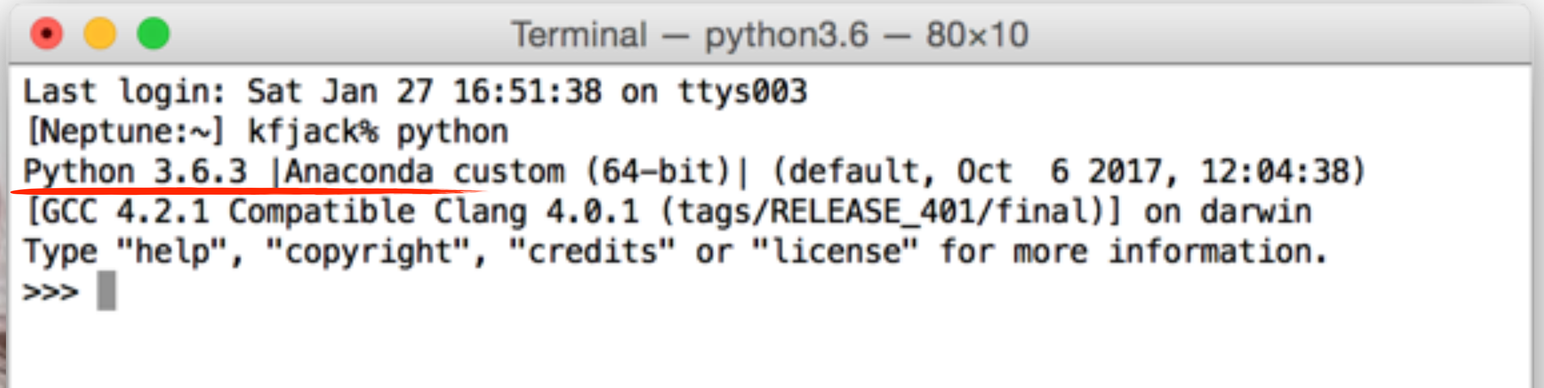
notebook
↗ 5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
↗ 3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

The jupter-notebook is something similar to Matlab.

If you want to use an IDE, you can try this one (spyder)!

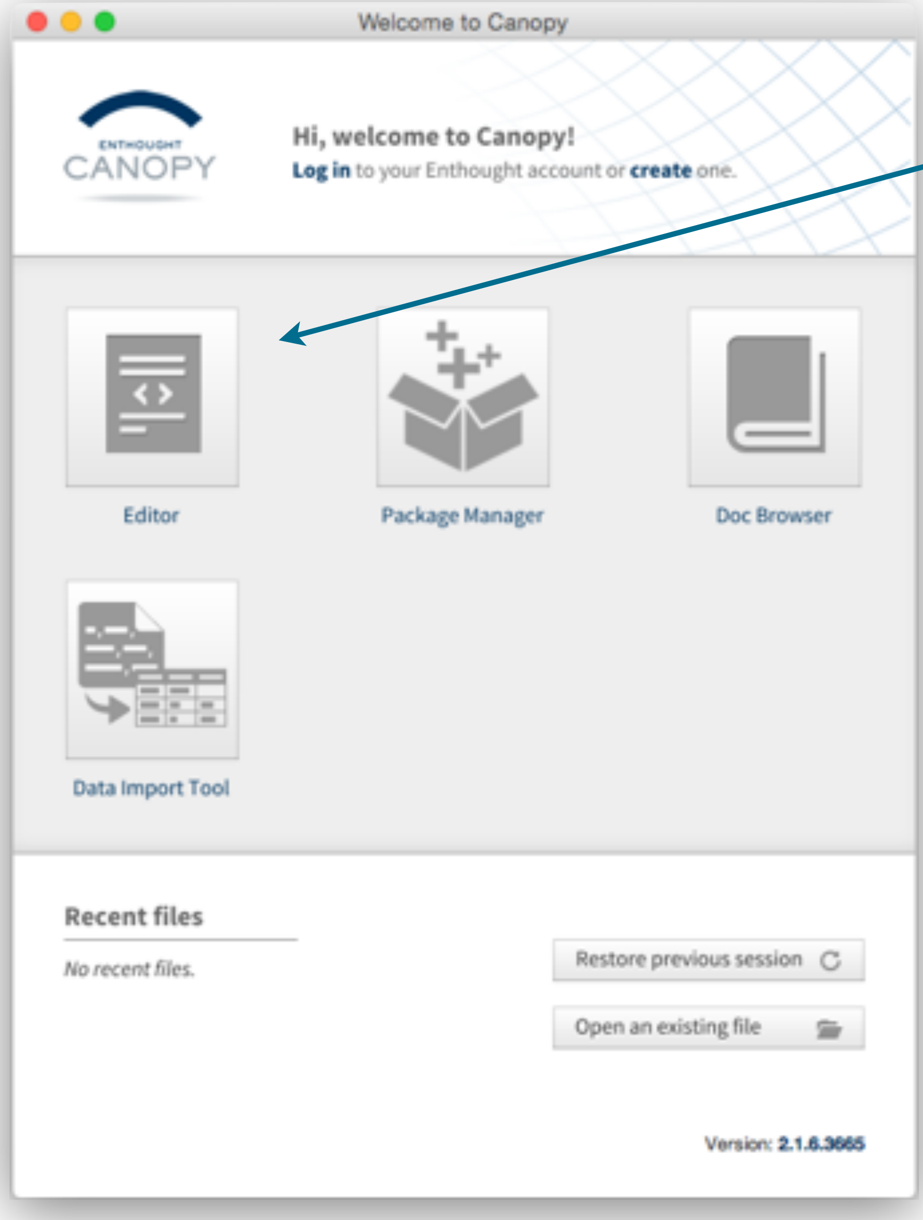


```
Last login: Sat Jan 27 16:51:38 on ttys003
[Neptune:~] kfjack% python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017, 12:04:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> ▋
```

You should see a similar command line integration!

GETTING START (IV)

- If you have installed the **Canopy Express**, this is what you will get:

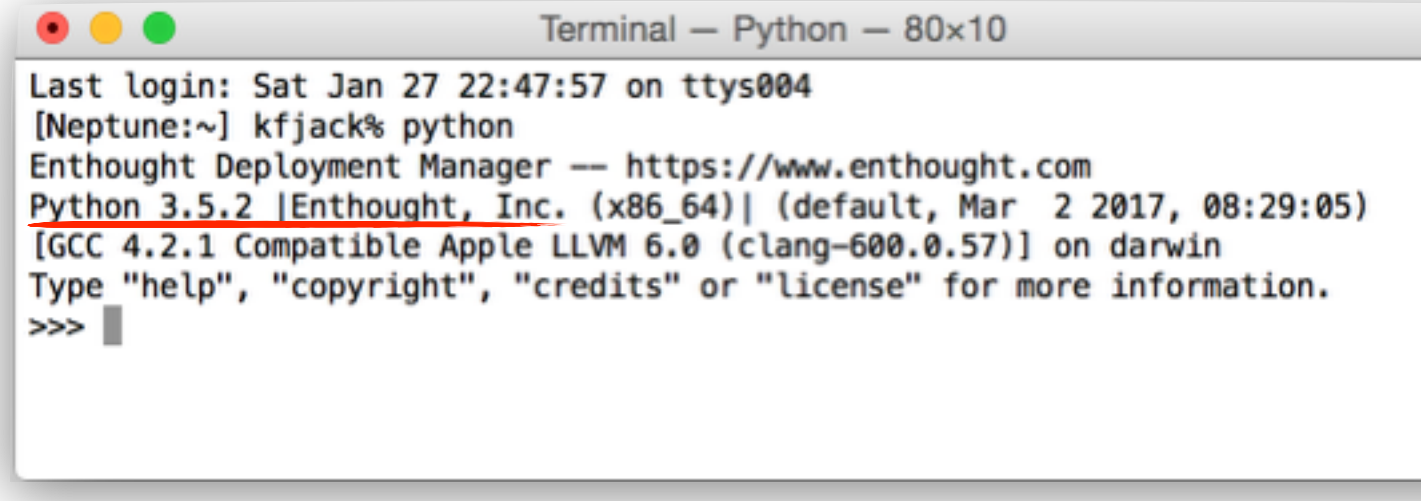


Hi, welcome to Canopy!

Log in to your Enthought account or create one.

Just click the "Editor" to start your coding work with the IDE!

You may also want to check the command line integration:



```
Last login: Sat Jan 27 22:47:57 on ttys004
[Neptune:~] kfjack% python
Enthought Deployment Manager -- https://www.enthought.com
Python 3.5.2 |Enthought, Inc. (x86_64)| (default, Mar  2 2017, 08:29:05)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

COMMENTS

- In principle you can install all of the required packages (**Python +SciPy+MumPy+Matplotlib+Scikit-learn+…**) by yourself without the integrated package like Canopy and Anaconda. But it will take much more efforts before you can actually work. This is not very straightforward for beginners.
- **IDE (integrated development environment)** — In principle this is not really a requirement. We will mostly use terminal (command line) in this course. However, a good IDE can be easier for some people. You can use it if you like. You can try the one came with Canopy, or the **spyder**, or the **IDLE** (which was developed by the original python author Guido van Rossum).

COMMENT: PYTHON 2 VERSUS 3

- Python 3 was released in 2008 already, but if you check the documents or books (and/or the official site) carefully, you may find there are still some issues between python 2 and 3.
- **Basically python 3 does not have the full backward capability with version 2. The syntax is also slightly different.**
- Before python 2 is more adopted, but given python 3 is more and more popular nowadays, in this lecture we will use **python 3.6 (Anaconda version)** as the default version.

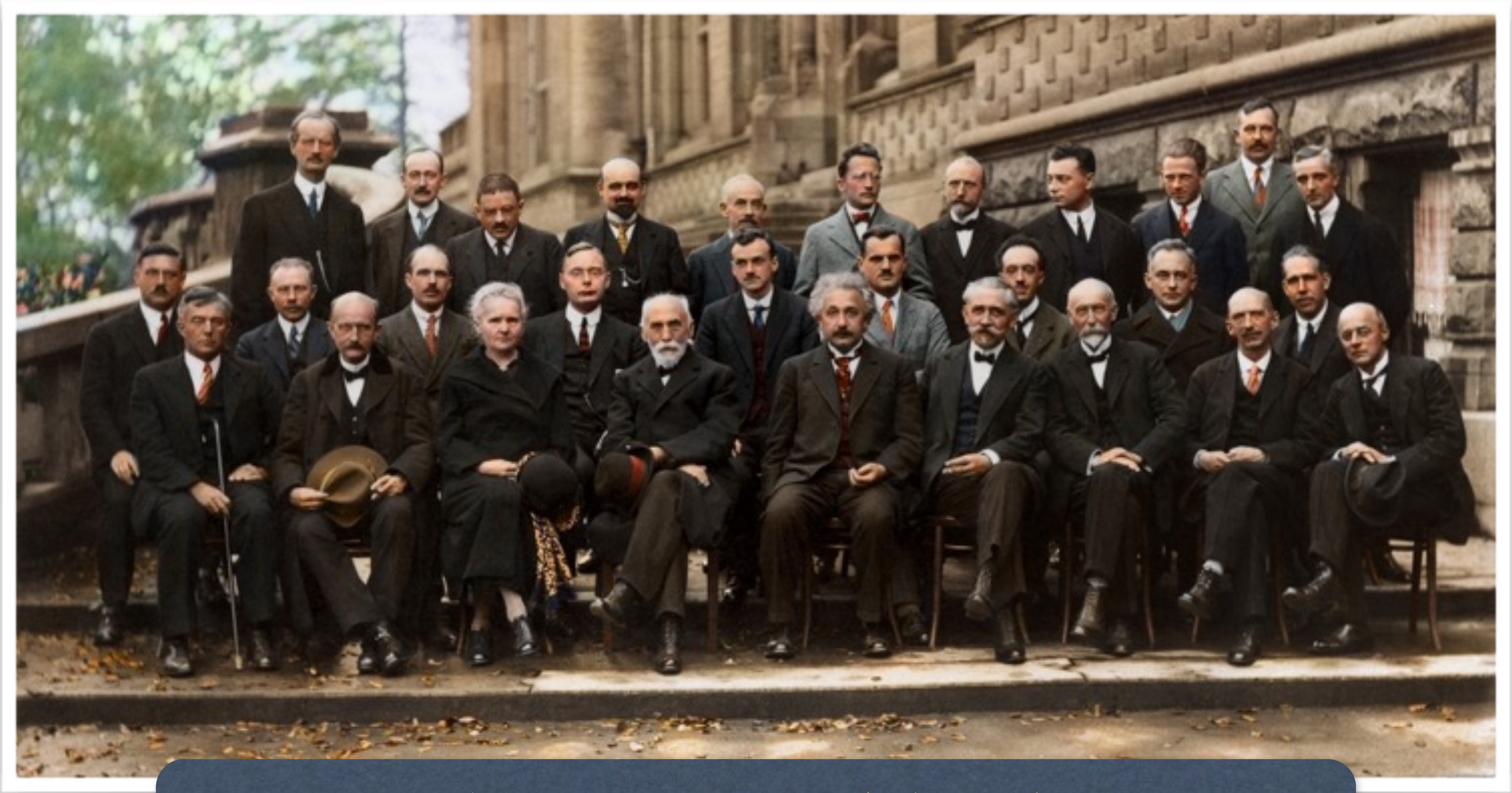
Please do not worry about the exact version for now. The key idea is to learn **how to solve a problem with programming**, not the language itself.

INTERMISSION

- Now it’s the time to get your working environment ready! *(switch on your laptop now!)*
- If you already have a python (whatever version/bundle) installed in your laptop/desktop, you may proceed immediately until we start to use SciPy/NumPy.
- It will take a while to install **Canopy** or **Anaconda**. You can do it later today.
- If you only have a pad/phone, you can even try some of the online python interpreter, e.g.:
  https://www.pythonanywhere.com/try-ipython/
  http://repl.it

A FUN DEMO

- Continue our lecture with a demo problem: show you how things can be sorted out easily, if you know how some coding + google.
- Let me ask — *how many people are there in this photo, roughly?*



You may reply: are you nuts? I do know how to count!

But how about this one? Ya…it might just take a while…

LET'S SOLVE THIS WITH PROGRAMMING

- You may start to think it may take a while to work out a code to count number of people in a photo.
- There might be some existing programs or app that can do such a thing (in some of the cases you will even need to pay!).
- In fact if you know how to do it, it won't cost you more than 10 lines of coding!
- Let me show you a small piece of python code which adopt the **OpenCV library** + **face detection**.

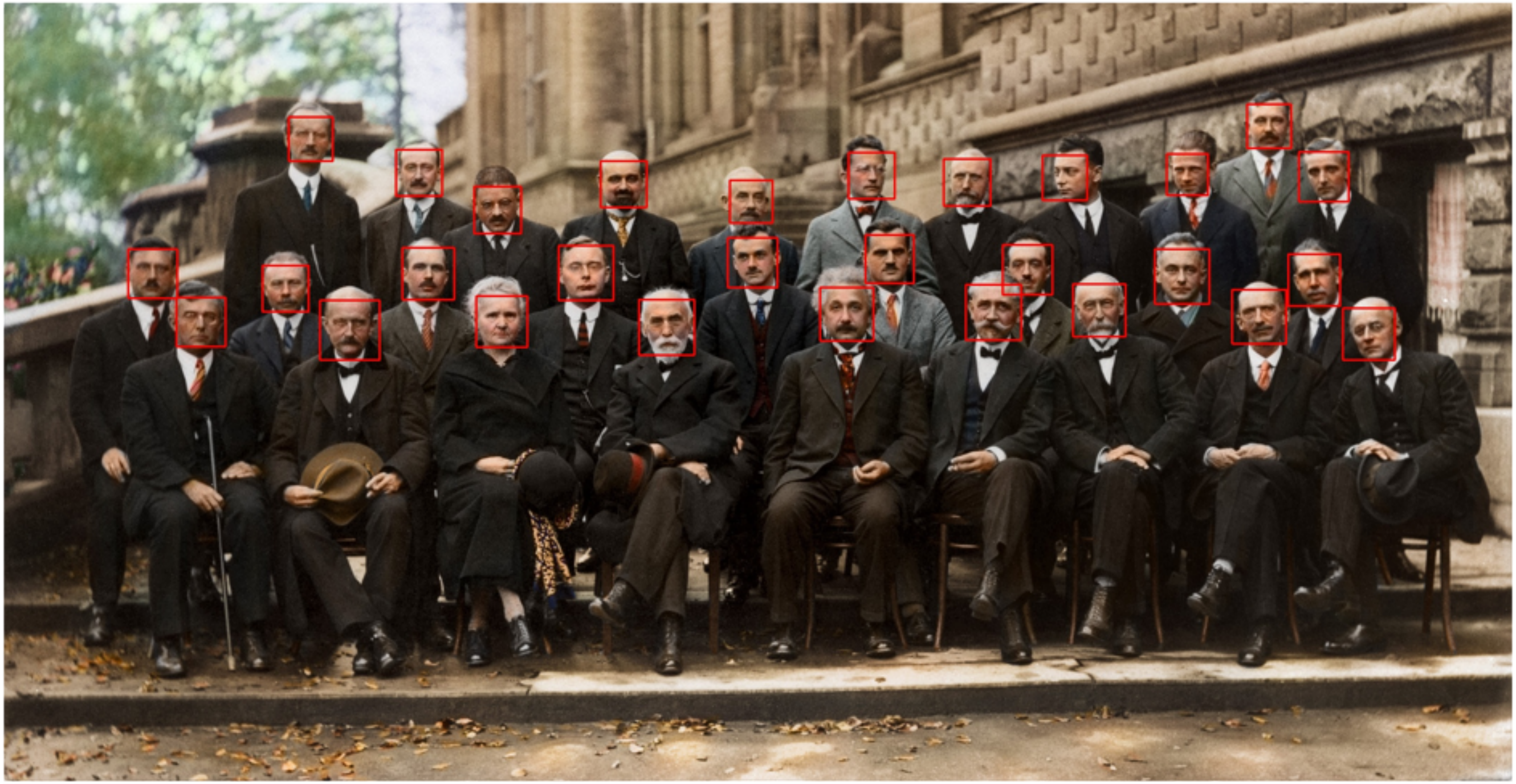
LESS10 LINES TO WORK IT OUT!

```
import cv2  ⇐ include OpenCV

img = cv2.imread('solvay.jpg') ⇐ load the image
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
faces = face_cascade.detectMultiScale(img, 1.1, 5)

print ('How many faces found:',len(faces)) ⇐ print out how many 'faces'

for (x,y,w,h) in faces:                                  add a small red box
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2) ⇐ to the faces found
cv2.imwrite('out.jpg',img) ⇐ save the output
```

count_faces_ex1.py

- Just run it directly, if you have all the needed files!

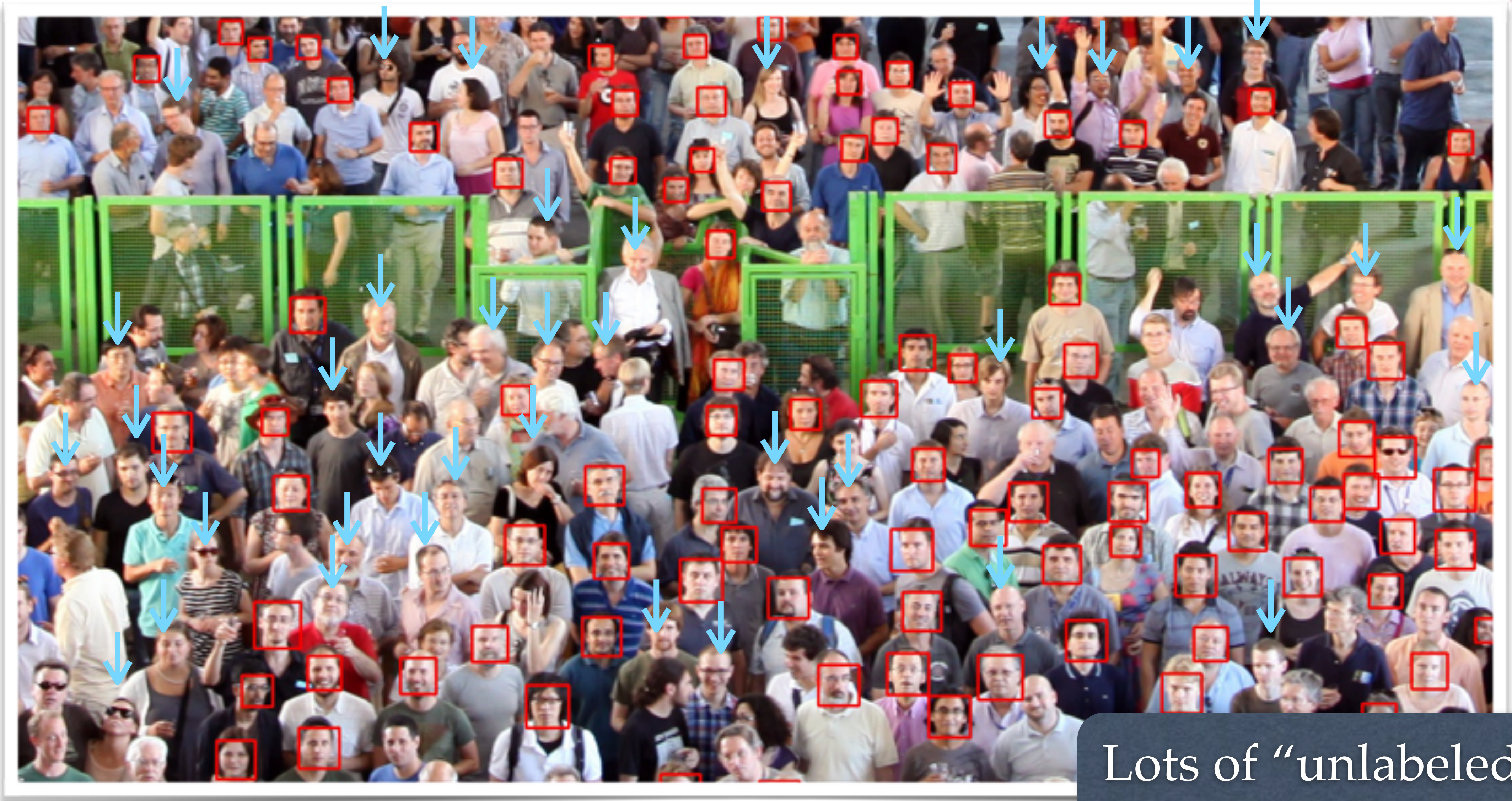
```
% python count_faces_ex1.py
How many faces found: 29
```



Yes, there are 29 people in total!

APPLY IT TO THE BIG PHOTO?

- Let's see what we can get — **211 faces in total?**



Lots of "unlabeled"↓ people.
Need some tuning!

SOME TUNING?

```
img = cv2.imread('cms.jpg') ⇐ just load a different image
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
faces = face_cascade.detectMultiScale(img, 1.1, 5)

print ('How many faces found:',len(faces))
```

count_faces_ex2.py

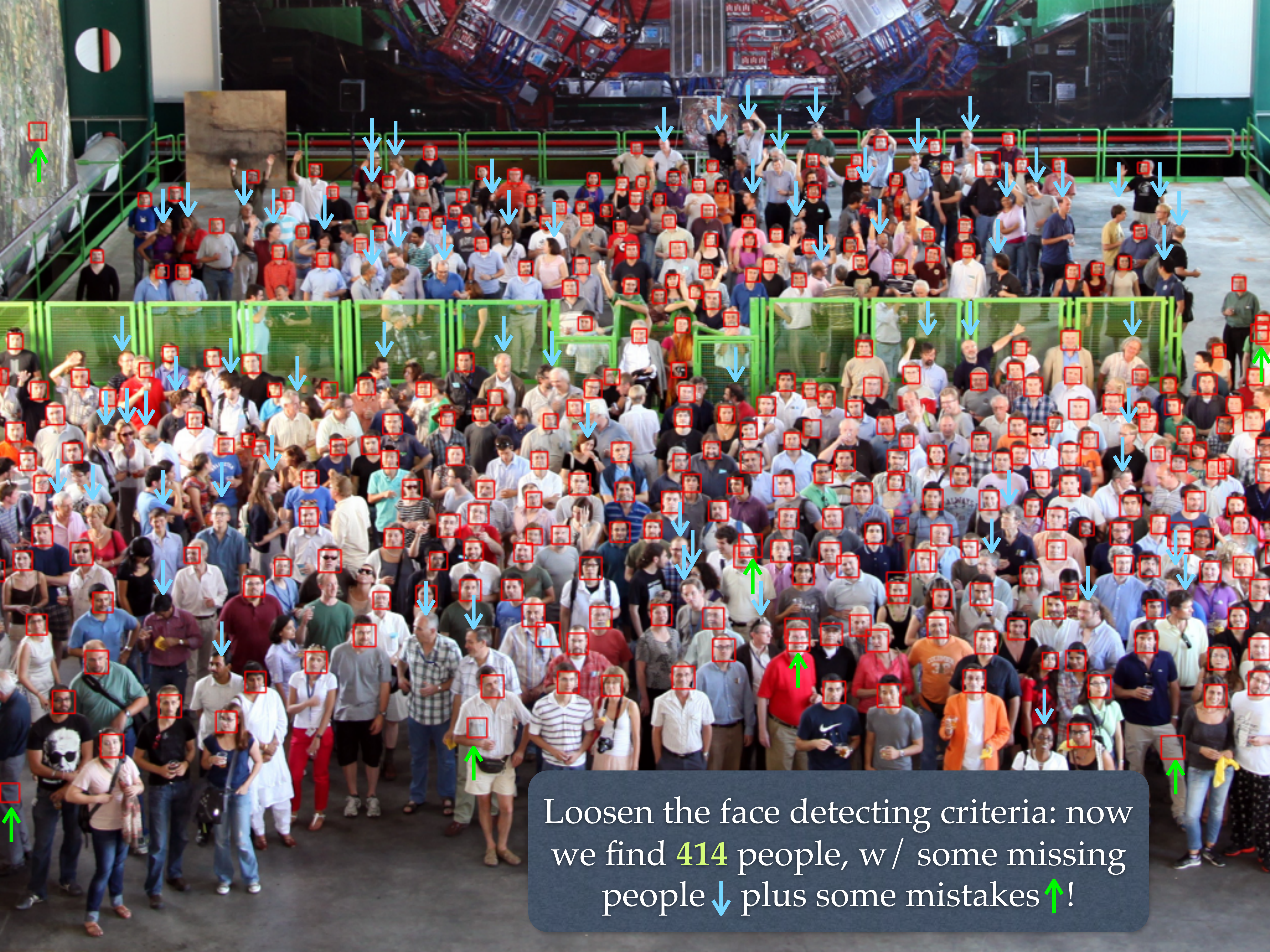
```
% python count_faces_ex2.py
How many faces found: 211
```

- Let's change the parameter settings a little bit:

```
                                                   ⇓ change the arguments a little bit!
faces = face_cascade.detectMultiScale(img, 1.03, 2, maxSize = (50,50))
```

count_faces_ex2a.py

```
% python count_faces_ex2a.py
How many faces found: 414
```



Loosen the face detecting criteria: now we find 414 people, w/ some missing people ↓ plus some mistakes ↑!

COMMENTS

- With only few lines of code one can already get a very rough guess of how many people (more precise — *how many faces!*) in the input photo!
- You many know that in many of the album program can do a similar thing as well: **locate and identify the faces**!
- Just want to show you how a (looks-like-to-be) difficult task can be worked out easily if you know there is such a tool existing.
- **Surely in the real life problem solving will take some real efforts, not just magical few lines of codes!**

Please do not flunk me, professors!