



15-110 PRINCIPLES OF COMPUTING – F21

LECTURE 1: INTRODUCTION AND LOGISTICS

TEACHER:
GIANNI A. DI CARO

The 110 team

Teacher:



Prof. Gianni A. Di Caro
(robotics, AI, ML, MAS)

Course Assistants:

- Rama Sulaiman
- Danagul Azimzhanova
- Temoor Tanveer
- Zhijie Xu
- Andrei-Horia Pacurar
- XYZ

Road map

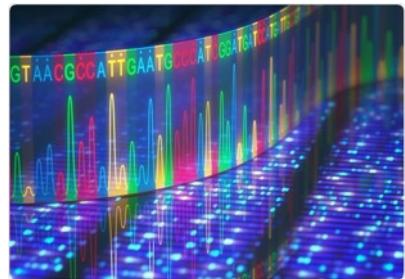
- General overview: what this course is about, general recommendations
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming

Road map

- General overview: what this course is about, general recommendations
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming

Introduction to the *science (art)* of computational problem solving

Daily life, professional activities, leisure, business, ... present a variety of **problems** that ask for finding a **solution**

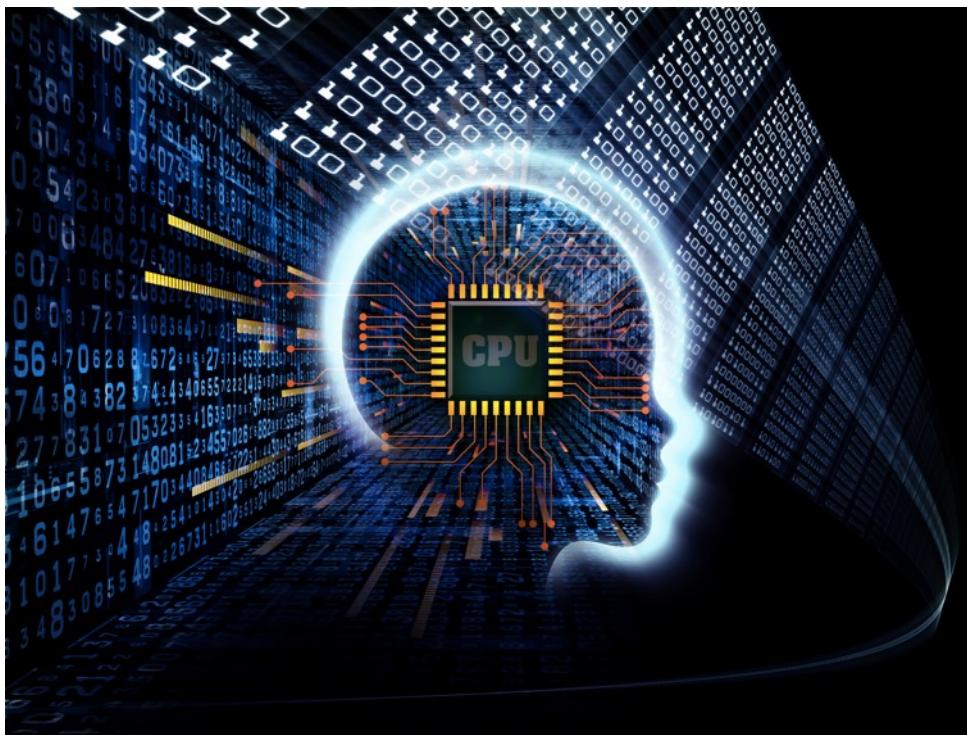


- Different **constraints** (e.g., time, budget)
- Different representations / models / **contexts**
- Different **objectives** (e.g., best nutritional balance, minimum traveling time, win!)
- Different **requirements for the solution** (e.g., provably the best, just one)
- Different **available knowledge** (e.g., chess vs. financial investment vs. path finding)
- Different **degree of (un)certainty** (e.g., poker vs. industrial manipulator)
- Different **dimensionality / number of variables** (e.g., DNA sequencing, root finding, flight control)
- ...

Introduction to the *science (art)* of computational problem solving

Often (usually) these problems are too complex to be **effectively** and **satisfactorily** solved by humans (alone)...

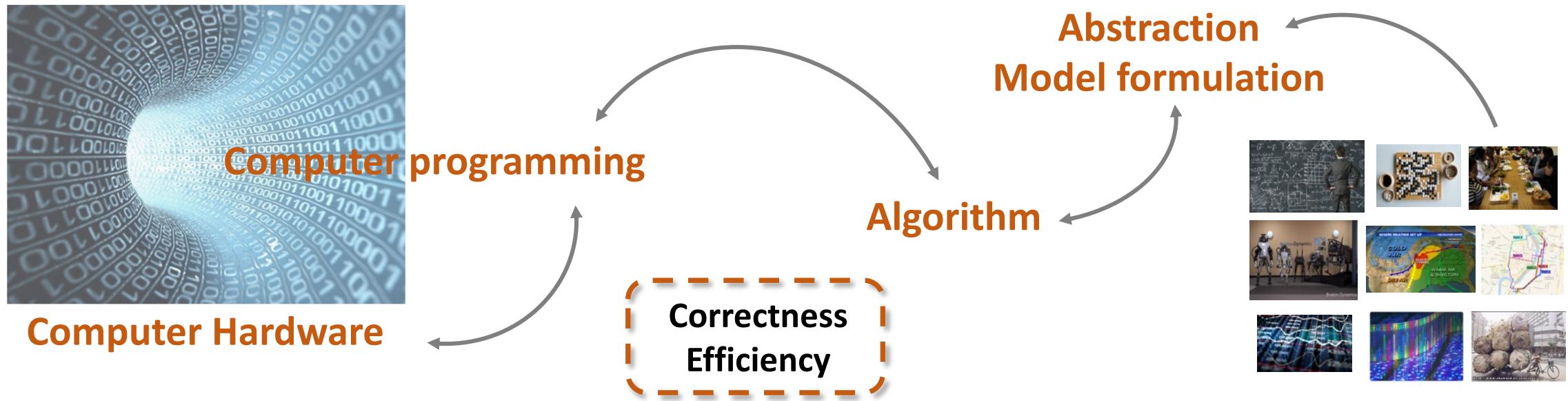
... **computers** can effectively help (us) solving the difficult problems!



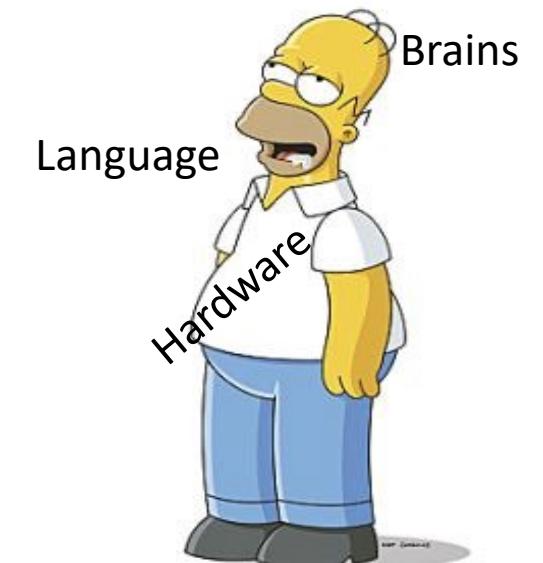
- Perform **calculations**, at impressively high rates
 - + - * / > < = ≠
- Fastest computer in the world (IBM AC922): $\approx 10^{15}$ operations per sec on real numbers:
1,000,000,000,000,000 ops/sec
- Mega = 10^6 , Giga = 10^9 , Tera = 10^{12} , Peta = 10^{15} , Exa = 10^{18}
- **Store** and **retrieve** data and results of calculations
 - 160TB internal memory, HP: 160,000,000,000,000 bytes
 - Single storage units > 100 TB: 100,000,000,000,000 bytes

... how do we connect *problem + humans + computers* for effective problem-solving?

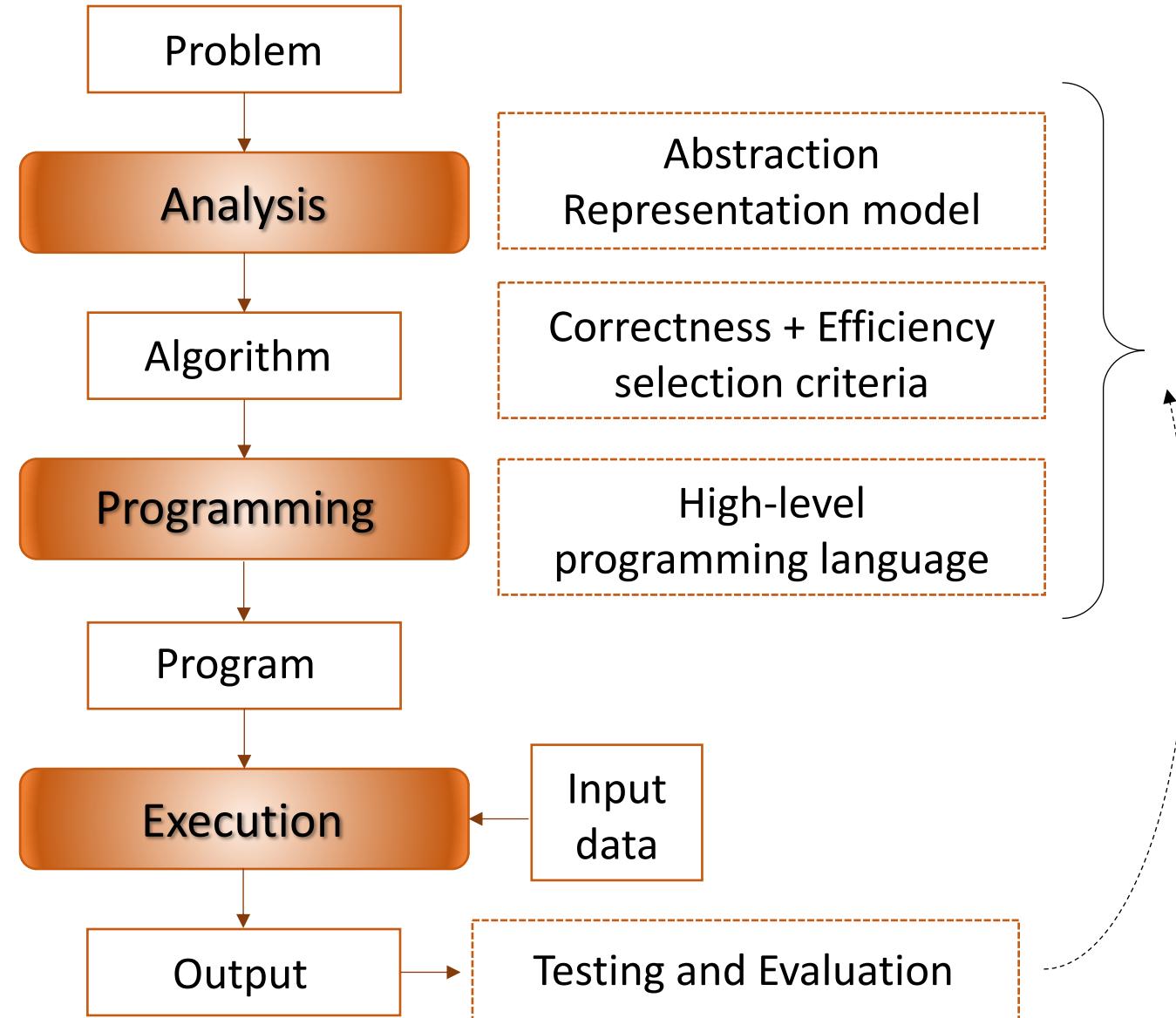
Introduction to the *science (art)* of computational problem solving



- ✓ **Algorithm**: What to do (*step by step*) to (efficiently) solve the problem based on the defined problem abstraction / model → **Problem-solving recipe**
- ✓ **Programming (coding)**: Use a high-level (computer) *language* to precisely describe (code) the algorithm → The code is used to tell the computer to do the things prescribed by the algorithm [**what to do and how**]
- ✓ **Computer hardware**: The high-level program is translated into a *machine-level* language which is then **executed** by the specific computer hardware



Introduction to the *science (art)* of computational problem solving



- **Algorithm:**

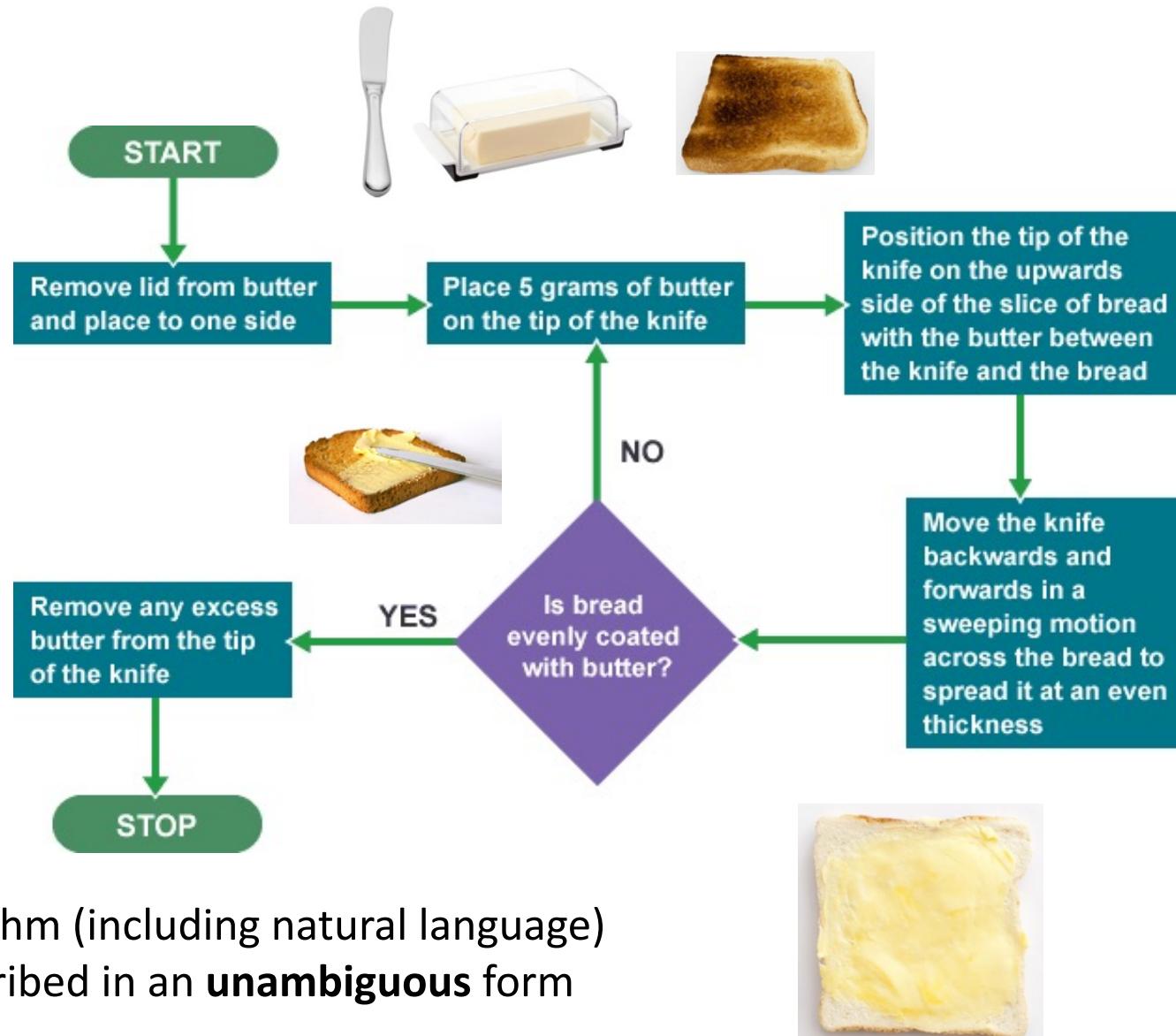
- ✓ A finite list of instructions that describe a **computation**
- ✓ when the instructions are executed on a provided set of inputs, the computation proceeds *step by step* through a set of well-defined states (configurations)
- ✓ eventually, it ends, with some outputs being produced

- **Program:**

- ✓ Algorithm encoding using a language that the computer understands
- ✓ > 700 *programming languages!*
- ✓ Primitive constructs, syntax, static semantics, semantics

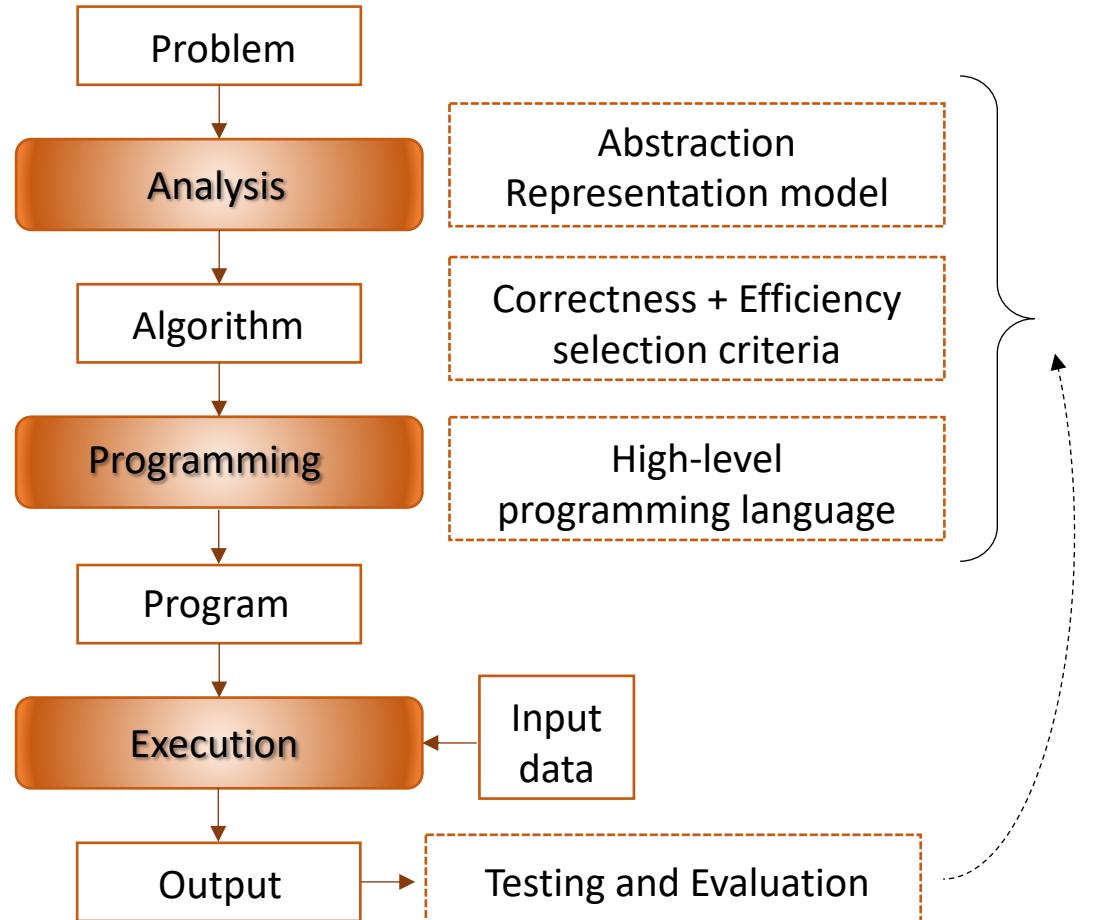
Algorithms

- ✓ A finite list of instructions that describe a **computation**
- ✓ when the instructions are executed on a provided set of inputs, the computation proceeds step by step through a set of well-defined states (configurations)
- ✓ eventually, it ends, with some outputs being produced



- Many ways to **describe** the same algorithm (including natural language)
- To be implemented, it needs to be described in an **unambiguous** form
- Can be applied to a **class of problems**

The 15-110 journey ...



- ✓ **Problem analysis:** abstraction and representation
- ✓ Concepts central to design and understand **computation** (i.e., *algorithms*)
- ✓ Fundamental concepts of **programming**:
 - *Knowledge representation* (data types & structures)
 - *Flow control* (conditionals, iteration, recursion)
 - *Data organization* (lists, matrices, dictionaries)
 - *Decomposition, reusability, information hiding* (modules, functions, objects)
- ✓ **Python programming language**
- ✓ Computational **problem-solving techniques**
- ✓ **Correctness and efficiency** of algorithms (testing, error finding, complexity, optimization)
- ✓ **Data visualization and data publishing**
- ✓ **Put all together into a data science project!**

Road map

- General overview: what this course is about
- **Utility of the course**
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

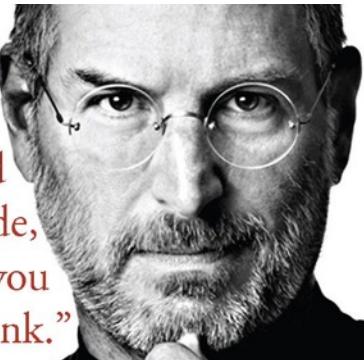
Motivations for following 15-110 ...

- Understanding computation and learning computer programming helps shaping your mind, develops your critical thinking: it's like *math* but more interactive and more fun!

Steve Jobs

1955-2011

“Everyone should
learn how to code,
it teaches you
how to think.”

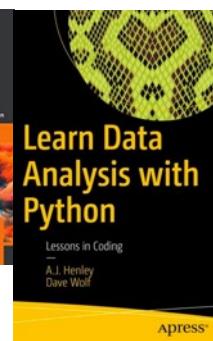
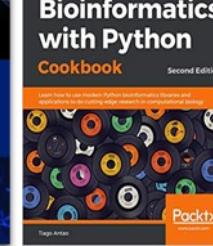
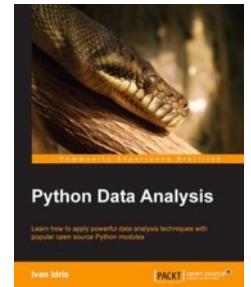
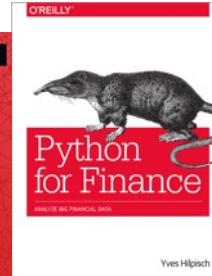
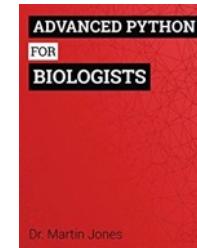
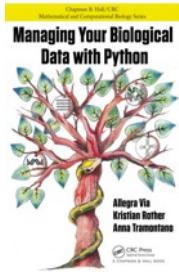
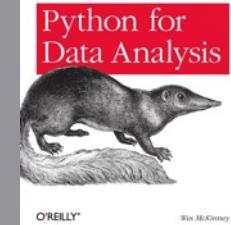
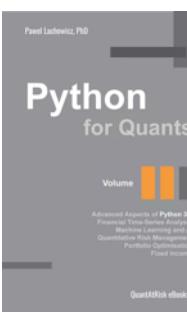
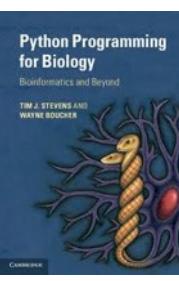
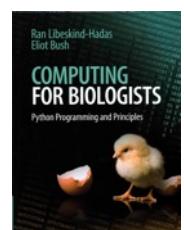
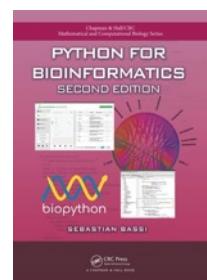
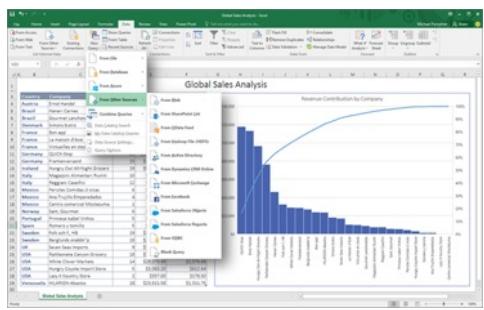
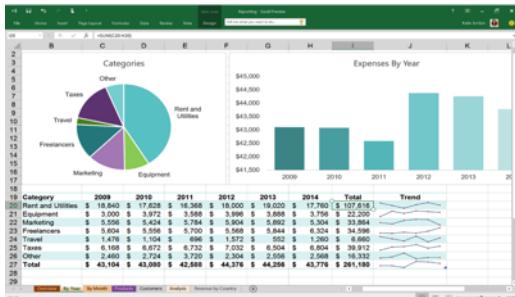


- ✓ Rational thinking for (general) problem analysis and solving
- ✓ Develop abstraction and formalization capabilities
- ✓ Devising an efficient solution to a difficult problem is always rewarding
- ✓ Finding errors in the code is as thrilling as finding the assassin! ☺

- Why understanding computation and learning computer programming can be rewarding?
 - ✓ So many (well paid) jobs around!

Motivations for following 15-110 ...

- Why understanding computation and learning computer programming can result in a professional advantage also for non-CS professionals?



Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- More about the need to use efficient algorithms + computers

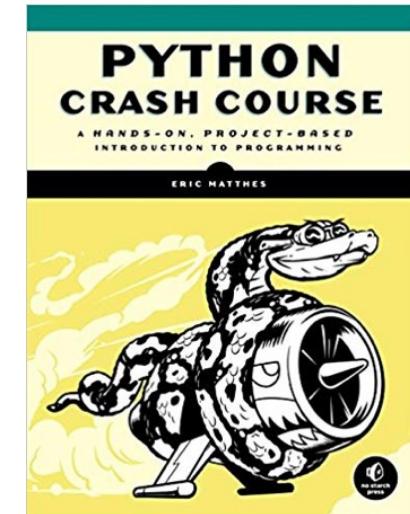
Logistics: Classes and Grading

- **Theory Classes:** Sundays, Tuesdays (2:30pm – 3:45pm), room 2163, In-presence
 - **Labs, hands-on, Quizzes:** Thursdays (2:30pm – 3:545pm), room 2163, In-presence
 - **Homework:** weekly, programming questions (autograded) + theory quizzes
 - **Quizzes:** weekly, during lab hours
 - **Midterms:** three tests during the course
 - **Final exam:** one general test at the end of course
 - **Project:** programming project issued at week 10 to be completed by the end of the course (deliver: code/notebook + oral presentation to teacher)
- **Homeworks: 15%**
 - Weekly, except on exam weeks
 - **To be solved individually** (check the course policy)
 - Check the schedule for dates and deadlines
 - **Quizzes: 25%**
 - Weekly
 - Administered at the beginning of the lecture or lab
 - Individual and closed book
 - **Three midterms: 30% (10% each)**
 - Administered during normal class hours
 - Individual and closed book
 - **Project: 10%**
 - **Final exam: 20%**

Logistics: Resources

- Main sources for material:

- Website: Lectures + lecture slides + technical notes
- Python reference book on **Canvas**
- Course website: <https://cs-cmuq.github.io/110-www/>



- Piazza for: Questions, announcements, discussions

A screenshot of a web browser showing the Piazza class page for "15-110Q". The address bar shows the URL "piazza.com/class/ksbs0bt7vsg3yo". The page has a blue header with the Piazza logo and navigation links for "15-110Q", "Q & A", "Resources", "Statistics", and "Manage Class".

- Autolab for submitting homework, labs, project

A screenshot of a web browser showing the Autolab course page for "15-110q: Principles of Computing (f21)". The address bar shows the URL "autolab.andrew.cmu.edu/courses/15110q-f21". The page has a red header with the "AUTOLAB" logo and navigation links for "Home" and "15-110q: Principles of Computing (f21)".

- Canvas for submitting Quizzes

A screenshot of a web browser showing the Canvas course page for "F21-15110-W". The address bar shows the URL "canvas.cmu.edu/courses/23267". The page has a red header with the Carnegie Mellon University logo and navigation links for "F21-15110-W" and "Modules".

Logistics: In-class rules

Be *alive* and **participate!**



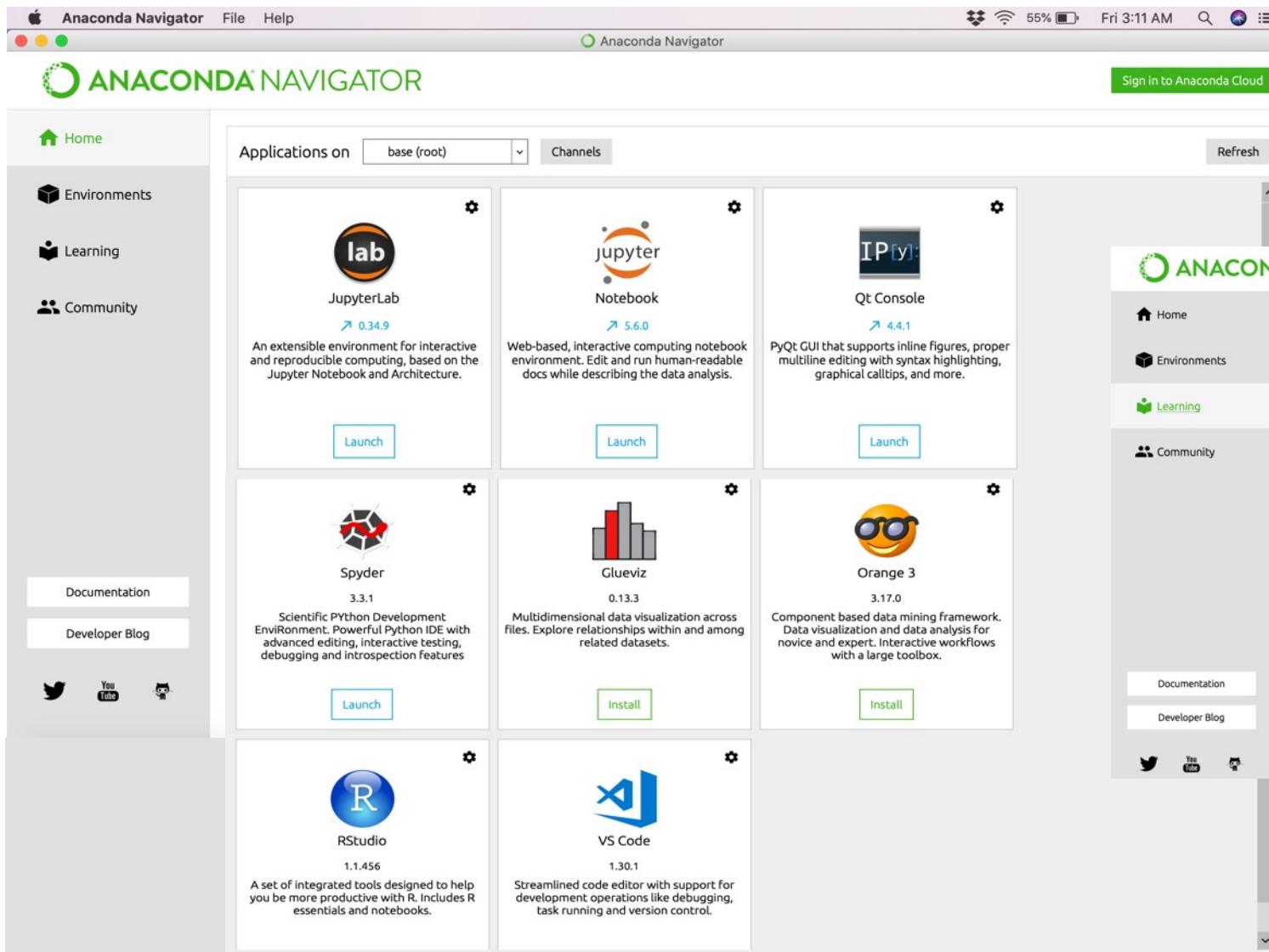
Ask questions!



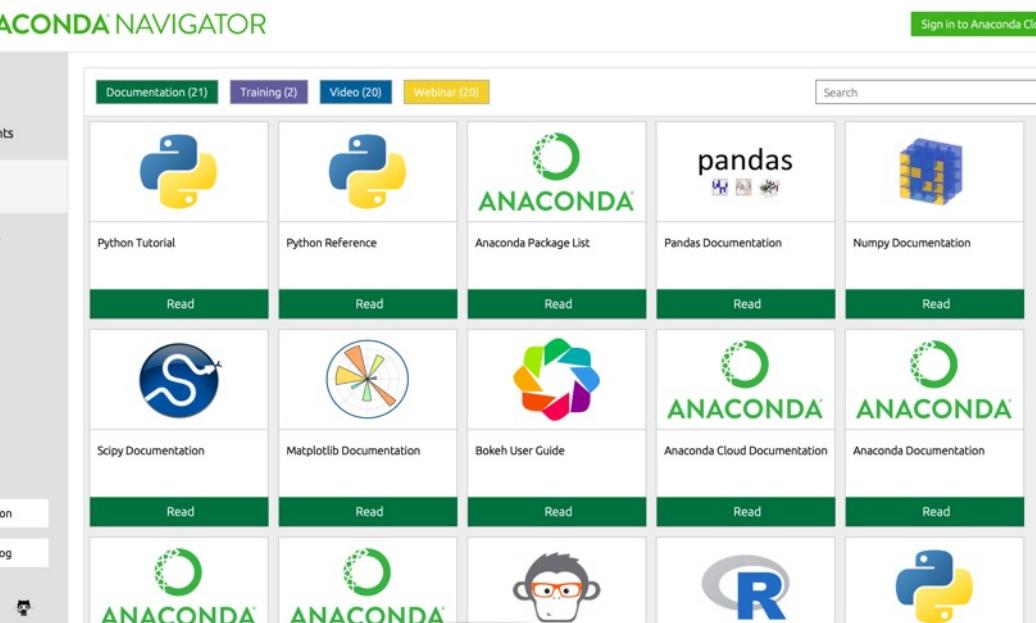
Road map

- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- **Software for python programming**

ANACONDA Navigator



<https://www.anaconda.com/>



ANACONDA: Install

anaconda.com/products/individual



ANACONDA

Products ▾

Pricing

Solutions ▾

Resources ▾

Blog

Company ▾

Anaconda Installers



Python 3.8

[64-Bit Graphical Installer \(457 MB\)](#)

[32-Bit Graphical Installer \(403 MB\)](#)



Python 3.8

[64-Bit Graphical Installer \(435 MB\)](#)

[64-Bit Command Line Installer \(428 MB\)](#)



Python 3.8

[64-Bit \(x86\) Installer \(529 MB\)](#)

[64-Bit \(Power8 and Power9\) Installer \(279 MB\)](#)

In Anaconda: Spyder, Python Integrated Development Environment (IDE)

Editor - /Users/giannidicaro/.spyder-py3/temp.py

```
temp.py
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 m = "hello class!"
8 print(m)
9
10 import numpy as np
11 import matplotlib.pyplot as plt
12
13 x1 = np.linspace(0.0, 5.0)
14 x2 = np.linspace(0.0, 2.0)
15
16 y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
17 y2 = np.cos(2 * np.pi * x2)
18
19 plt.subplot(2, 1, 1)
20 plt.plot(x1, y1, '-')
21 plt.title('A tale of 2 subplots')
22 plt.ylabel('Damped oscillation')
23
24 plt.subplot(2, 1, 2)
25 plt.plot(x2, y2, '.')
26 plt.xlabel('time (s)')
27 plt.ylabel('Undamped')
28
29 plt.show()
30
31 |
```

Variable explorer

Name	Type	Size	Value
m	str	1	hello class!
x1	float64	(50,)	[0. 0.10204082 0.20408163 ... 4.79591837 4.89795918 5. ...]
x2	float64	(50,)	[0. 0.04081633 0.08163265 ... 1.91836735 1.95918367 2. ...]
y1	float64	(50,)	[1. 0.72367065 0.2320026 ... 0.00235117 0.00597998 0.00673795 ...]
y2	float64	(50,)	[1. 0.96729486 0.8713187 ... 0.8713187 0.96729486 1. ...]

IPython console

```
In [2]: runfile('/Users/giannidicaro/.spyder-py3/temp.py', wdir='/Users/giannidicaro/.spyder-py3')
hello class!
```

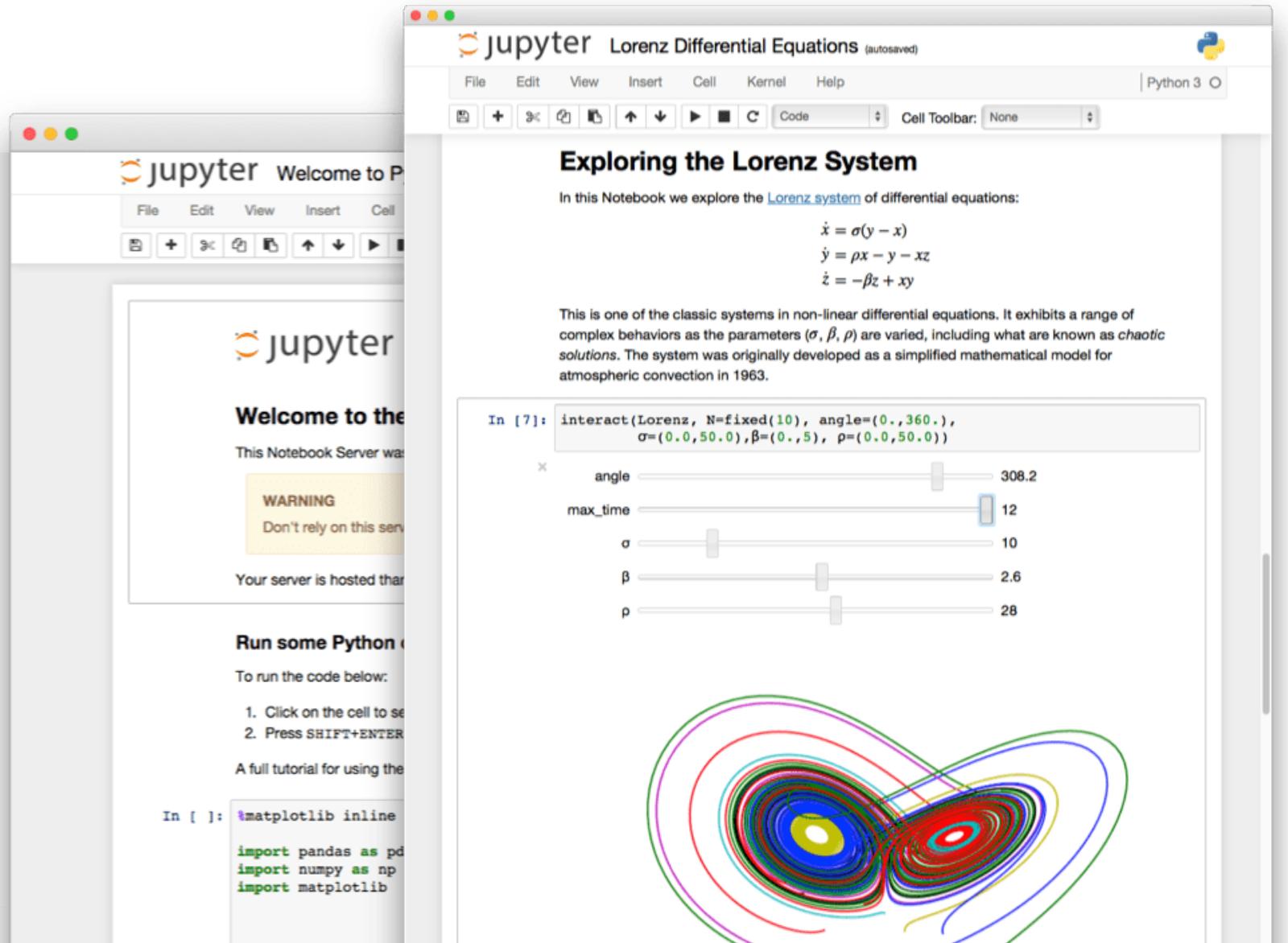
A tale of 2 subplots

time (s)

IPython console History log

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 31 Column: 1 Memory: 60 %

In Anaconda: Jupyter Notebooks



Summary

- **Overview of the topics:** a journey into computation
 - Problem solving
 - Designing and understanding algorithms
 - Fundamentals of programming constructs
 - Core techniques for computational problem-solving
 - Python as programming language
 - Tools (graphics, data manipulation, data publishing)
 - Applications
- **Action strategies:** first think, reason, abstract and model, design the algorithm, implement it by starting with a little chunk of code for a part of it, test it out, revise it, keep adding pieces of code, test with different inputs, understand what you've done!
- **The good student:** Actively participates to lectures + READs the slides + READs technical notes + READs the book chapters + Asks questions + Tries out code, tries out code, tries out code.... 😊