15-110 Principles of Computing – F23

Lecture 1:
Introduction and Logistics

Teacher:
Gianni A. Di Caro

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

The 110 team

Teacher:

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

Teacher:

Prof. Madhavi Ganapathiraju
(Comp. biology, Bioinformatics)

Course Assistants:

- Danagul Azimzhanova
- Rama Sulaiman
- Reem Kensouh
- Aboud Abdelrahman Abdalla
- Eric Jingxiang Gao
- Mohamed Mohamed
- Lorenzo Yunze Xiao

Road map

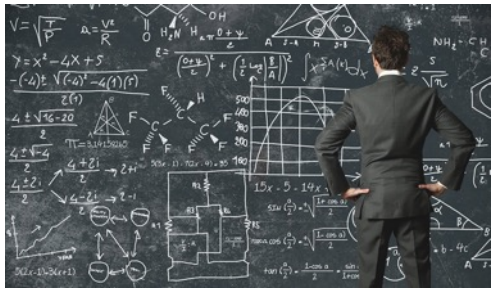
- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- A first exercise together (time permitting)

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
- A first exercise together (time permitting)

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, likes, win!)
- Different requirements for the solution (e.g., provably the best, just one, a good one)
- Different available knowledge (e.g., chess vs. financial investment vs. path finding)
- Different degree of (un)certainty (e.g., poker vs. tic-tac-toe)
- Different dimensionality / number of variables (e.g., DNA sequencing, root finding, robot 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)…

Touring through 85 cities in Europe

One not so bad solution

Optimal/Best solution

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

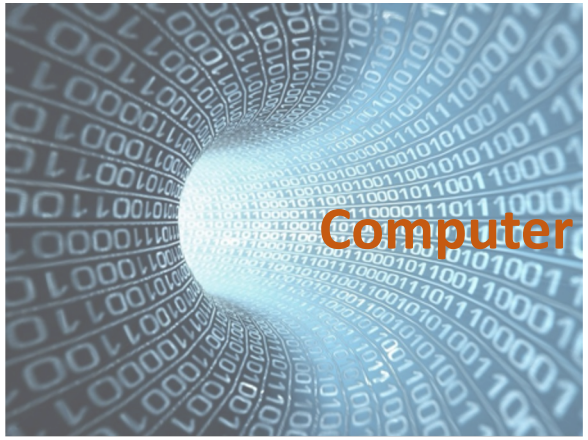
Introduction to the *science (art)* of computational problem solving

What computers are good at?

- Perform **calculations**, at impressively high rates
  - $+ \; - \; * \; / \; > \; < \; = \; \neq$
  - One the fastest computer in the world (IBM AC922): ≈ 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?

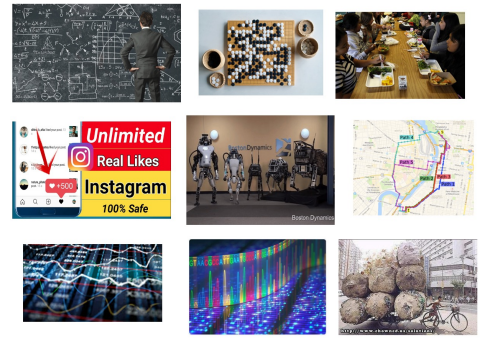
Introduction to the *science (art)* of computational problem solving



Computer Hardware

Computer programming

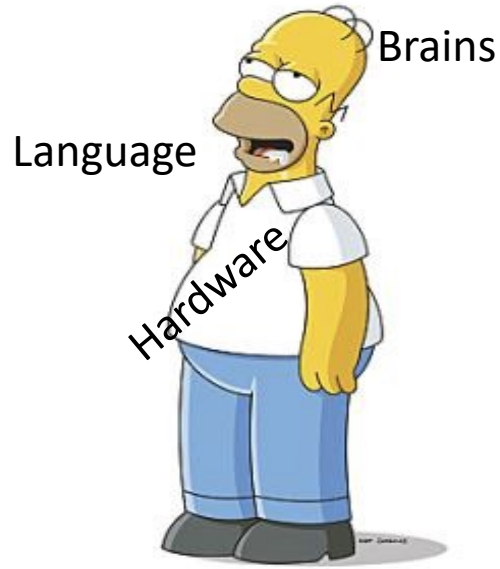
Abstraction
Model formulation



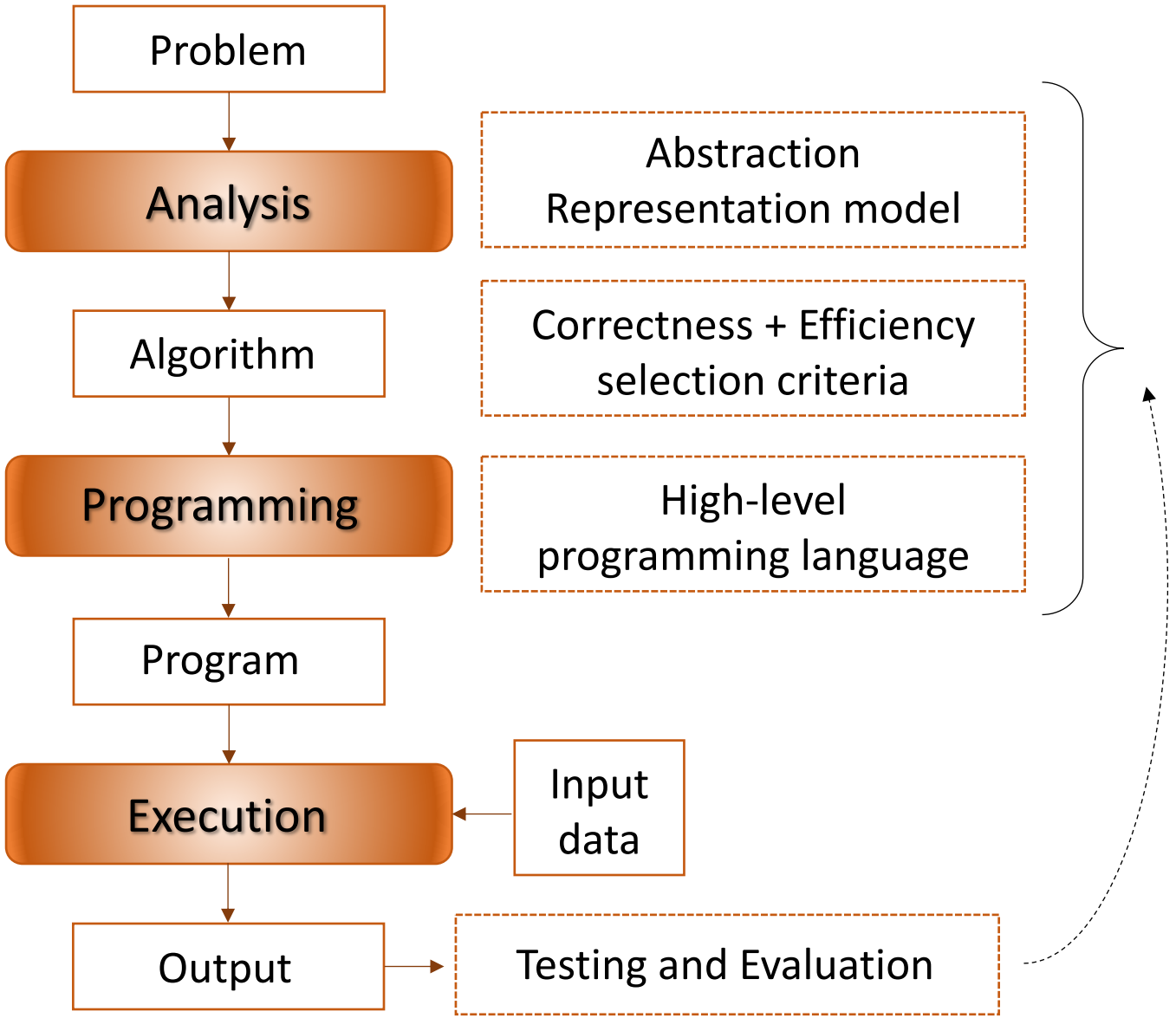
Algorithm

Correctness
Efficiency

- ✓ **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



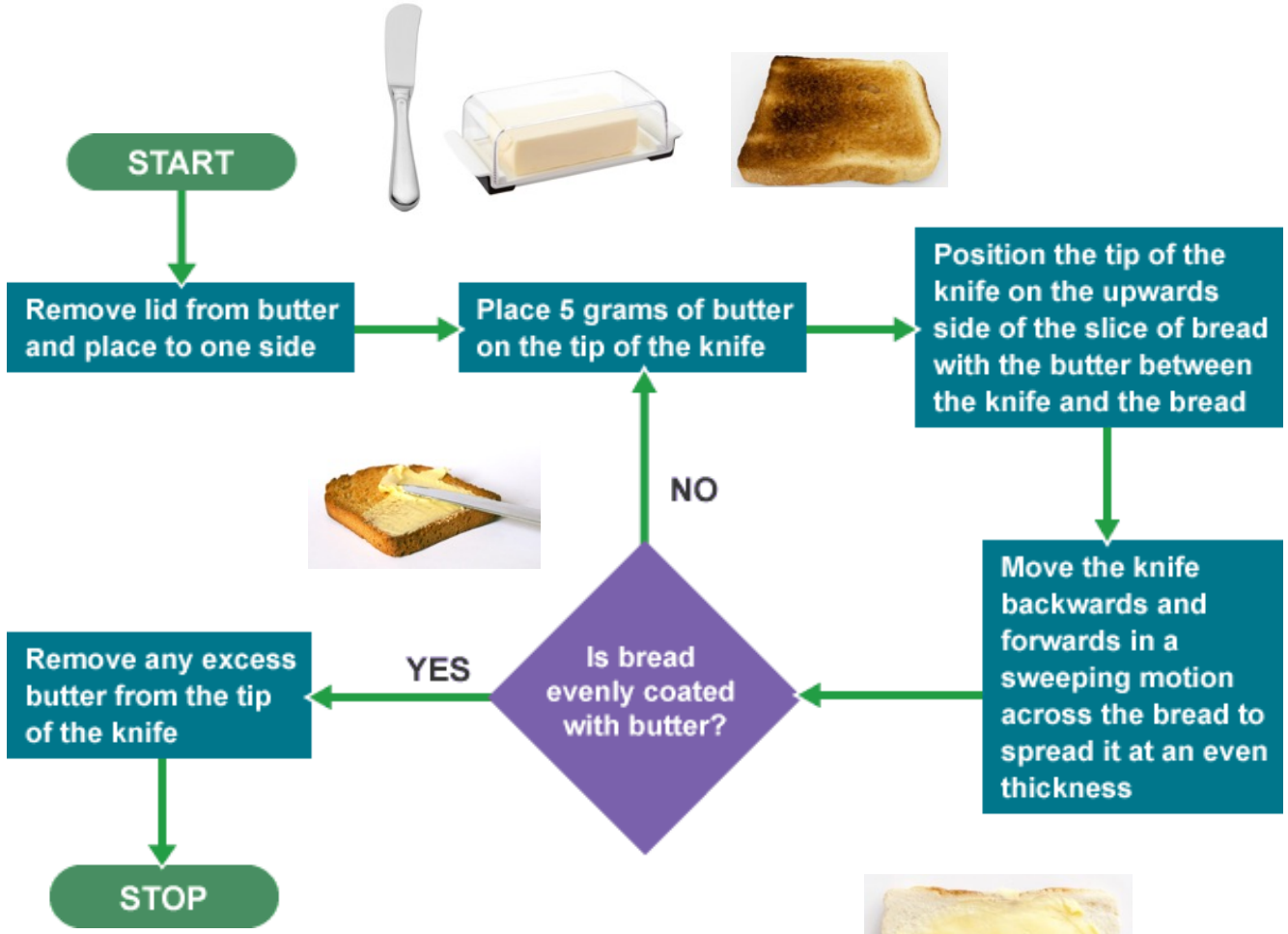
Introduction to the *science (art)* of computational problem solving



- **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

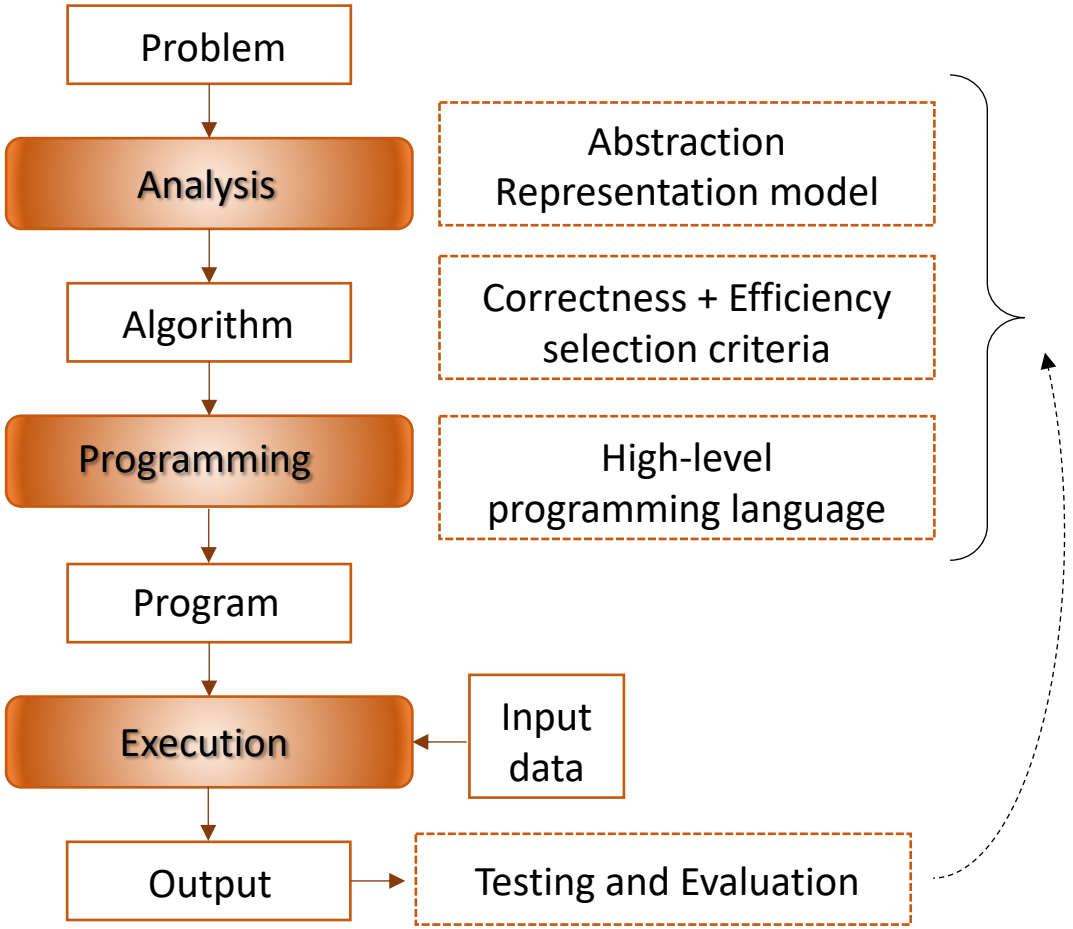
Algorithms

- ✓ 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



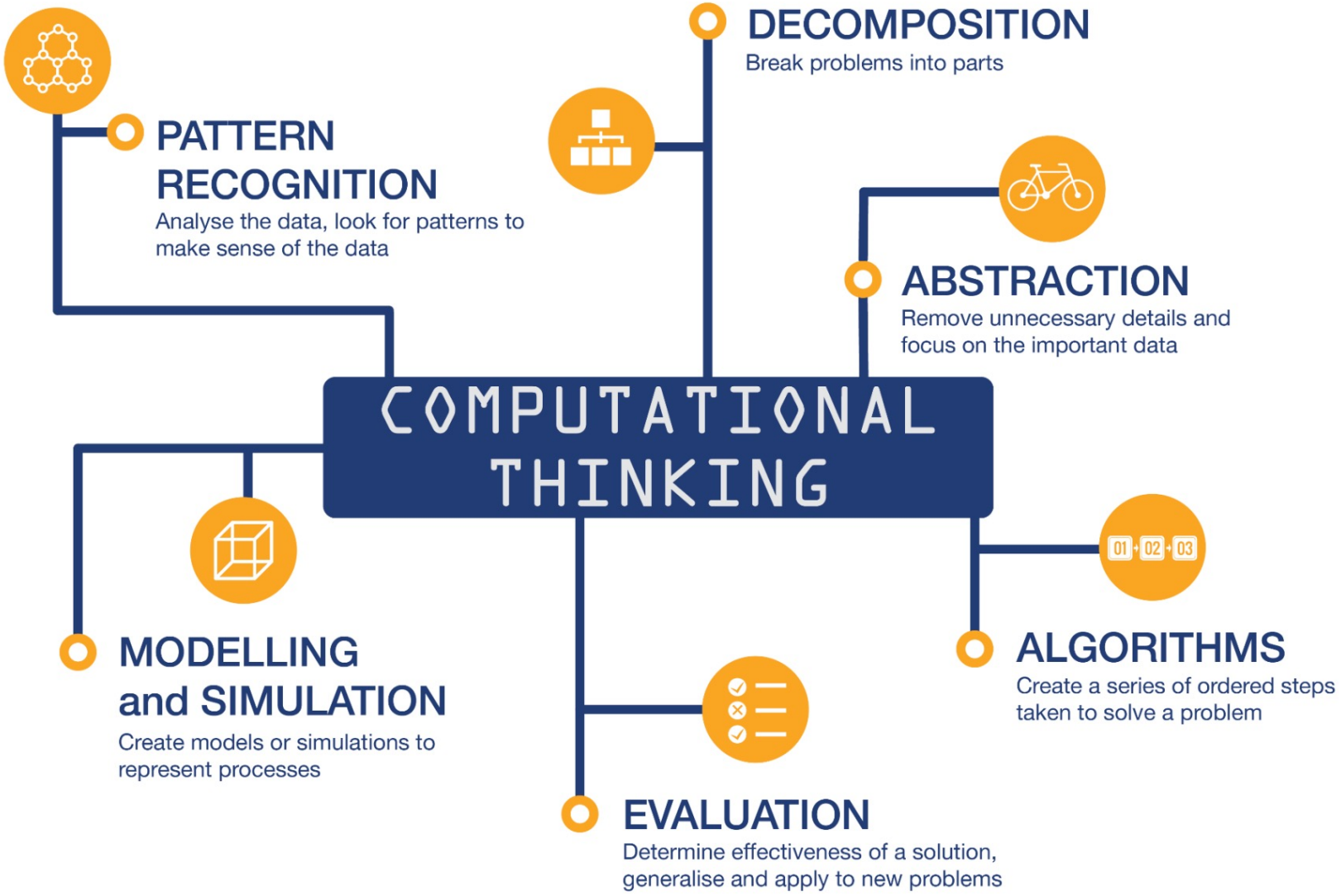
- Many ways to **describe** the same algorithm (including natural language)
- To be implemented, it needs to be described in an **unambiguous** form
- Can be applied to a **class of problems**

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!**

A first journey into Computational Thinking

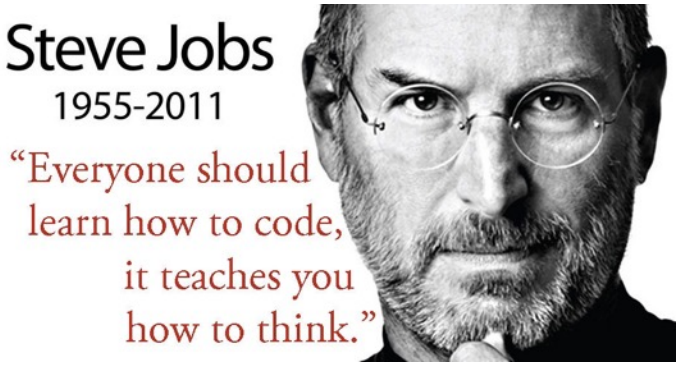


Road map

- 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
- A first exercise together (time permitting)

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!

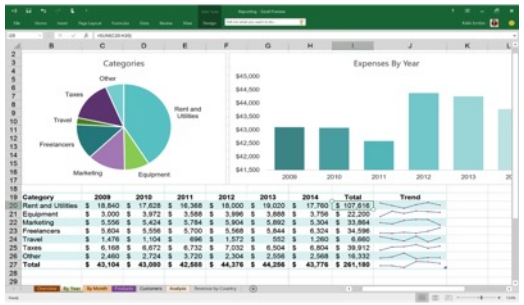


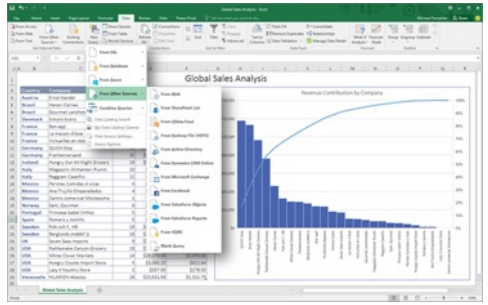
- ✓ Rational thinking for (general) problem analysis and solving
- ✓ Develop abstraction and formalization capabilities
- ✓ Devising an efficient solution to a difficult problem is always rewarding
- ✓ Finding errors in the code is as thrilling as as finding the assassin!

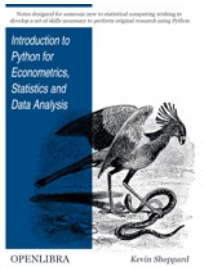
- Why understanding computation and learning computer programming can be rewarding?
  - ✓ So many (and well paid) jobs around!

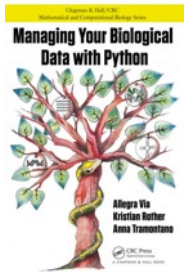
Motivations for following 15-110 …

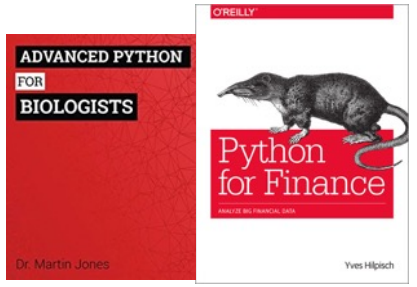
- Why understanding computation and learning computer programming can result in a professional advantage also for non-CS professionals?

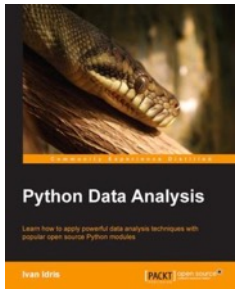


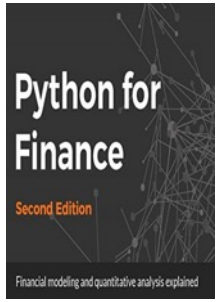


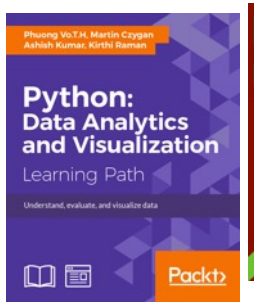


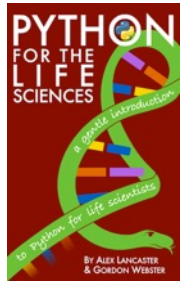


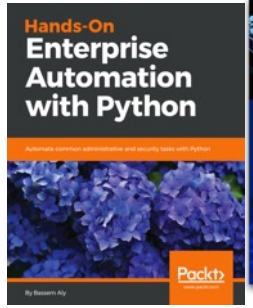


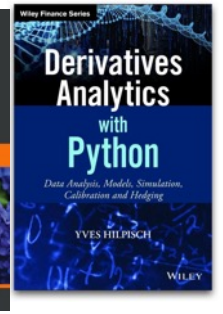


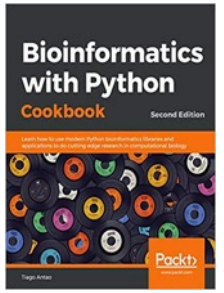


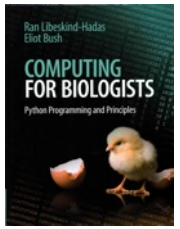


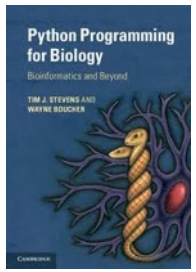


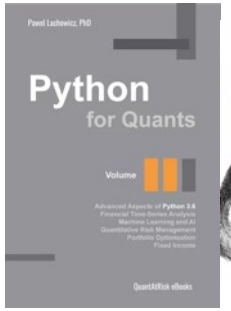


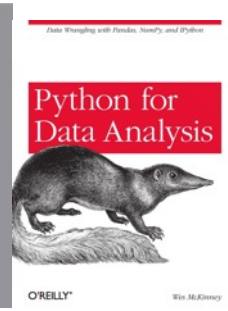


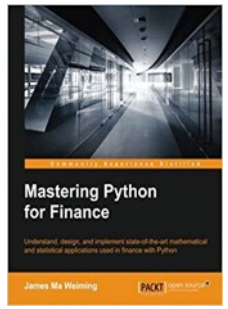


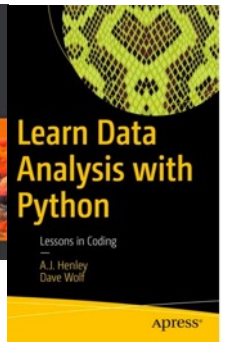












Road map

- 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
- A first exercise together (time permitting)

Logistics: Classes and Grading

- **Theory Classes**: Sundays, Tuesdays (2:30pm – 3:45pm), room 1202
- **Labs, hands-on, Quizzes**: Thursdays (2:30pm – 3:545pm), room 1202
- **Homework:** weekly, programming questions (autograded) + INTERVIEWS
- **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)

Grading

- **Homeworks + Interviews: 18%**
  - Weekly, except on exam weeks
  - Grade divided into: submitted and autograded files (8%) + interview (10%)
  - To be solved individually (check the course policy on collaboration)
  - Submission of files on Autolab due on Sundays, at 9am
  - Interviews conducted by CAs during the week of the submission
- **Quizzes: 18%**
  - Weekly, except on exam weeks
  - Administered at the beginning of the lecture or lab
  - Individual and closed book, written
  - Good practice for exams!
- **Three midterms: 30%** (10% each)
  - Administered during normal class hours
  - Individual and closed book, written
- **Project: 10%**
- **Final exam: 20%**
- **Attendance and Participation: 4%**

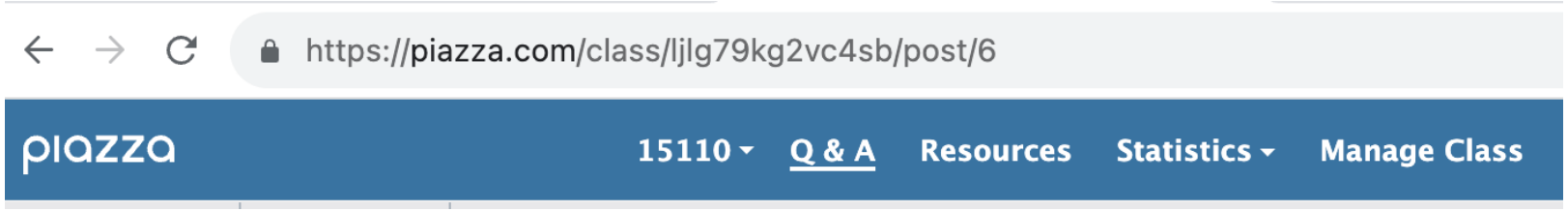
Logistics: Resources

- Course website: https://web2.qatar.cmu.edu/cs/15110/
- **Website:** Lectures + lecture slides + technical notes + problems
- Many python books in the library

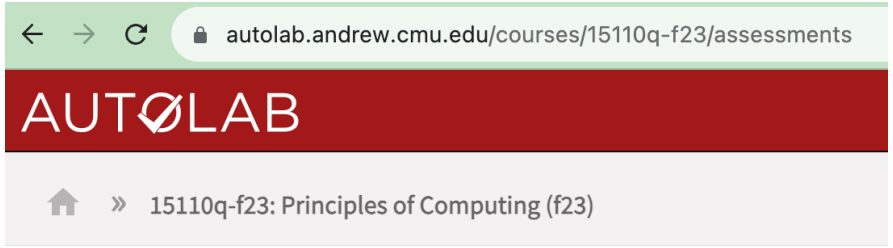
Date	Topic	Lecture notes	HW due	Lab tests	Practice
Aug 20	Introduction, fundamental concepts	[Lecture PDF], [Technical notes]			
Aug 22	Algorithms and abstraction	[Technical notes]			
Aug 24	Quiz01, Lab01: Problem solving			[Lab]	Excercises
Aug 27	Simplifying instructions and abstraction	[Technical notes]			
Aug 29	From algorithms to python	[Technical notes]			
Aug 31			hw01 due		
Aug 31	Quiz02, Lab02: IDE, Autolab			[Lab]	Excercises
Sep 03	Arithmetic	[Technical notes]			
Sep 05	Conditionals	[Technical notes]			
Sep 07			hw02 due		
Sep 07	Quiz03, Lab03: arithmetics, conditionals				
Sep 10	Break, no classes				
Sep 12	For Loops	[Technical notes]			
Sep 14	Quiz04, Lab04: for loops				
Sep 17	Midterm I				

Logistics: Platforms for course interactions

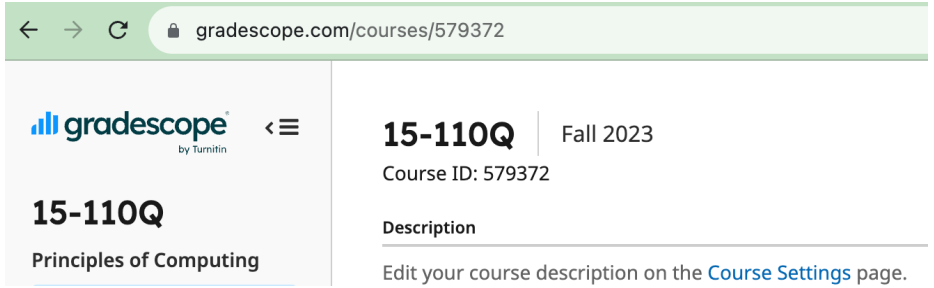
- **Piazza** for: Questions, announcements, discussions



- **Autolab** for submitting homework

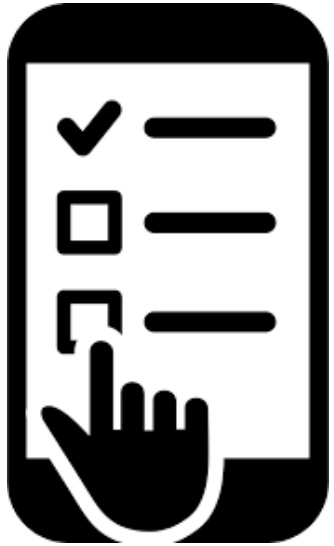


- **Gradescope** for quizzes, exams, project



Logistics: In-class rules

Be *alive* and **participate**!



Answer **polls**!

Ask **questions!**

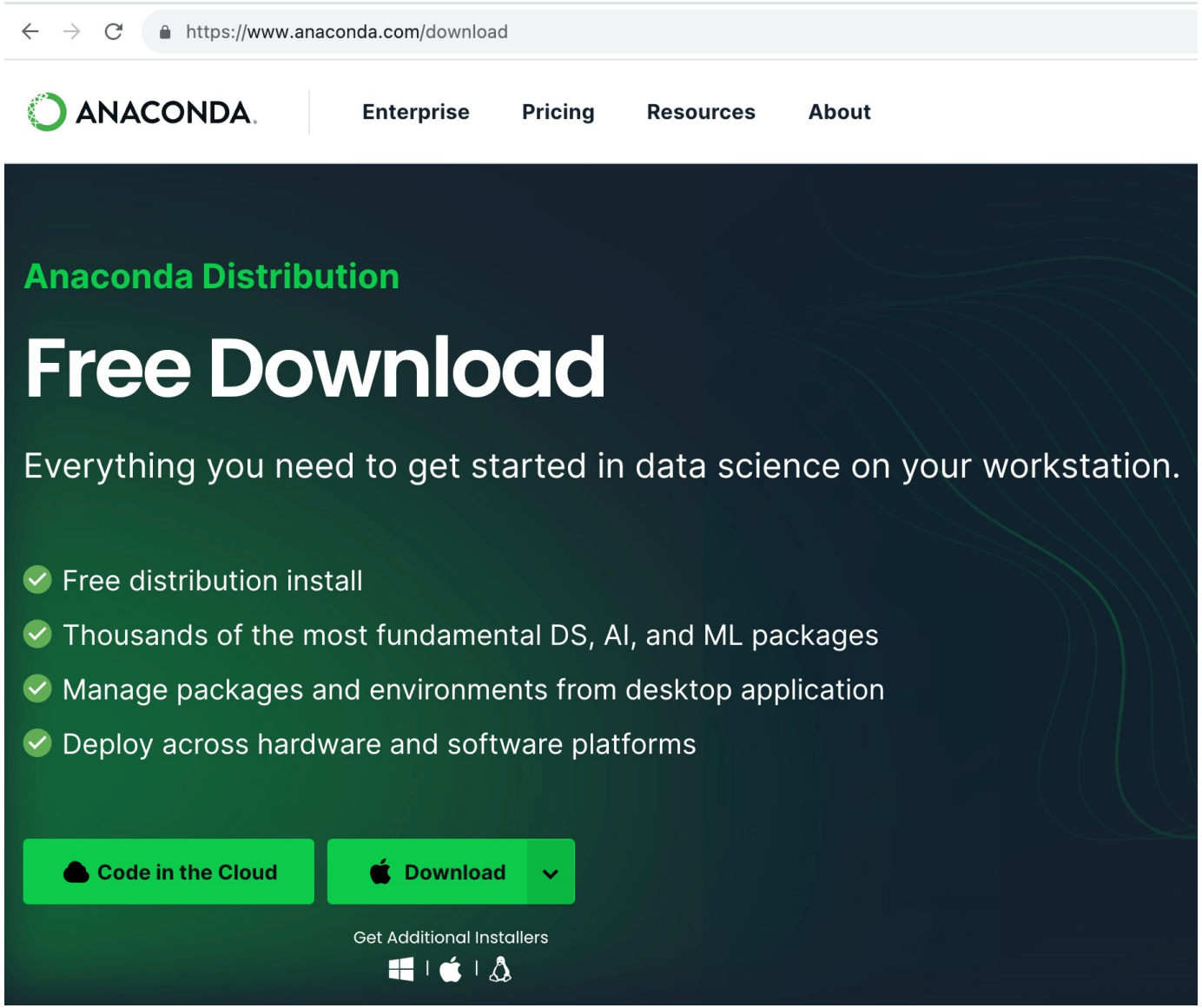
4% of your grade ….

Academic integrity: Don’t mess it up!!

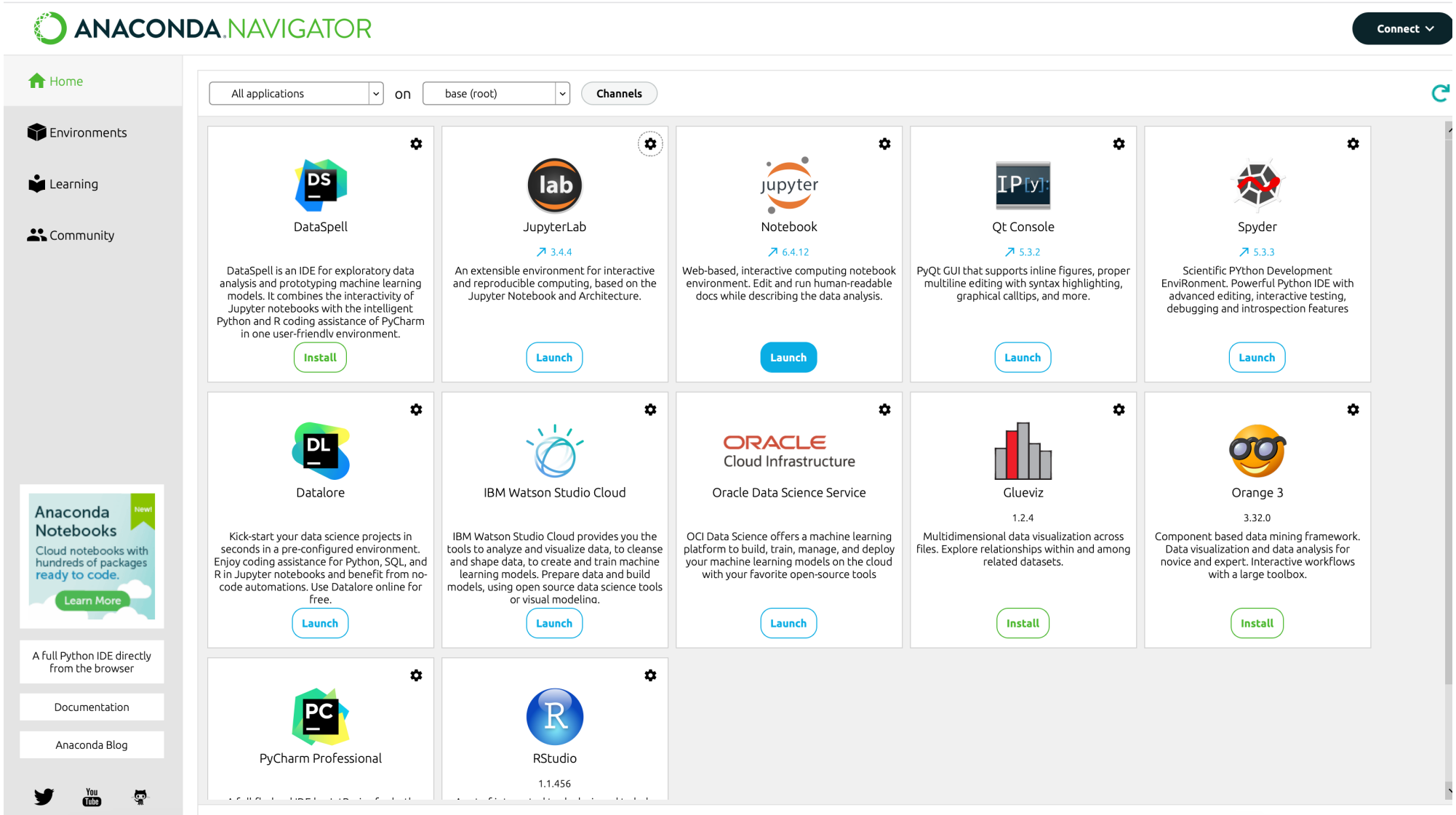
Road map

- 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
- A first exercise together (time permitting)

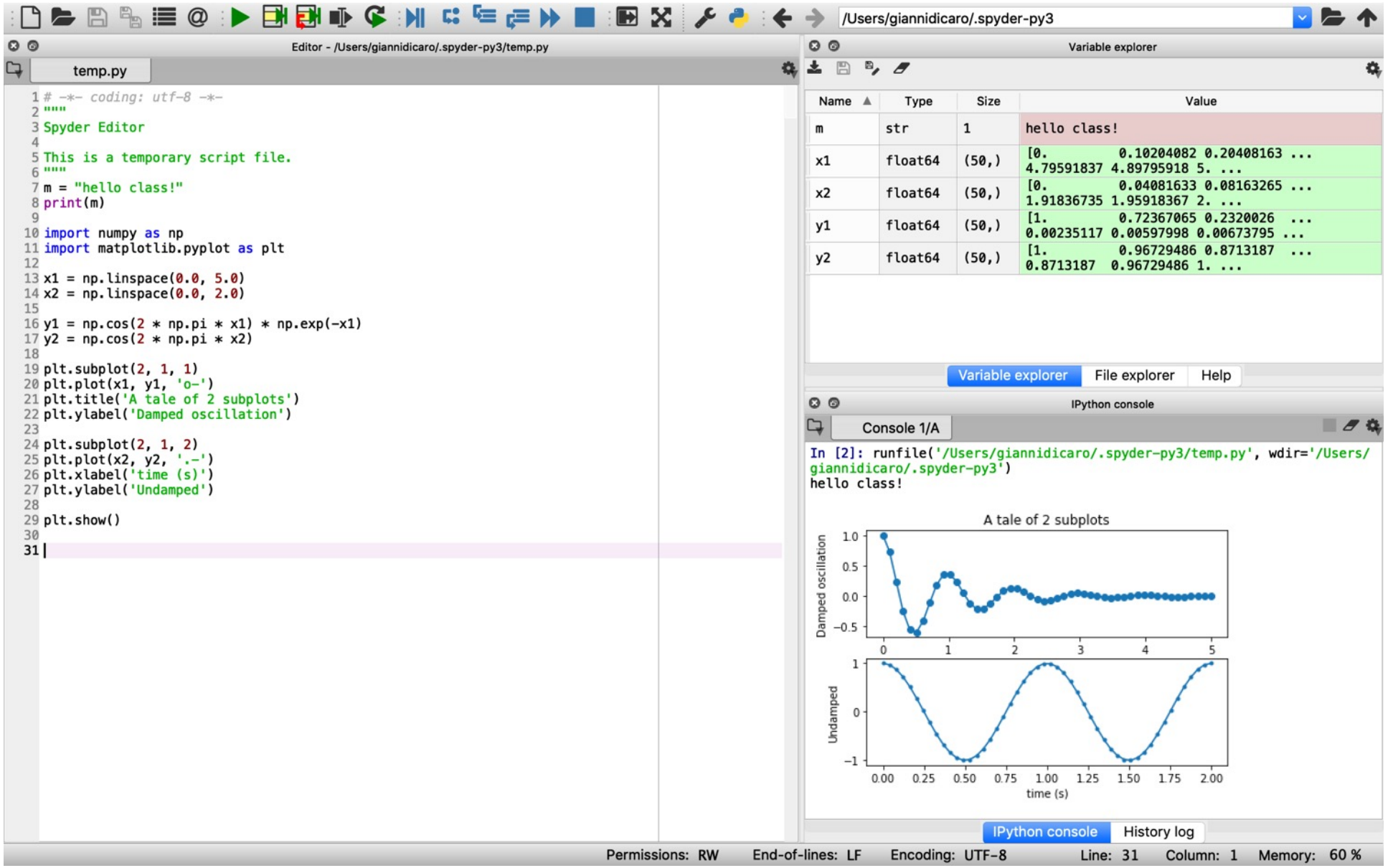
ANACONDA: Install



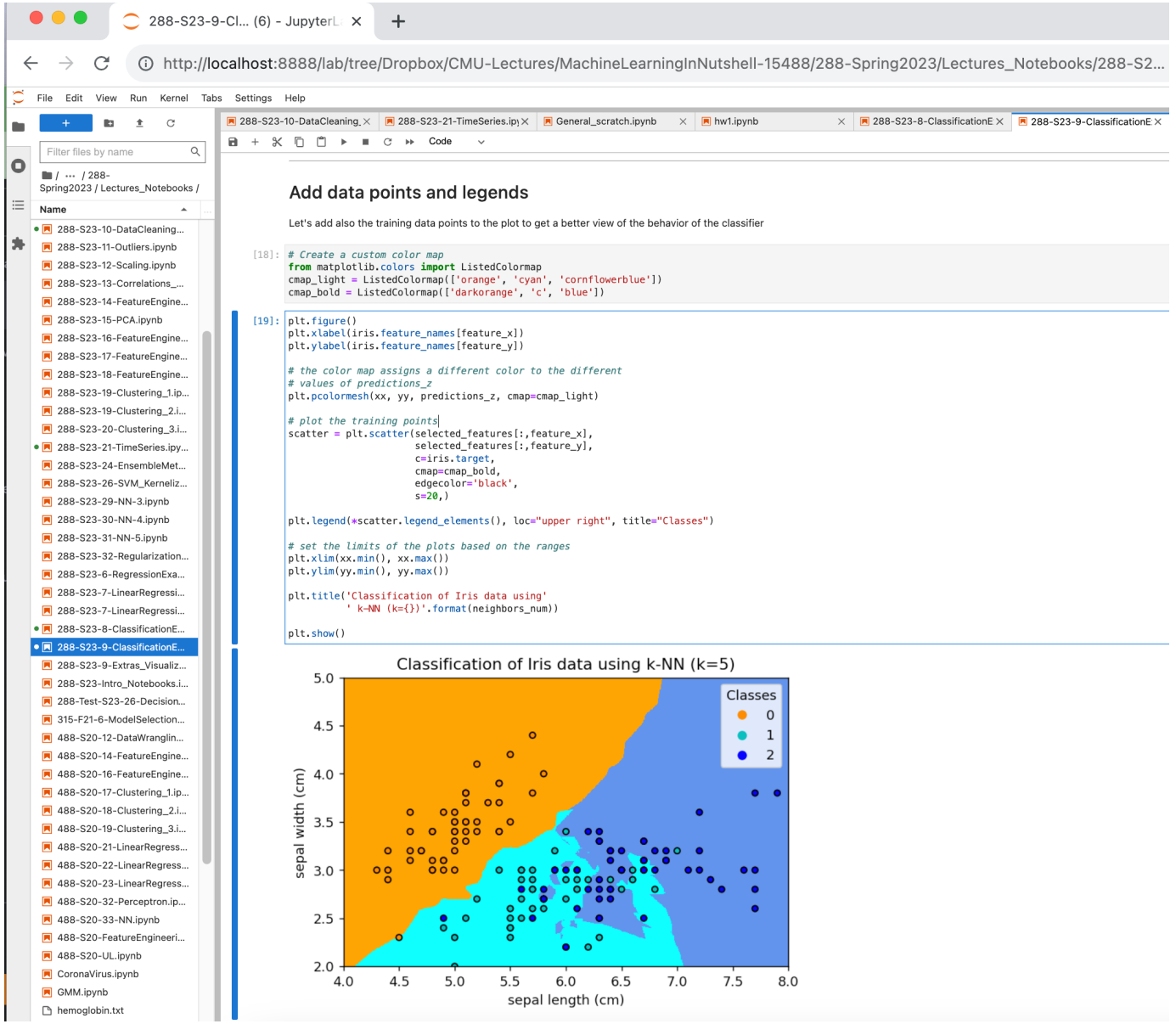
ANACONDA Navigator



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



In Anaconda: Python Notebooks in JupyterLab



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
- A first exercise together (time permitting)

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 …. ☺