15-110 Principles of Computing – F19

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)

Course Assistants:

- Andrej Pozderc
- Kaustubh Iyer
- Sara Almohanadi
- Mohammed Yusuf Ansari
- Maimoon Siddiqui
- ….

Road map

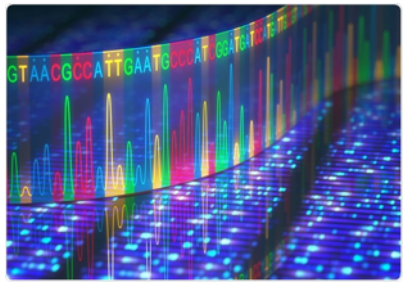
- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming

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

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
- Different requirements for the solution (e.g., provably the best, just one)
- Different available knowledge (e.g., chess, financial investment, path finding)
- Different degree of (un)certainty (e.g., poker, industrial manipulator)
- Different dimensionality (e.g., DNA sequencing, root finding, flight control)
- …

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

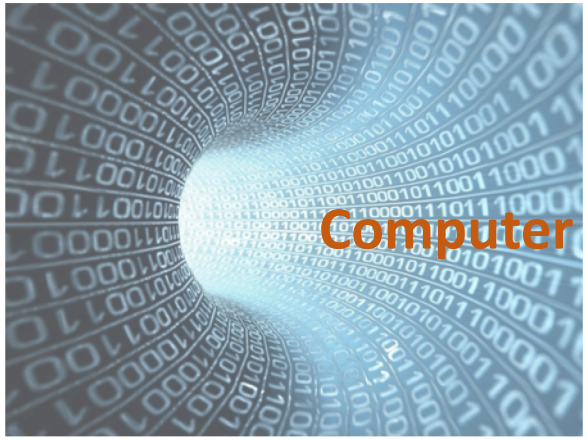
Introduction to the *science (art)* of computational problem solving

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

- Perform **calculations**, at impressively high rates
  - $+ \; - \; * \; / \; > \; < \; = \; \neq$
  - 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

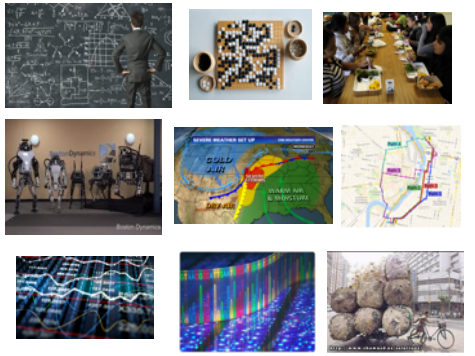


Computer Hardware

Computer programming

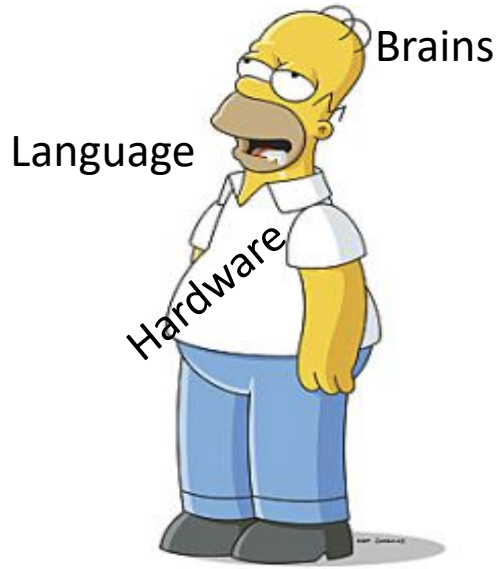
Algorithm

Abstraction
Model formulation

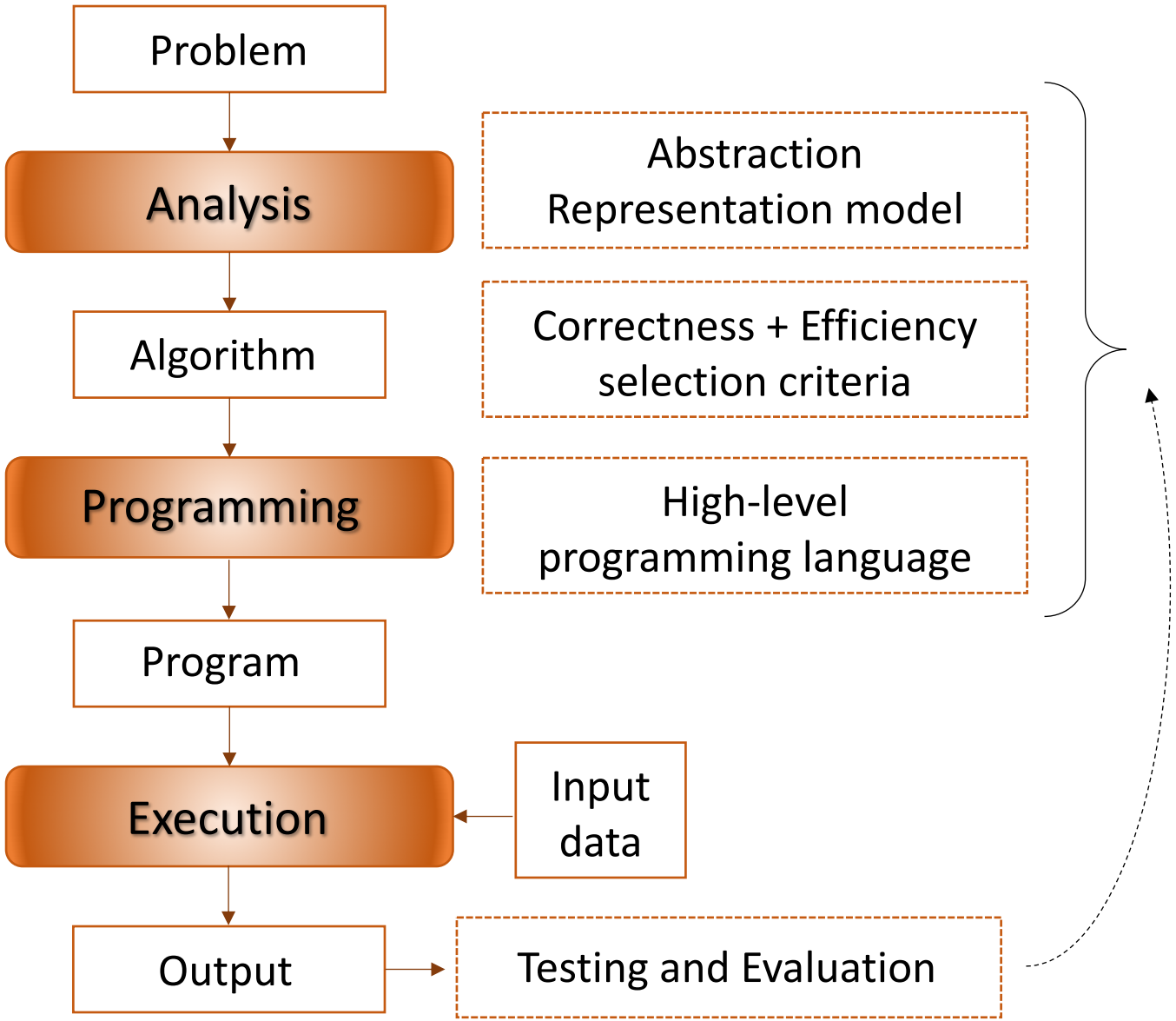


Correctness
Efficiency

- ✓ **Algorithm:** What to do (*step by step*) to 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



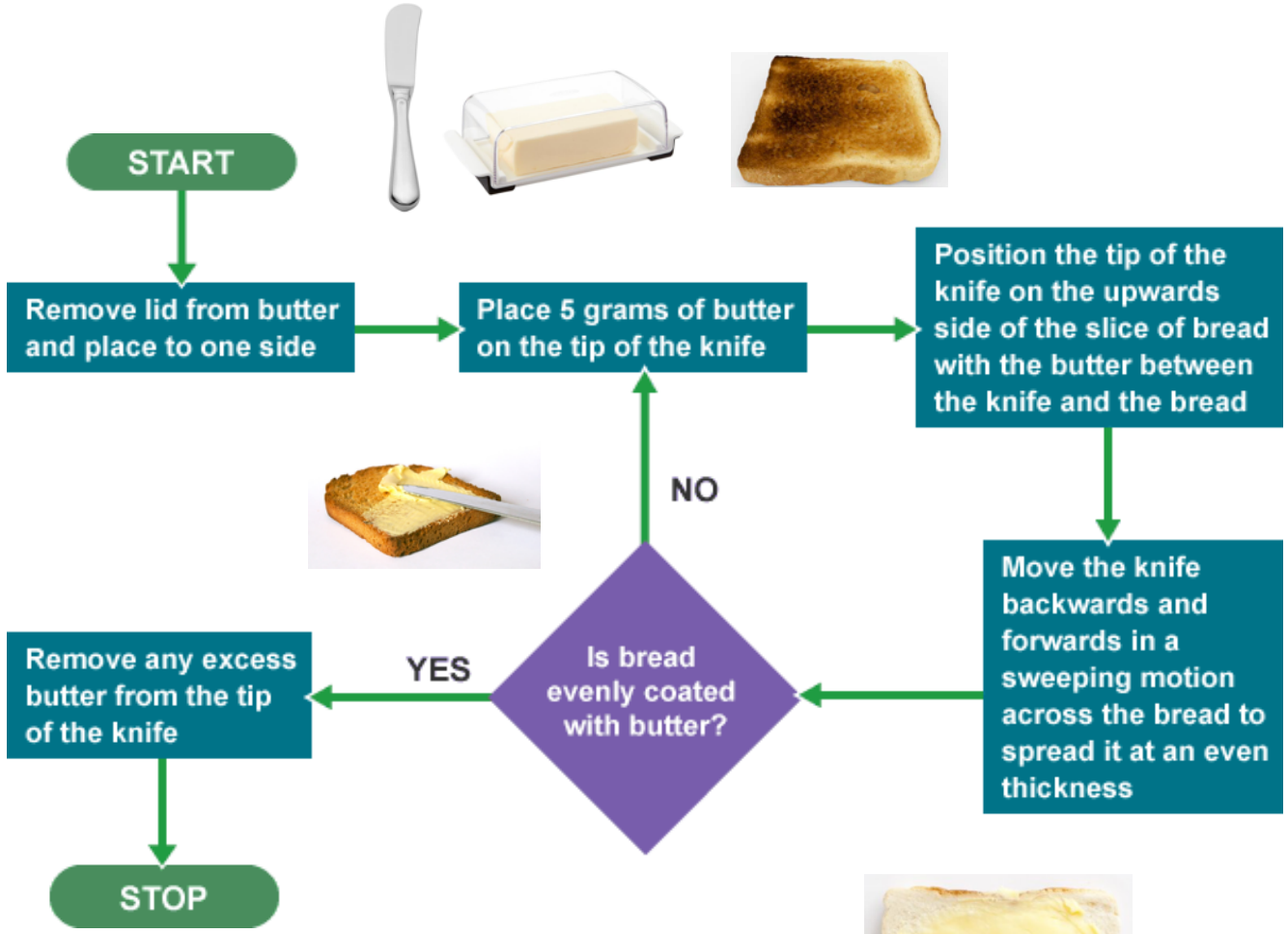
Introduction to the *science (art)* of computational problem solving



- **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

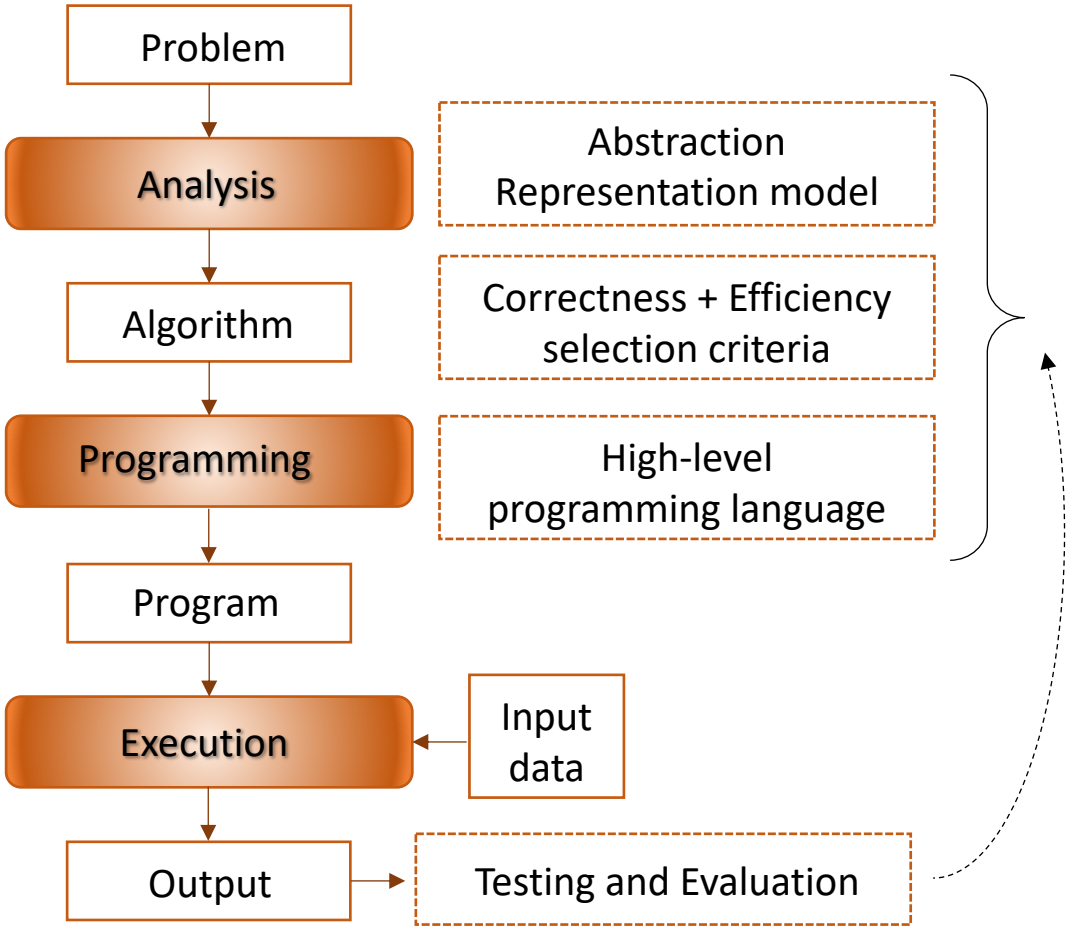
Algorithms

- ✓ 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 …



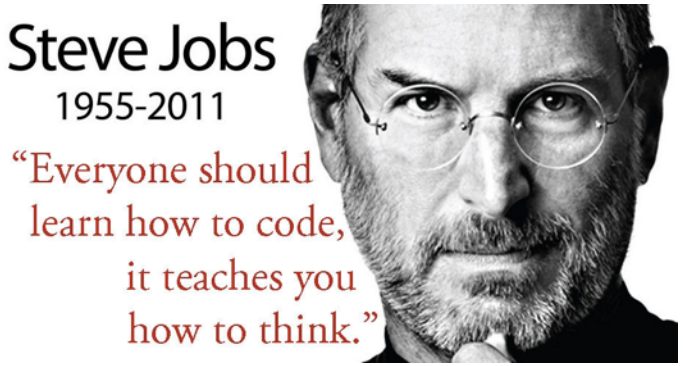
- ✓ **Problem analysis**: abstraction and representation
- ✓ Concepts central to design and understand **computation** (i.e., *algorithms*)
- ✓ Fundamental concepts of **programming**:
  - *Knowledge representation* (data types & structures)
  - *Flow control* (conditionals, iteration, recursion)
  - *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 **statistical analysis**
- ✓ **Randomness** and **simulation** (*Monte Carlo)*

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

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?

- Financial analysts: **$84,300**
- Market research analysts: **$63,230**
- Sales managers: **$121,060**
- Human resource specialists: **$60,350**
- Accountants and auditors: **$69,350**
- Management analysts: **$82,450**
- Personal financial advisors: **$90,640**

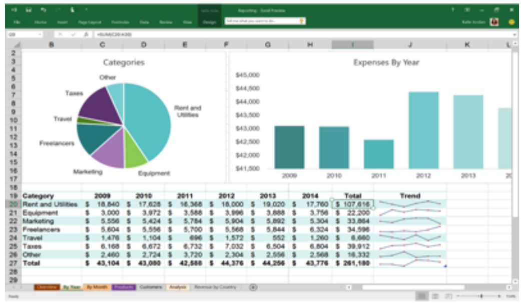
- **Software Engineer:** Average Salary $76,194
- **Senior Software Engineer:** Average Salary $89,599
- **Software Developer:** Average Salary $62,803
- **Senior Software Developer or Programmer:** Average Salary $100,303
- **Web Developer:** Average Salary $62,500
- **Programmer Analyst:** Average Salary $55,250

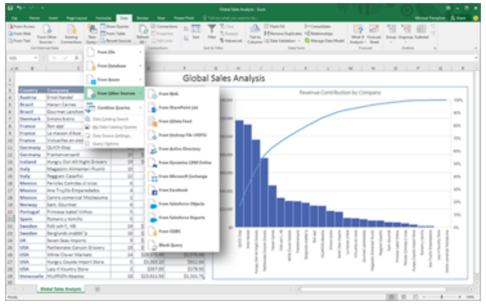
Salary range for a **Biologist**

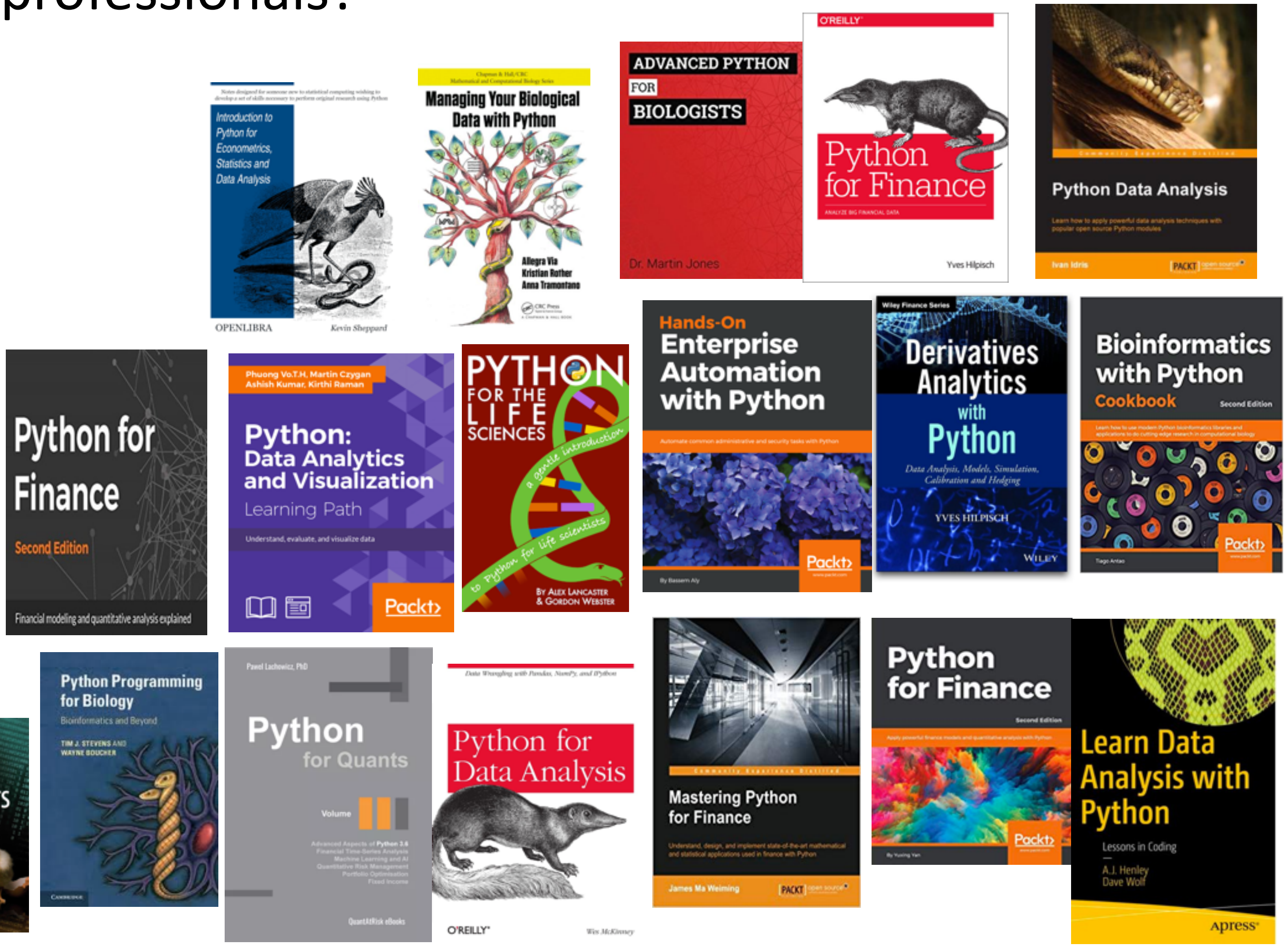
$44,946 to $56,017

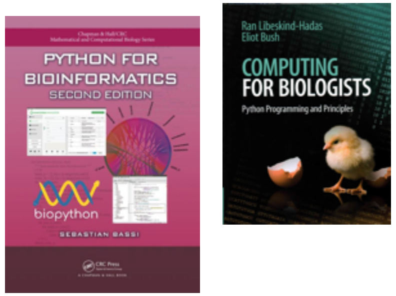
Motivations for following 15-110 …

- 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

Logistics: Classes and Grading

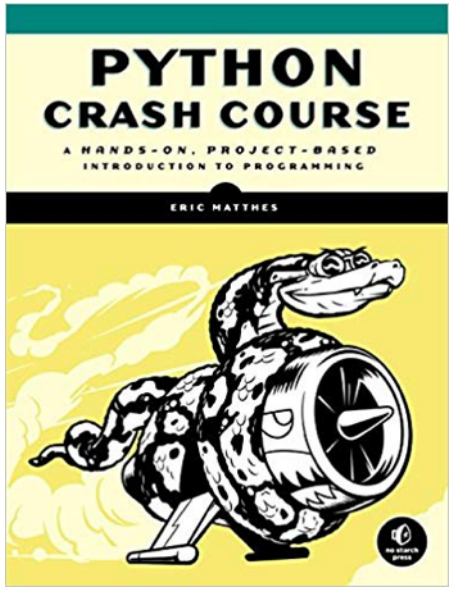
- **Theory Classes**: Sundays, Tuesdays (3:00pm – 4:20pm), room 2035
- **Lab, hands-on**: Thursdays (3:00pm – 4:20pm), room 2035
- **Homework:** weekly, programming questions (autograded) + theory quizzes
- **Partial assessment tests:** three tests during the course
- **General assessment test:** one general test at the end of the course
- **Project:** programming project issued at week 10 to be completed by the end of the course (deliver: code + in-class presentation or, most likely, video presentation)

Grading

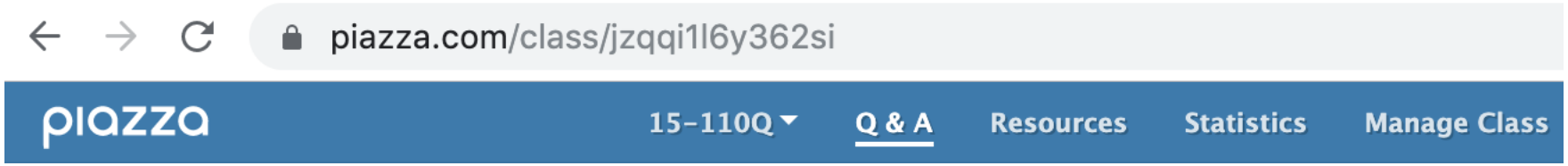
- **Homework: 30%**
- **Midterm I: 10%**
- **Midterm II: 10%**
- **Midterm III: 10%**
- **Project: 20%**
- **Final exam: 20%**

Logistics: Resources

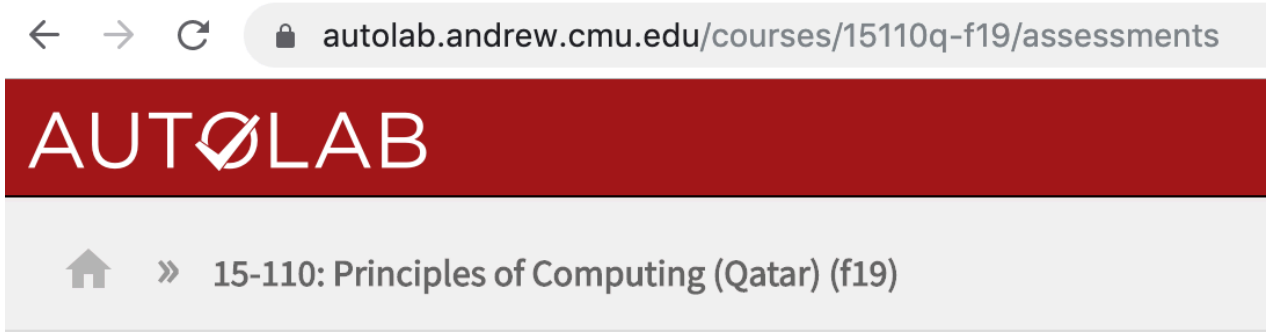
- **Main sources for material:**
  - Lectures + lecture slides
  - Python reference book
  - Course website: https://web2.qatar.cmu.edu/cs/15110/



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



- **Autolab** for submitting homework and project, and getting grades



Logistics: In-class rules

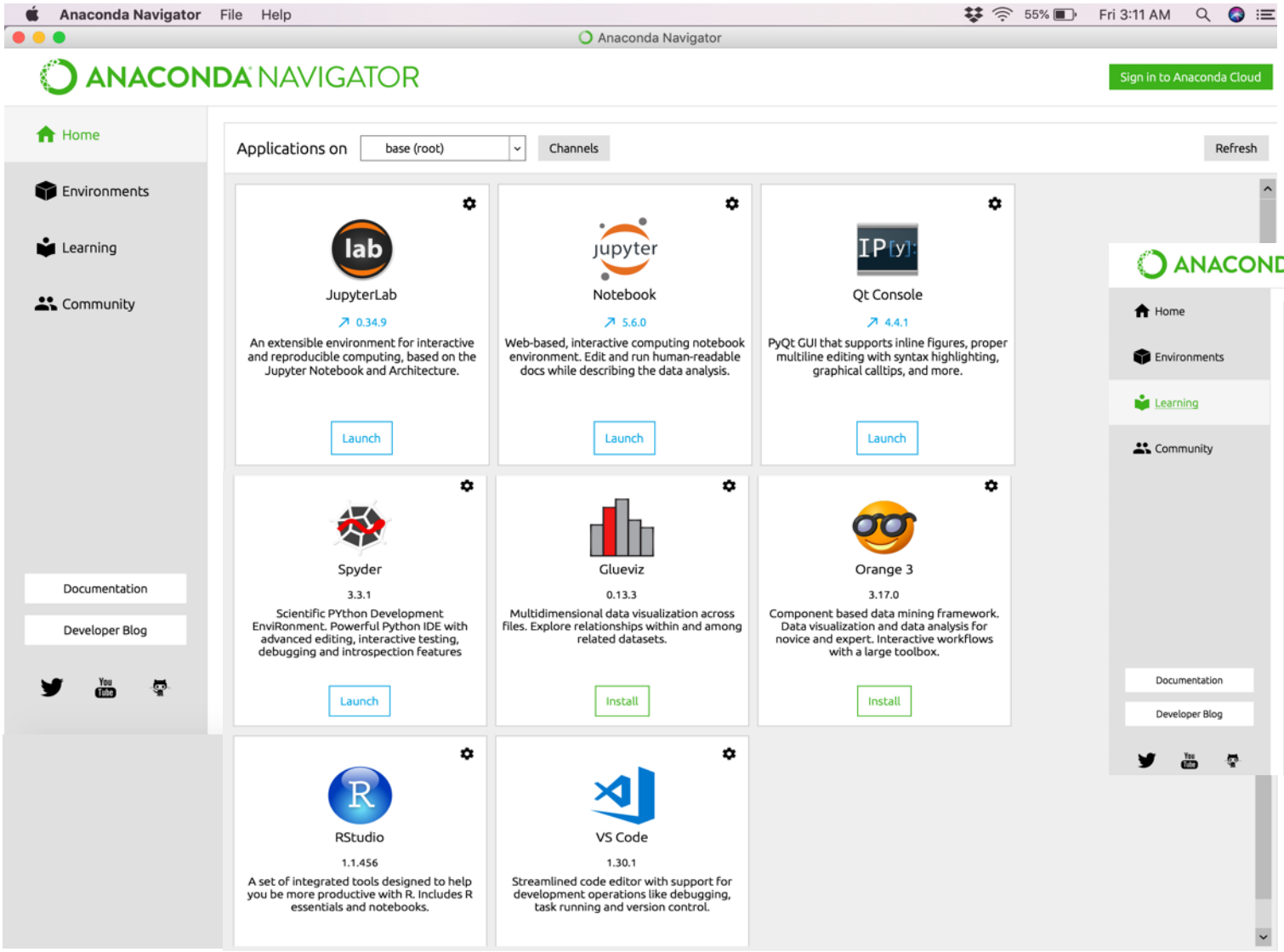
Be *alive* and **participate**!

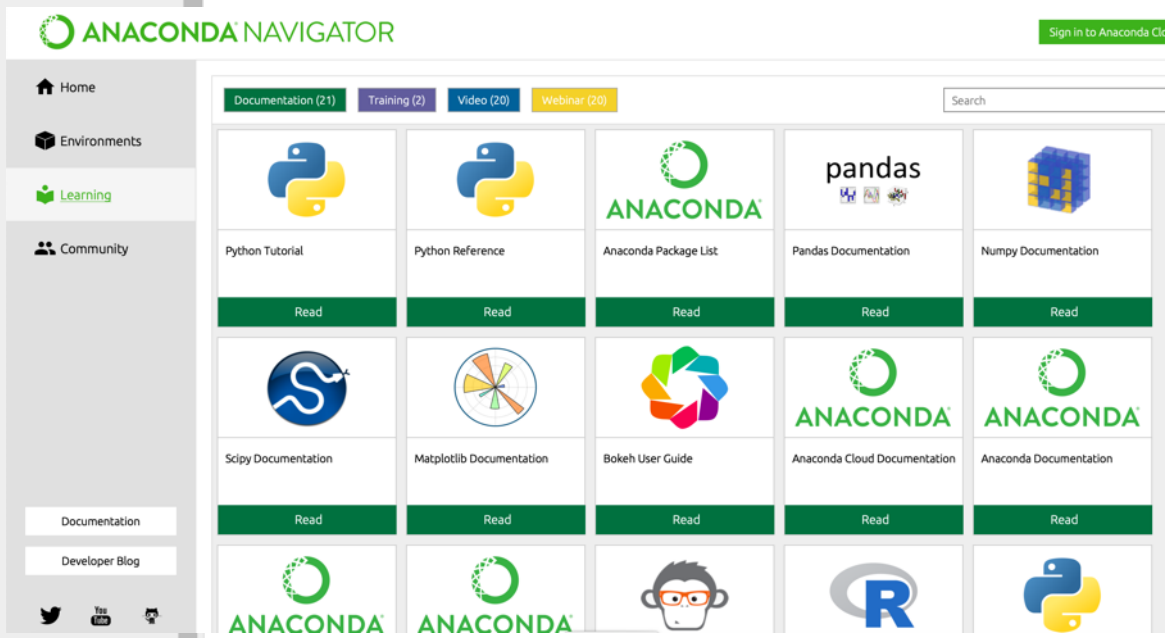
Ask **questions!**

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

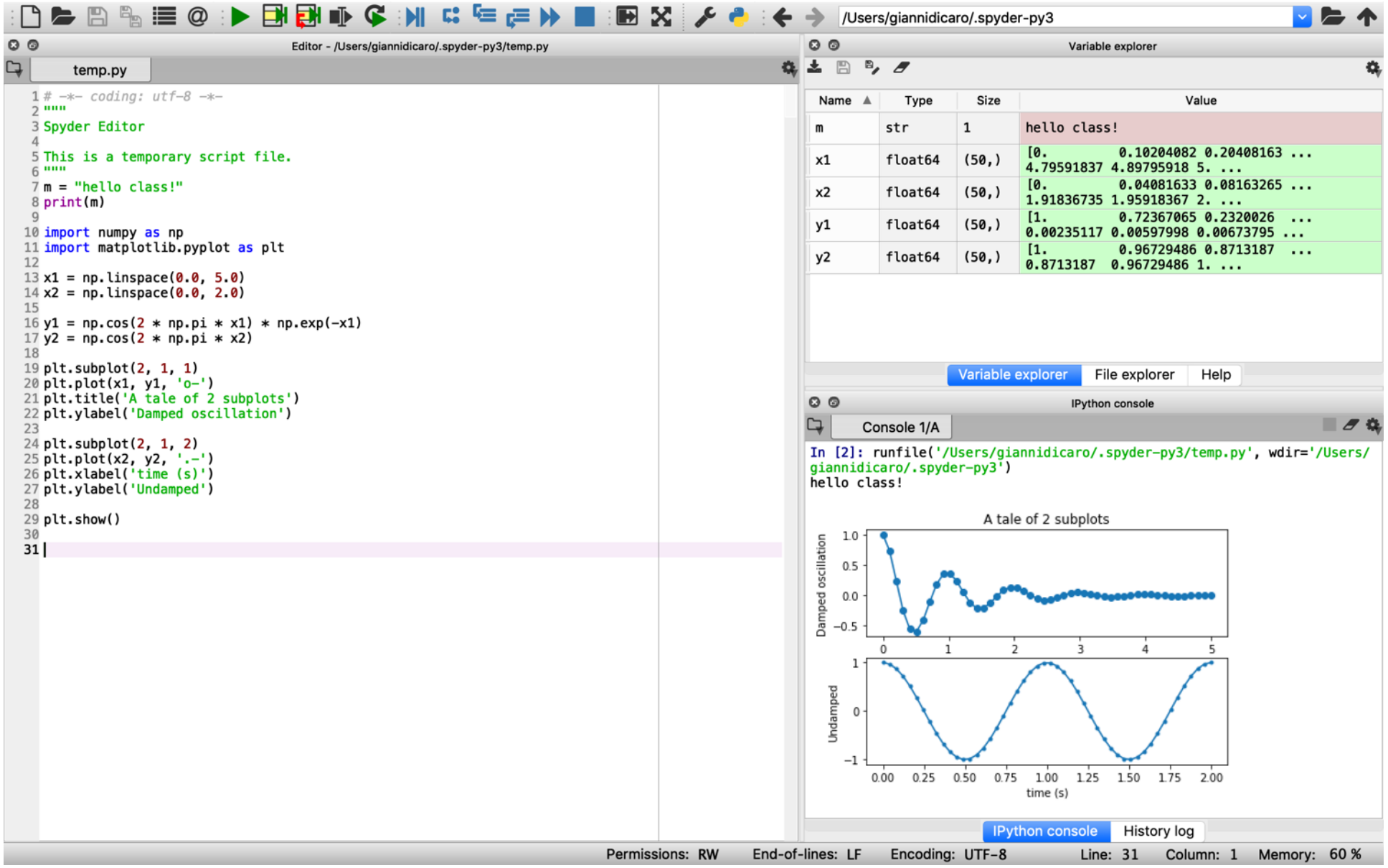
ANACONDA Navigator





https://www.anaconda.com/

Spyder: Python Integrated Development Environment (IDE)



Summary

- Overview of the topics: a journey into computation
  - Problem solving
  - Designing and understanding algorithms
  - Fundamentals programming constructs
  - Core techniques for computational problem-solving
  - Python as programming language
  - Tools (graphics, simulation, data manipulation)
  - Applications
- Action strategies: first think, reason, abstract and model, implement a little chunk of code, test it out, revise it, keep adding pieces of code, test with different inputs, understand what you've done!
- The good student: READs the slides (main material) + READs the book chapters + Tries out code, tries out code, tries out code …. ☺
- We have set the basics for the course, next we will start with Python!