



15-110 PRINCIPLES OF COMPUTING – F23

LECTURE 1: INTRODUCTION AND LOGISTICS

TEACHER:
GIANNI A. DI CARO

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

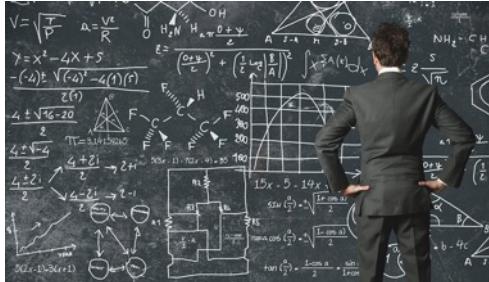
- General overview: what this course is about
- Utility of the course
- Logistics: classes, labs, homework, exams, grading, website, books, piazza, rules
- Software for python programming
- A first exercise together (time permitting)

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

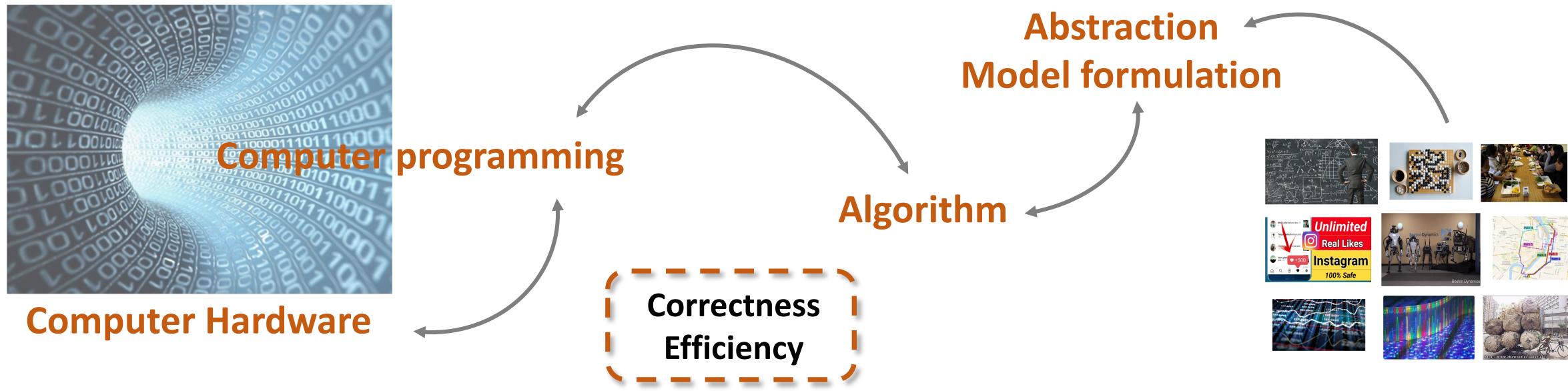
What computers are good at?



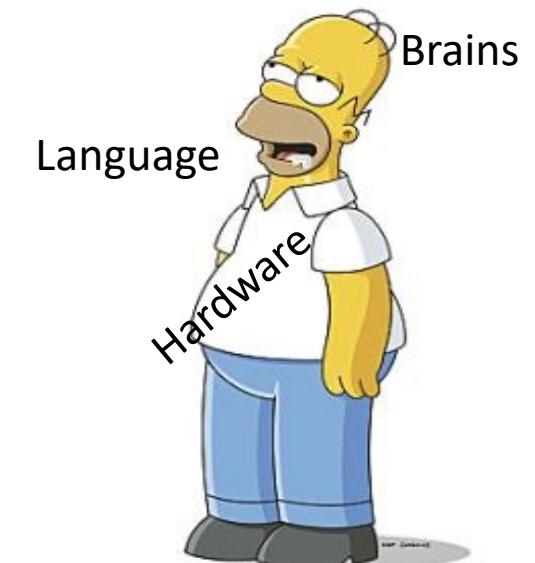
- Perform **calculations**, at impressively high rates
 - + - * / > < = ≠
 - One the fastest computer in the world (IBM AC922): $\approx 10^{15}$ operations per sec on real numbers:
1,000,000,000,000,000 ops/sec
 - Mega = 10^6 , Giga = 10^9 , Tera = 10^{12} , Peta = 10^{15} , Exa = 10^{18}
- **Store 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?

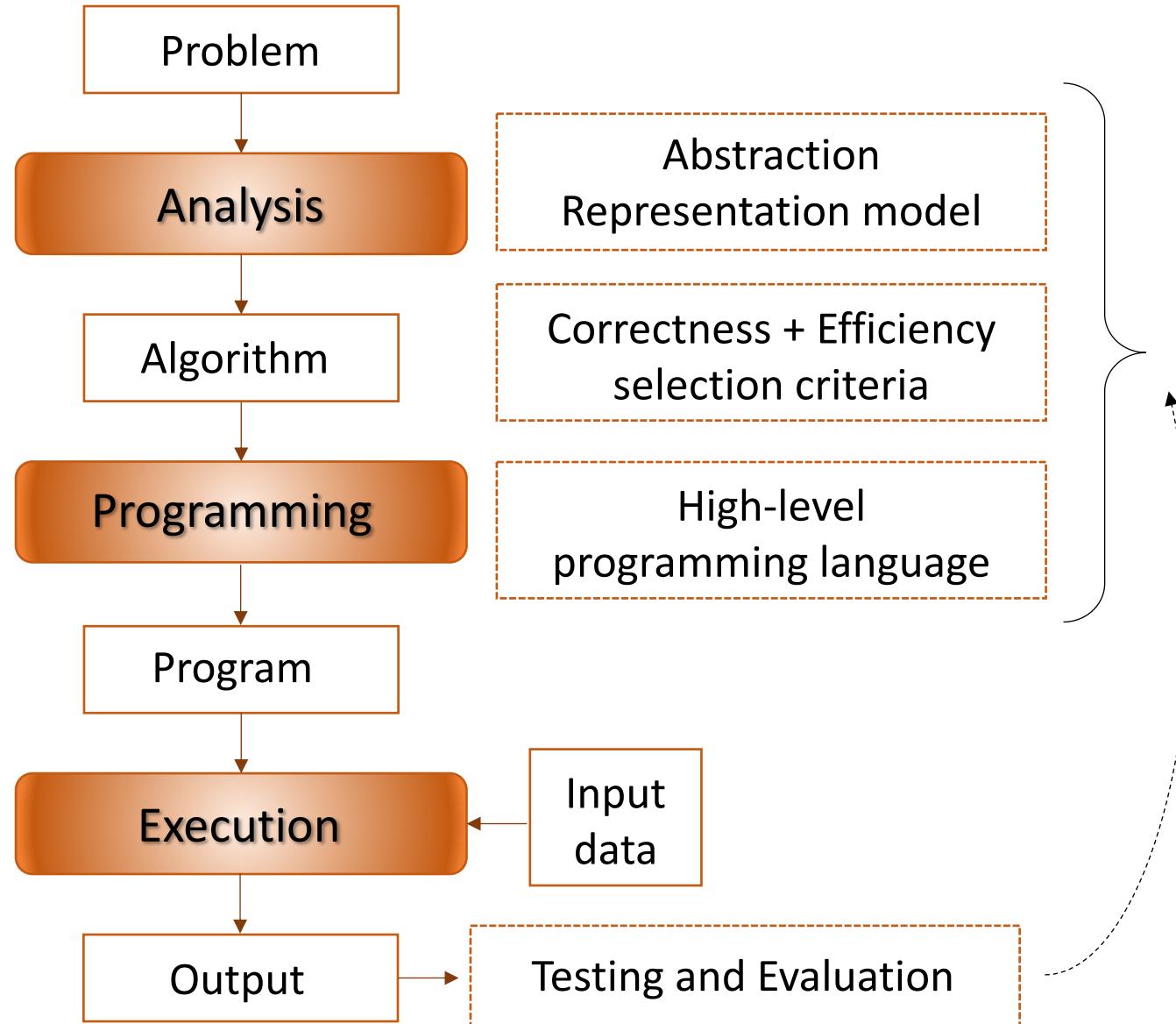
Introduction to the *science (art)* of computational problem solving



- ✓ **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:**

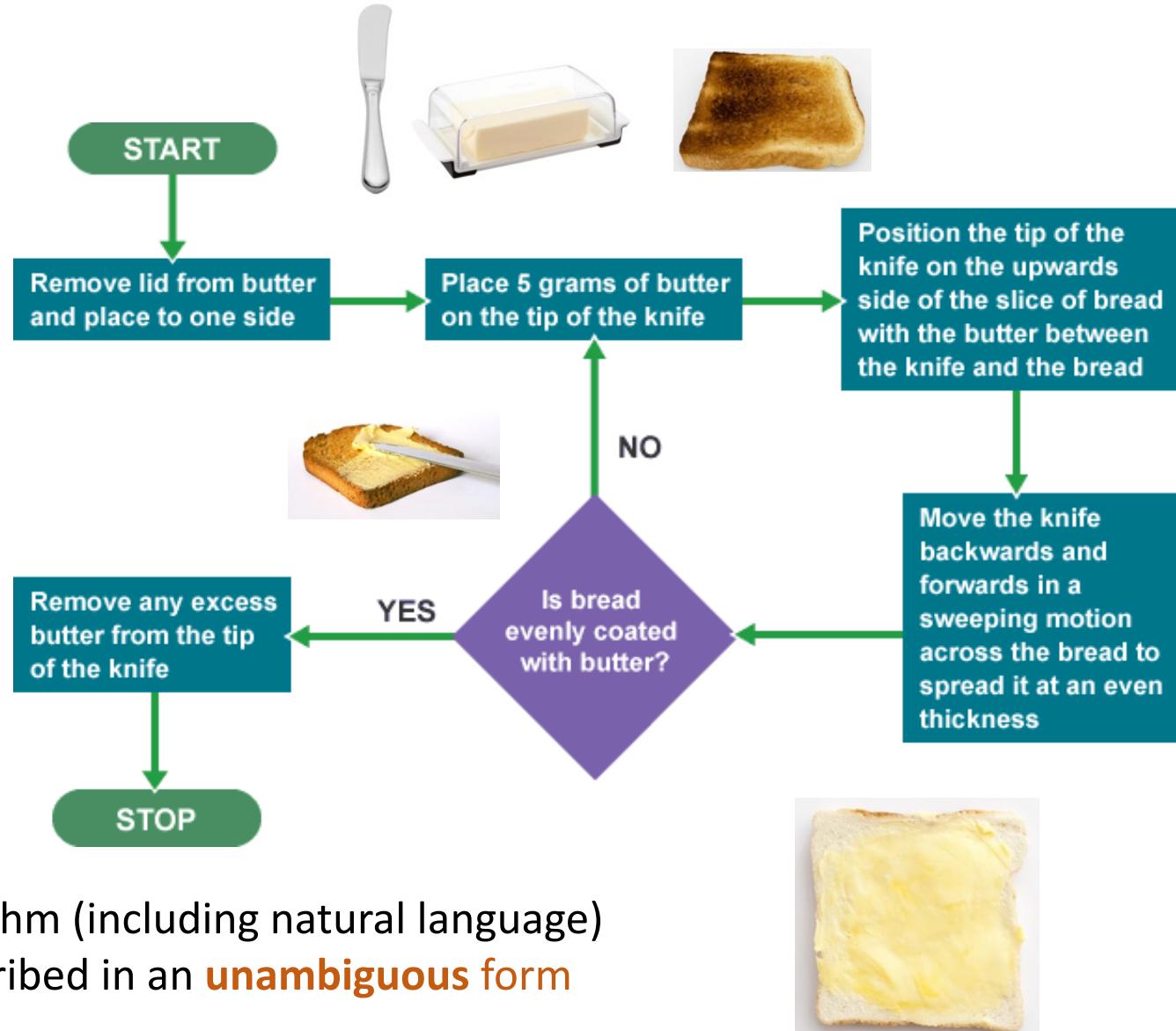
- ✓ A finite list of instructions that describe a **computation**
- ✓ when the instructions are executed on a provided set of inputs, the computation proceeds **step by step** through a set of well-defined states (configurations)
- ✓ eventually, it ends, with some outputs being produced

- **Program:**

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

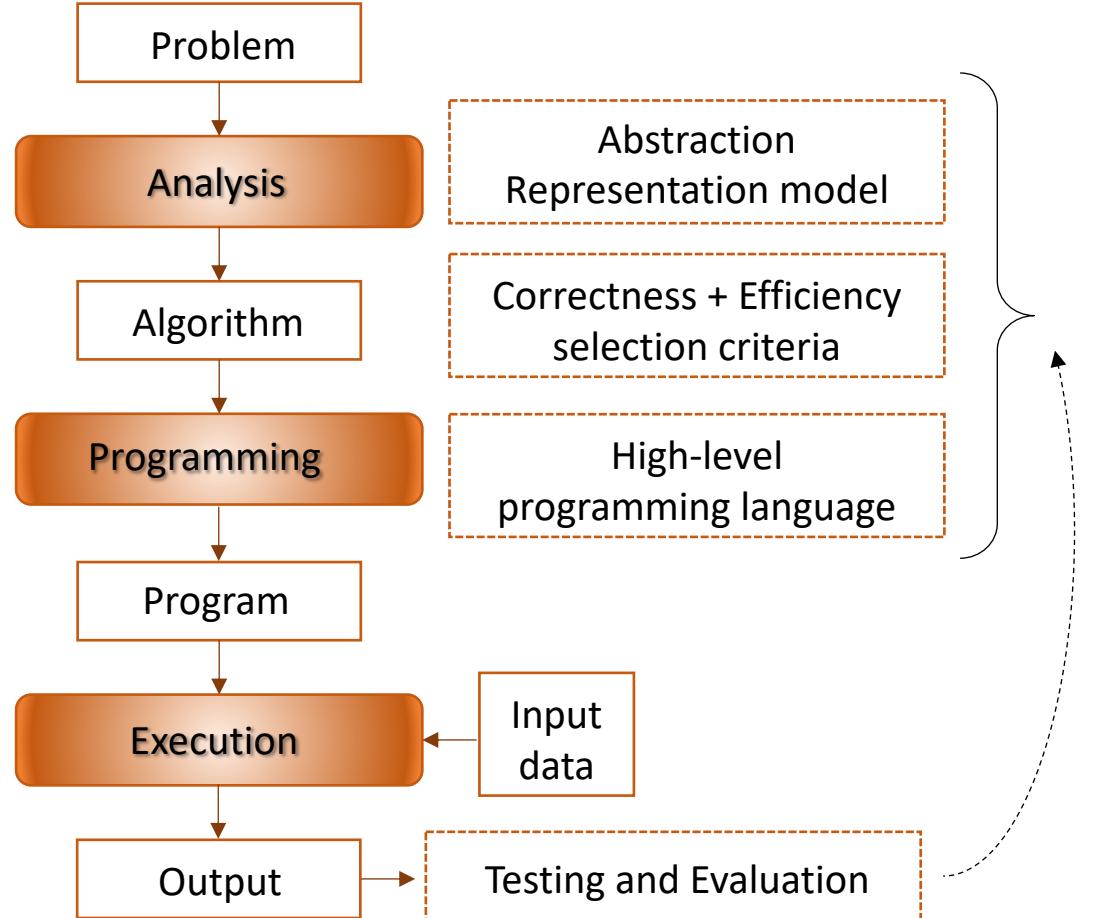
Algorithms

- ✓ 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



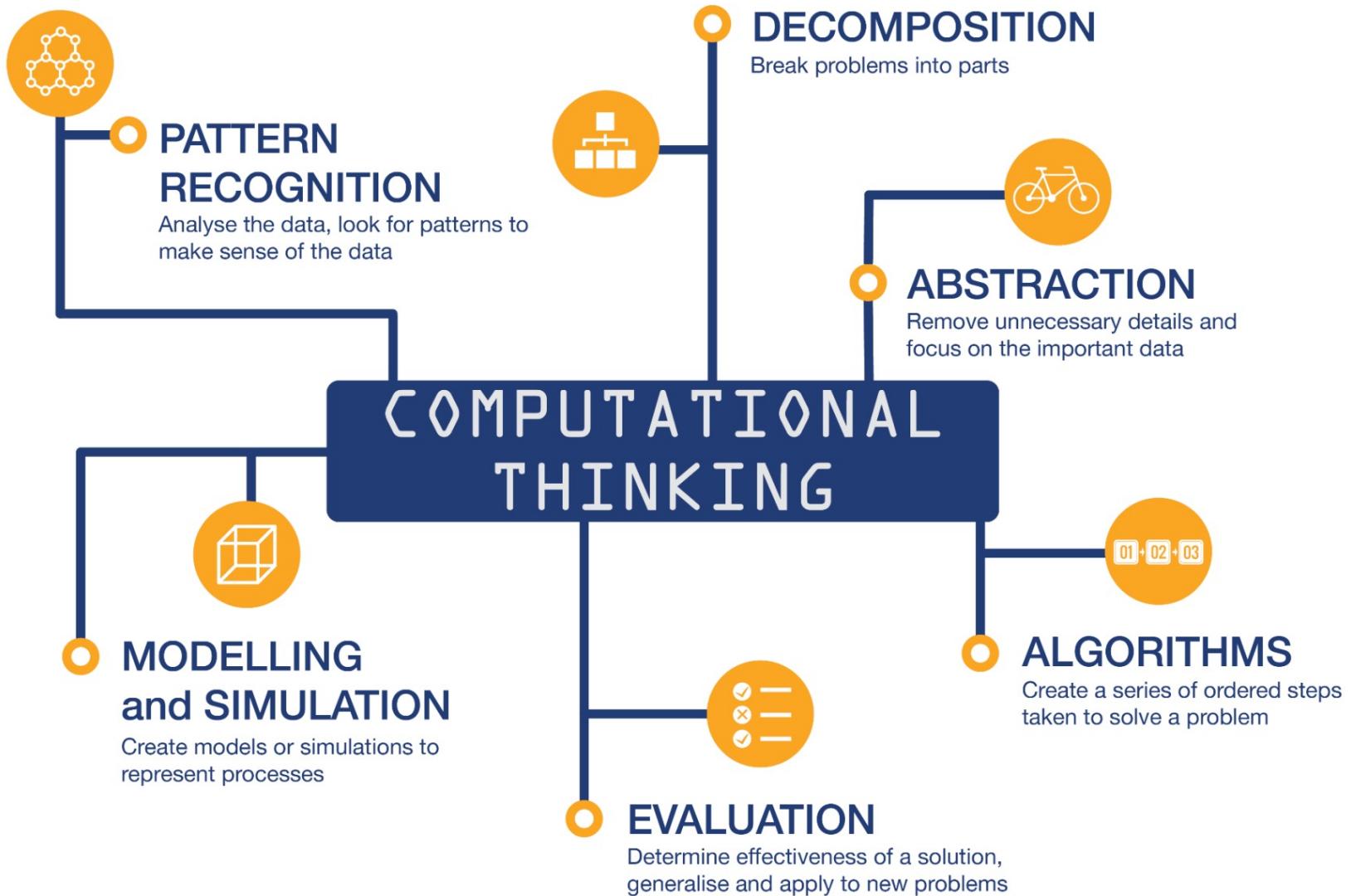
- Many ways to **describe** the same algorithm (including natural language)
- To be implemented, it needs to be described in an **unambiguous** form
- Can be applied to a **class of problems**

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!**

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)

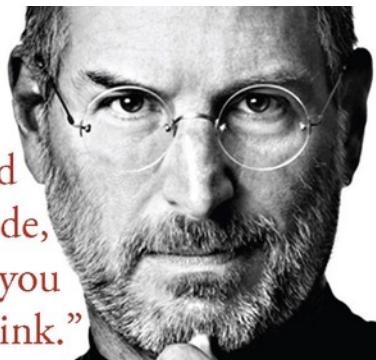
Motivations for following 15-110 ...

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

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!



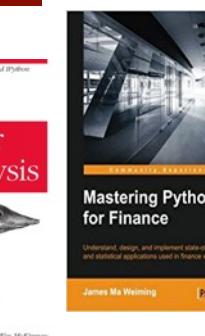
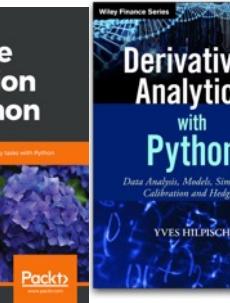
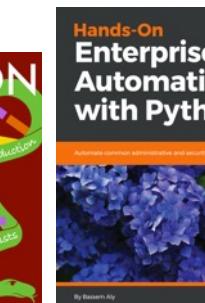
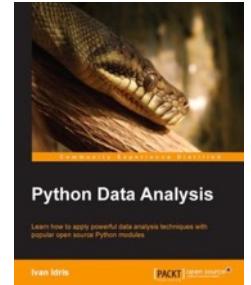
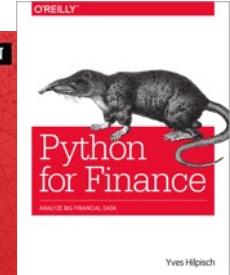
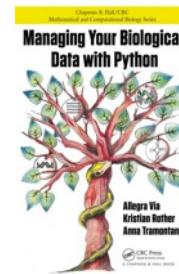
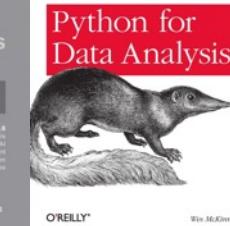
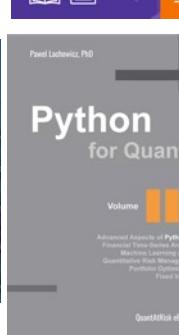
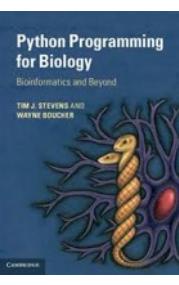
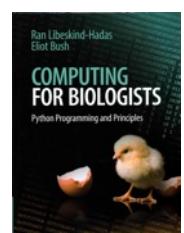
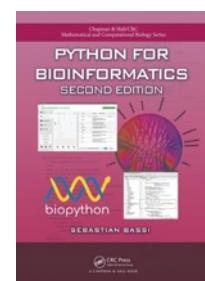
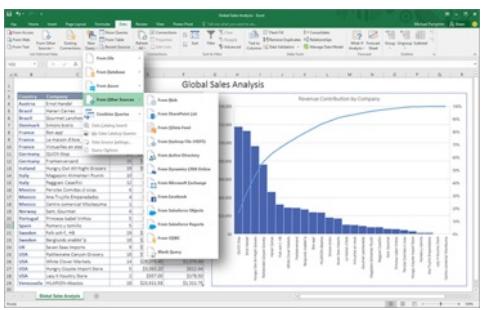
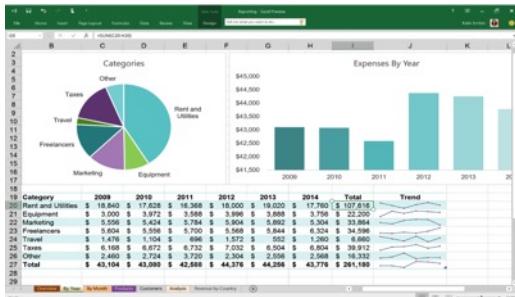
- Why understanding computation and learning computer programming can be rewarding?

- ✓ So many (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

<https://web2.qatar.cmu.edu/cs/15110/>

15-110 Schedule Staff Grading Assignments Interviews Tools Policies Piazza Autolab

15-110: Principles of Computing (Fall 2023)

Overview

Computing can be defined as the study of computational processes that manipulate information. A *computational process* is one that can be automated, and thus executed by a computer. Therefore, one of the main underlying questions is: **what can be (efficiently) automated?** This course aims at introducing the science (and the art) of computing to students with little or no prior background in this subject.



Given the great amount of topics involved in computing, the course will focus on a subset of its core aspects, providing a brief, yet substantial introduction to many concepts. The goal is to provide an idea of what can be automated, and how to realize when it is useful (or, most often, necessary) to employ computation and computers to accomplish a complex goal.

The course will take the student along the way that starts from a complex, possibly large problem to solve, and then move step by step to its *abstraction*, to its formalization into an *algorithmic recipe*, to the encoding of the algorithm using the *constructs of the python language*, to the run-time *execution* and *error correction* of the programming code, to the *efficiency analysis* of the developed algorithm and code.

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

A screenshot of a web browser displaying the Piazza platform. The address bar shows the URL: <https://piazza.com/class/ljlg79kg2vc4sb/post/6>. The page header includes the Piazza logo, the course number 15110, and navigation links for Q & A, Resources, Statistics, and Manage Class.

- **Autolab** for submitting homework

A screenshot of a web browser displaying the Autolab platform. The address bar shows the URL: <autolab.andrew.cmu.edu/courses/15110q-f23/assessments>. The page header features the Autolab logo, and the main content area shows a course navigation menu with the path: 15110q-f23: Principles of Computing (f23).

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

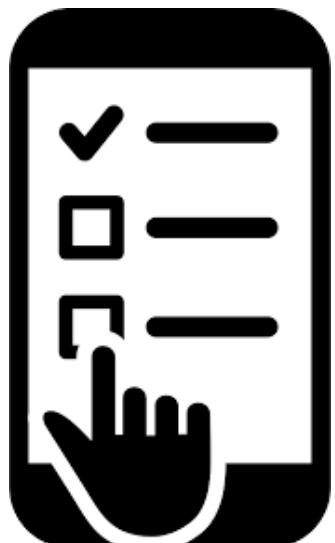
A screenshot of a web browser displaying the Gradescope platform. The address bar shows the URL: <gradescope.com/courses/579372>. The page header includes the Gradescope logo and the course identifier 15-110Q. The main content area displays course information for Fall 2023, Course ID: 579372, and a Description section with a note to edit on the Course Settings page.

Logistics: In-class rules

Be *alive* and **participate!**



Ask questions!



Answer polls!

4% of your grade

Academic integrity: Don't mess it up!!

Plagiarism

Cheating

Inappropriate
Collaboration

Duplicate
Submission

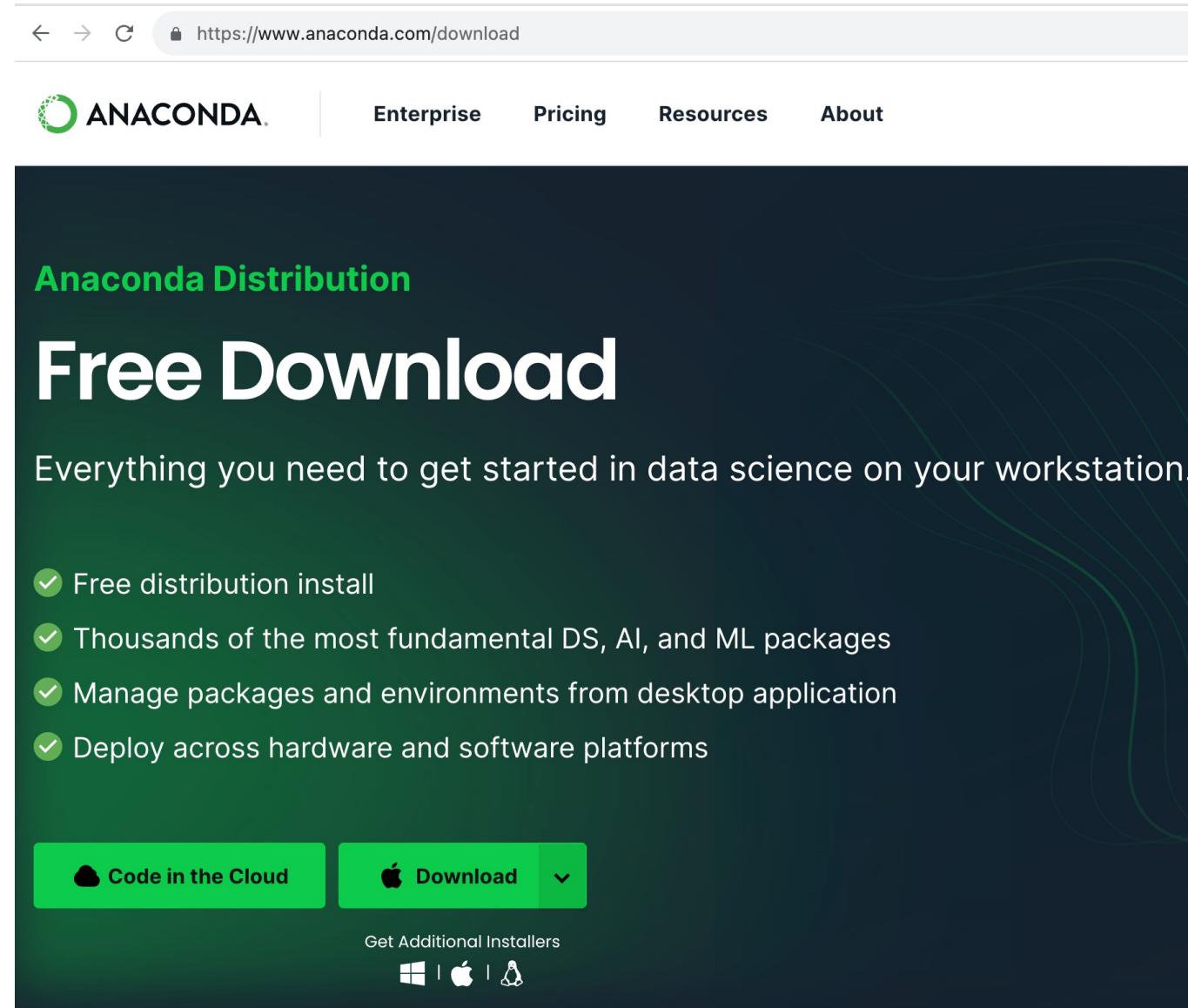
Personation

Academic Fraud

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



The screenshot shows a web browser displaying the Anaconda distribution download page at <https://www.anaconda.com/download>. The page has a dark green background with white text. At the top, there is a navigation bar with the Anaconda logo, links for Enterprise, Pricing, Resources, and About, and a back/forward/search bar. The main heading is "Anaconda Distribution" followed by a large "Free Download" button. Below the button, a sub-headline reads "Everything you need to get started in data science on your workstation." A bulleted list of features follows, each preceded by a green checkmark icon. At the bottom, there are two prominent green buttons: "Code in the Cloud" and "Download" (with a dropdown arrow). Below these buttons, there is a link to "Get Additional Installers" and icons for Windows, macOS, and Linux.

← → ⌂ https://www.anaconda.com/download

ANACONDA Enterprise Pricing Resources About

Anaconda Distribution

Free Download

Everything you need to get started in data science on your workstation.

- ✓ Free distribution install
- ✓ Thousands of the most fundamental DS, AI, and ML packages
- ✓ Manage packages and environments from desktop application
- ✓ Deploy across hardware and software platforms

Code in the Cloud Download ▾

Get Additional Installers

Windows | macOS | Linux

ANACONDA Navigator

ANACONDA NAVIGATOR

Connect ▾

All applications on base (root) Channels

Home Environments Learning Community

Anaconda Notebooks New! Cloud notebooks with hundreds of packages ready to code. Learn More

A full Python IDE directly from the browser Documentation Anaconda Blog

 DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.
[Install](#)

 JupyterLab
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
[Launch](#)

 Notebook
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
[Launch](#)

 IP[y]:
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
[Launch](#)

 Spyder
Scientific PYthon Development EnvIRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
[Launch](#)

 Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.
[Launch](#)

 IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.
[Launch](#)

 ORACLE Cloud Infrastructure
Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools
[Launch](#)

 Glueviz
1.2.4 Multidimensional data visualization across files. Explore relationships within and among related datasets.
[Install](#)

 Orange 3
3.32.0 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
[Install](#)

 PyCharm Professional

 RStudio
1.1.456

[Twitter](#) [YouTube](#) [GitHub](#)

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

The screenshot displays the Spyder IDE interface with the following components:

- Editor:** Shows the script file `temp.py` containing Python code for generating two plots. The code imports `numpy` and `matplotlib.pyplot`, defines variables `m`, `x1`, `x2`, `y1`, and `y2`, and uses `plt.subplots` to create two subplots showing damped and undamped oscillations over time.
- Variable explorer:** A table showing the values of variables defined in the script:

Name	Type	Size	Value
<code>m</code>	str	1	hello class!
<code>x1</code>	float64	(50,)	[0. 0.10204082 0.20408163 ... 4.79591837 4.89795918 5. ...]
<code>x2</code>	float64	(50,)	[0. 0.04081633 0.08163265 ... 1.91836735 1.95918367 2. ...]
<code>y1</code>	float64	(50,)	[1. 0.72367065 0.2320026 ... 0.00235117 0.00597998 0.00673795 ...]
<code>y2</code>	float64	(50,)	[1. 0.96729486 0.8713187 ... 0.8713187 0.96729486 1. ...]
- Console:** Displays the command `In [2]: runfile('/Users/giannidicaro/.spyder-py3/temp.py', wdir='/Users/giannidicaro/.spyder-py3')` and the output "hello class!" followed by two plots.
- Plots:** Two subplots titled "A tale of 2 subplots". The top plot, labeled "Damped oscillation", shows a decaying sinusoidal wave starting at approximately 1.0. The bottom plot, labeled "Undamped", shows a sinusoidal wave oscillating between -1.0 and 1.0.

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

In Anaconda: Python Notebooks in JupyterLab

The screenshot shows a JupyterLab interface with a sidebar containing a file tree and a main area with a code cell and a plot.

File Tree:

- / ... / 288-Spring2023 / Lectures_Notebooks /
- Name
- 288-S23-10-DataCleaning.ipynb
- 288-S23-11-Outliers.ipynb
- 288-S23-12-Scaling.ipynb
- 288-S23-13-Correlations.ipynb
- 288-S23-14-FeatureEngineering.ipynb
- 288-S23-15-PCA.ipynb
- 288-S23-16-FeatureEngineering.ipynb
- 288-S23-17-FeatureEngineering.ipynb
- 288-S23-18-FeatureEngineering.ipynb
- 288-S23-19-Clustering_1.ipynb
- 288-S23-20-Clustering_2.ipynb
- 288-S23-21-TimeSeries.ipynb
- 288-S23-24-EnsembleMethods.ipynb
- 288-S23-26-SVM_Kernel.ipynb
- 288-S23-29-NN-3.ipynb
- 288-S23-30-NN-4.ipynb
- 288-S23-31-NN-5.ipynb
- 288-S23-32-Regularization.ipynb
- 288-S23-6-RegressionExperiments.ipynb
- 288-S23-7-LinearRegression.ipynb
- 288-S23-8-Classification.ipynb
- 288-S23-9-Classification.ipynb
- 288-S23-9-Extras_Visualizations.ipynb
- 288-S23-Intro_Notebooks.ipynb
- 315-F21-6-ModelSelection.ipynb
- 488-S20-12-DataWrangling.ipynb
- 488-S20-14-FeatureEngineering.ipynb
- 488-S20-16-FeatureEngineering.ipynb
- 488-S20-17-Clustering_1.ipynb
- 488-S20-18-Clustering_2.ipynb
- 488-S20-19-Clustering_3.ipynb
- 488-S20-21-LinearRegression.ipynb
- 488-S20-22-LinearRegression.ipynb
- 488-S20-23-LinearRegression.ipynb
- 488-S20-32-Perceptron.ipynb
- 488-S20-33-NN.ipynb
- 488-S20-FeatureEngineering.ipynb
- 488-S20-UL.ipynb
- CoronaVirus.ipynb
- GMM.ipynb
- hemoglobin.txt

Main Area:

Add data points and legends

Let's add also the training data points to the plot to get a better view of the behavior of the classifier

```
[18]: # Create a custom color map
from matplotlib.colors import ListedColormap
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'blue'])

[19]: plt.figure()
plt.xlabel(iris.feature_names[feature_x])
plt.ylabel(iris.feature_names[feature_y])

# the color map assigns a different color to the different
# values of predictions_z
plt.pcolormesh(xx, yy, predictions_z, cmap=cmap_light)

# plot the training points
scatter = plt.scatter(selected_features[:,feature_x],
                      selected_features[:,feature_y],
                      c=iris.target,
                      cmap=cmap_bold,
                      edgecolor='black',
                      s=20)

plt.legend(*scatter.legend_elements(), loc="upper right", title="Classes")

# set the limits of the plots based on the ranges
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())

plt.title('Classification of Iris data using'
          ' k-NN (k={})'.format(neighbors_num))

plt.show()
```

Plot:

Classification of Iris data using k-NN (k=5)

The plot shows the relationship between Sepal Length (cm) on the x-axis (ranging from 4.0 to 8.0) and Sepal Width (cm) on the y-axis (ranging from 2.0 to 5.0). The data points are colored by class (0, 1, or 2). Overlaid on the scatter plot are three distinct colored regions representing the decision boundaries of a k-Nearest Neighbors classifier with k=5. The orange region covers the upper-left area, the cyan region covers the lower-left area, and the blue region covers the lower-right area. A legend titled "Classes" indicates the color mapping for classes 0, 1, and 2.

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.... ☺