Data Science and Advanced Programming — Lecture 1

Introduction to the course logistics, Computer Hardware and Software, "Hello, World" in Python

Simon Scheidegger
Department of Economics, University of Lausanne, Switzerland

September 15th, 2025 | 12:30 - 16:00 | Internef 263

What is this course about?

Roadmap of the course

- Part I: Python Foundations (Weeks 1–6)
- Part II: Basics of Data Science (Weeks 7–11)
- Part III: Advanced Programming & Wrap-Up (Weeks 12-14)
- Software Engineering (distributed across the lectures)
- Libraries

Data Science

The activities of Greater Data Science are classified into 6 divisions:

1. Data Exploration and Preparation
2. Data Representation and Transformation
3. Computing with Data
4. Data Modeling
5. Data Visualization and Presentation
6. *Science about Data Science

$\rightarrow$ For all those activities, you need to be able to program a computer.

To become a Data Scientist (n.):

Data Scientist (n.):
Person who is better at statistics than any software engineer and better at software engineering than any statistician.

Course Schedule: Part I (Weeks 1–6)

- **Week 1 (Sep 15):** Course Overview, Setup, Unix Basics
- **Week 2 (Sep 22):** No Class – *Swiss Federal Fast*
- **Week 3 (Sep 29):** Python Fundamentals I (Basics, Control Flow, Git)
- **Week 4 (Oct 6):** Python Fundamentals II (Functions, Data Structures, Recursion)
- **Week 5 (Oct 13):** Special Session: Generative AI (Guest Lecture)
- **Week 6 (Oct 20):** Python Fundamentals III (OOP, Classes, Efficiency, Debugging)

Course Schedule: Part II & III (Weeks 7–14)

- **Week 7 (Oct 27):** Linear Regression
- **Week 8 (Nov 3):** Classification
- **Week 9 (Nov 10):** Unsupervised Learning
- **Week 10 (Nov 17):** Deep Learning Primer
- **Week 11 (Nov 24):** Best Practices in Data Science
- **Week 12 (Dec 1):** Introduction to High-Performance Computing
- **Week 13 (Dec 8):** High-Performance Computing with Python
- **Week 14 (Dec 15):** Capstone Project Presentations & Wrap-Up

What is this course about?

- This is a VERY FAST-paced course! → Solve the problem sets when they come out.
- Not a regular programmer? PRACTICE. PRACTICE? PRACTICE!
- Programming is like Sports/ Playing an instrument: → You CANNOT passively absorb programming as a skill.
- Look at the example codes before the lecture and follow along.
- Don't be afraid to try out the codes in the lectures!

→ We want to acquire skills, not pure knowledge

"Non scholae sed vitae": Admin and Logistics

- Meeting time: Mondays, 12:30 - 15:15.
- **Exercises**: Weekly, Mondays 15:15 – 16:00 (details communicated by the TAs).
- **Lecturer:** Simon Scheidegger (simon.scheidegger@unil.ch).
- **TA team**: Anna Smirnova (anna.smirnova@unil.ch; TA lead), Francesco Brunamonti (francesco.brunamonti@unil.ch), Zhongshan Chen (zhongshan.chen@unil.ch).
- **Nuvolos Cloud Support**: support@nuvolos.cloud
- **Course Website**: Lecture notes are on *Nuvolos.cloud*
- To enroll in this class, please click on this enrollment key, and follow the steps.
- E-office hours for details/exercises: ask the TAs directly.
- Common questions, I will answer at the beginning of every lecture.

Lecture Materials & Questions for the class

- Lecture Materials and questions related to the lecture can/should be posted ahead of time on this link:

Class Website

Your lecturer: Simon Scheidegger

- Associate Prof. in Economics.
- Visiting Senior Fellow at the Grantham Research Institute, LSE.
- BIS Research Fellow.
- Research in Computational finance and economics, Deep learning, Climate change economics, and Machine learning applied to economics and finance, high-performance computing, macro-finance.
- Associate Editor, The Journal of Financial Econometrics.
- Find some of my research here in case you are interested.

The TA team

- Anna Smirnova: Ph.D. student in Economics.
- Francesco Brunamonti: Ph.D. student in Finance.
- Zhongshan Chen: Ph.D. student in Finance.

Prerequisites — What do you need to know?

You should know how to do **math**:

- Multivariate calculus
- Probability/statistics
- A bit of Algorithms / Big O notation
- Linear Algebra
- Optimization

Assignments will be in Python

Enhance your productivity: manage your work/codes with Github

- Open a free GitHub account!
- github.com
- Soon, we will learn how to use Git

Grading

- Every student has to provide a capstone project that illustrates what was learned.
- Each student individually has to propose a data science project and work on it over the course of the semester.
- The due date to submit the project is in the last week of the semester.
- The deliverables are:
  - a report of about 10 pages lengths.
  - a GitHub repository with the related code and data.
  - a video recording of a maximum of 10 minutes length that presents the project, the findings, etc.
- We will award the grades based on whether the captstone project demonstrates an understanding of the material. There will be no exams.

Previous projects

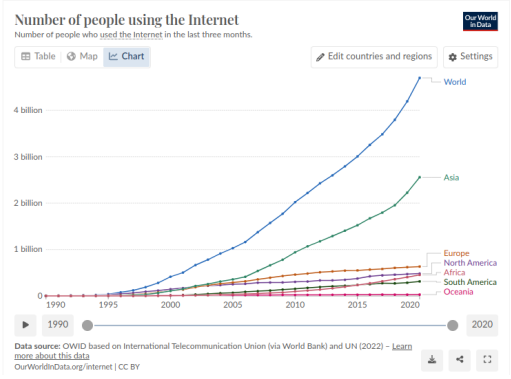
- A few success stories out of this class
- Florence Hugard (now at E4S at EPFL)
- Malik Lechekhab (Ph.D. student in computational science at USI Lugano)
- Tim Holt (Ph.D. student in computational science at USI Lugano)
- $\rightarrow$ all got a grade of 6 in their projects.
- Other examples that got a top grade see on Nuvolos in the folder: `capstone_project/examples`

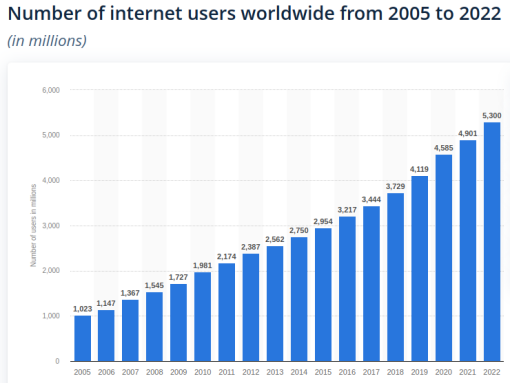
"We are drowning in information and starving for knowledge."

— Rutherford D. Roger

Big Data and its availability

https://ourworldindata.org/internet





Big Data and its availability

http://www.live-counter.com/how-big-is-the-internet

Size of the internet as we speak: TBD Petabytes

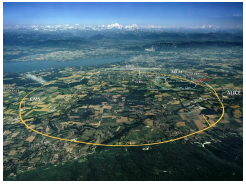
- 1 Gigabyte $\sim$ 1000 MB
- 1 Terabyte $\sim$ 1000 GB
- 1 Petabyte $\sim$ 1000 TB
- 1 Exabyte $\sim$ 1000 PB
- 1 Zettabyte $\sim$ 1000 EB

1 Gigabyte: If an author writes a book of about 190 pages, more specifically, of 383,561 characters (with spaces and punctuation included) every week for 50 years — this would be a billion letters or bytes.
1 Exabyte: 212 million DVDs weighing 3,404 tons.
1 Zettabyte: 1,000,000,000,000,000,000,000 bytes or characters.
This, printed on graph paper (with one letter in each mm^2 square) would be a paper measuring a billion km. The entire surface of the Earth (510 million km^2) would be covered by a layer of paper almost twice.

Other sources of Big Data

- **Scientific experiments**
  - CERN (e.g., LHC) generates $\sim$ **25 petabytes** per year (2012).
  - LIGO generates $\sim$ **1 Petabyte** per year
- **Numerical computations**
- ...

https://www.olcf.ornl.gov/summit/



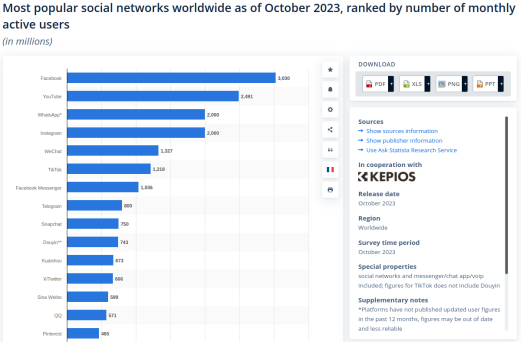
https://home.cern/

https://www.ligo.caltech.edu/

The need for Data Analytics

- Widespread use of personal computers and wireless communication leads to "big data".
- We are both producers and consumers of data.
- Data is often not random, it has structure, e.g., customer behavior.
- We need "big theory" to extract that structure from data for
  - Understanding the data-generating process.
  - Making predictions for the future.

$\Longrightarrow$ **We need Data Analytics**



The purpose of Data Analytics

Xia, B. S., & Gong, P. (2014). Review of business intelligence through data analysis. *Benchmarking: An International Journal*, 21(2), 300-311

Data analysis is a process of

- inspecting data
- cleansing data
- transforming data
- modeling data

with the goal of

1. discovering useful information.
2. informing conclusions.
3. supporting decision-making.

Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, while being used in different business, science, and social science domains.
In today's business, data analysis is playing a role in

- making decisions more scientific.
- helping the business achieve effective operation.

Data Mining

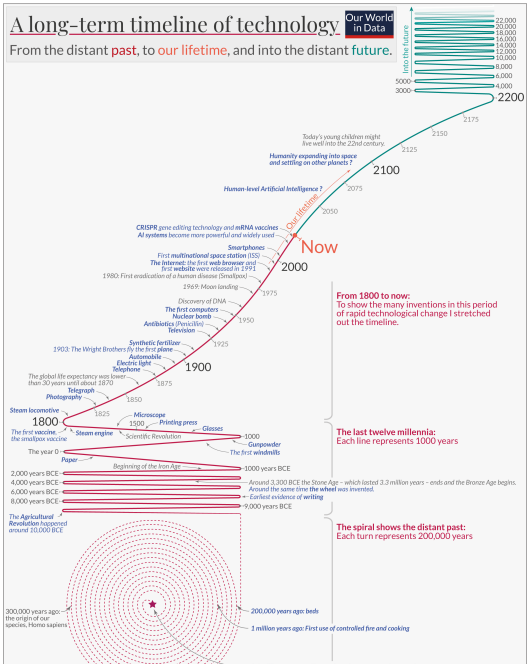
Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

- **Retail:** Market basket analysis, Customer relationship management (CRM).
- **Finance:** Credit scoring, fraud detection, trading.
- **Manufacturing:** Control, robotics, troubleshooting.
- **Medicine:** Medical diagnosis.
- **Telecommunications:** Spam filters, intrusion detection.
- **Bioinformatics:** Motifs, alignment.
- **Web mining:** Search engines

Why Machine Learning?

- **Machine learning** aims at gaining insights from data and making predictions based on it.
- Build a model that is a good and useful approximation to the data.
- Machine learning methods have been investigated for more than 60 years, but became mainstream only recently due to more data being available and advances in computing power ("*Moore's Law*").
- There is no need to "learn" to calculate, for example, the payroll.
- Learning is used when:
  - Human expertise does not exist (navigating on Mars).
  - Humans are unable to explain their expertise (speech recognition).
  - Solution changes in time (routing on a computer network).
  - Solution needs to be adapted to particular cases (user biometrics).
- You're relying on machine learning every day, maybe without being aware of it! You certainly use a Smart Phone?

Speed of Scientific Discovery

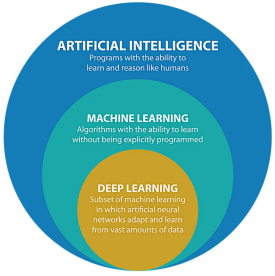


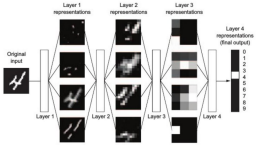
Set some terminology straight

Artificial intelligence (AI)
Can computers be made to "think"? — a question whose ramifications we're still exploring today. A concise definition of the field would be as follows: The effort to automate intellectual tasks normally performed by humans.

Machine Learning (ML)

Deep Learning as a particular example of an ML technique



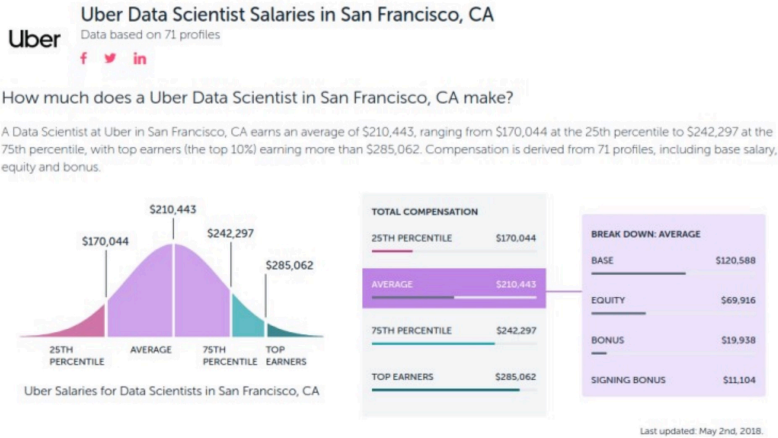


ML – the "coolest thing" in science

"*A breakthrough in machine learning would be worth ten Microsofts*"

— Bill Gates

Dozens of Billions/year USD$ globally spend on AI & ML.



Structured Data

- We often deal with structured data.
- Example: Data about monthly rent of real estate in a city.

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Unstructured and Semi-structured Data

- Many interesting datasets are unstructured, typically natural language texts written by humans, or semi-structured, interleaving structured and unstructured data.
- Example: Newspaper articles with assigned categories.

Category	Content
Sports	Bayern Munich was defeated by Real Madrid in the Champions League quarter finals...
Politics	Theresa May called for a snap general election on Monday...

Feature Types (I)

When working with structured data, we distinguish different types of features, depending on which operations can be applied.

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Dataset consists of five features.

Feature Types (II)

- **Nominal** features can be compared (==, !=) and counted (e.g., gender of a person, color of a car).
- **Ordinal** features can, in addition, be compared (<, >) (e.g., customer satisfaction level, energy class of car).
- **Numerical** features allow in addition for arithmetic operations (+, -, *, /), so that we can compute the difference between values, compute their mean, compute their variance, etc. (e.g., the fuel consumption of a car, income of a household).

Feature Types (III)

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Area: Num; EstateType: Nom; DistanceToCenter: Num; EnergyClass: Ord; MonthlyRent:Num.

ML Applications

- **Association**.
- **Supervised Learning**.
  Assume that training data is available from which they can learn to predict a target feature based on other features (e.g., monthly rent based on area).
  - **Classification**.
  - **Regression**.
- **Unsupervised Learning**
  Take a given dataset and aim at gaining insights by identifying patterns, e.g., by grouping similar data points.
- **Reinforcement Learning**.

Structured Data

- We often deal with structured data.
- Example: Data about monthly rent of real estate in a city.

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Unstructured and Semi-structured Data

- Many interesting datasets are unstructured, typically natural language texts written by humans, or semi-structured, interleaving structured and unstructured data.
- Example: Newspaper articles with assigned categories.

Category	Content
Sports	Bayern Munich was defeated by Real Madrid in the Champions League quarter finals...
Politics	Theresa May called for a snap general election on Monday...

Feature Types (I)

When working with structured data, we distinguish different types of features, depending on which operations can be applied.

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Dataset consists of five features.

Feature Types (II)

- **Nominal** features can be compared (==, !=) and counted (e.g., gender of a person, color of a car).
- **Ordinal** features can, in addition, be compared (<, >) (e.g., customer satisfaction level, energy class of car).
- **Numerical** features allow in addition for arithmetic operations (+, -, *, /), so that we can compute the difference between values, compute their mean, compute their variance, etc. (e.g., the fuel consumption of a car, income of a household).

Feature Types (III)

Area	EstateType	DistanceToCenter	EnergyClass	MonthlyRent
58.20	Apartment	1.1	A	450
122.20	House	5.6	A	620
28.00	Apartment	0.5	B	320
8.00	Storage	6.3	B	150
75.78	House	2.2	C	375
66.00	Office	1.6	A	525
10.00	Storage	9.2	D	120

Area: Num; EstateType: Nom; DistanceToCenter: Num; EnergyClass: Ord; MonthlyRent:Num.

ML Applications

- **Association**.
- **Supervised Learning**.
  Assume that training data is available from which they can learn to predict a target feature based on other features (e.g., monthly rent based on area).
  - **Classification**.
  - **Regression**.
- **Unsupervised Learning**
  Take a given dataset and aim at gaining insights by identifying patterns, e.g., by grouping similar data points.
- **Reinforcement Learning**.

Supervised Regression

- Regression aims at predicting a numerical target feature based on one or multiple other (numerical) features.
- Example: Price of a used car.
  - x : car attributes
  - y : price
  - $y = h(x|\theta)$
  - $h()$: model
  - θ : parameters

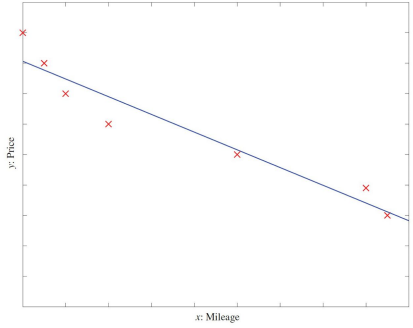


Fig. from Alpaydin (2014)

Supervised Classification

Example 1: Spam Classification

- Decide which emails are Spam and which are not.
- Goal: Use emails seen so far to produce a good prediction rule for **future** data.

Example 2: Credit Scoring

- Differentiating between low-risk and high-risk customers from their income and savings.
- Discriminant: `IF income >` θ_2 `AND savings >` θ_2 `THEN low-risk ELSE high-risk`

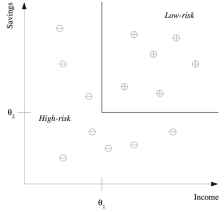


Fig. from Alpaydin (2014)

Classification: More applications

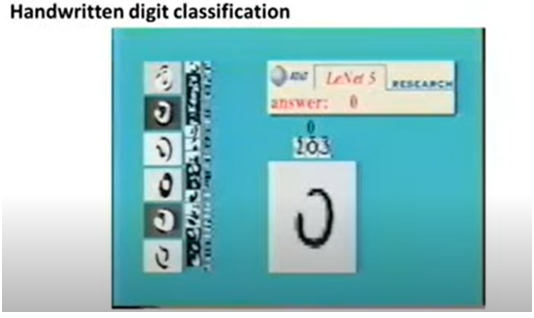
- **Face recognition**: Pose, lighting, occlusion (glasses, beard), make-up, hairstyle.
- **Character recognition**: Different handwriting styles.
- **Speech recognition**: Temporal dependency.
- **Medical diagnosis**: From symptoms to illnesses.
- **Biometrics**: Recognition/authentication using physical and/or behavioral characteristics such as face, iris, signature, etc.
- Outlier/novelty detection.

Random sampling of MNIST

Handwritten Digit Classification (LeNet)

Movie from the early 90's. We have come a long way since then...

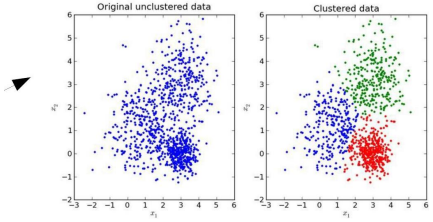
Handwritten Digit Classification – by Yann Lecun



Unsupervised Learning

Learning “what normally happens”.

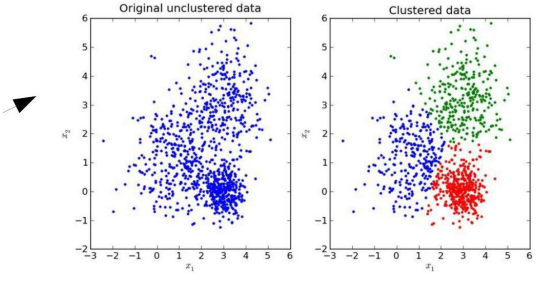
- No output.
- Clustering: Grouping similar instances.
- Example applications:
  - Customer segmentation.
  - Image compression: Color quantization.
  - Bioinformatics: Learning motifs.



Unsupervised Learning

Learning “what normally happens”.

- No output.
- Clustering: Grouping similar instances.
- Example applications:
  - Customer segmentation.
  - Image compression: Color quantization.
  - Bioinformatics: Learning motifs.



Reinforcement Learning

- Learning a policy: A sequence of outputs.
- No supervised output but delayed reward.
- Credit assignment problem.
- Game playing.
- Robot in a maze.
- Multiple agents, partial observability, ...

→ *DeepMind's Qlearning*.

Self-Driving Cars

- Carnegie Mellon University — 1990's: Self Driving Cars S1E2: ALVINN.
- Google — 2017: Waymo.
- Classification.
- Regression.
- Reinforcement learning.
- Prediction.

$\rightarrow$ *BostonDynamics' AtlasRobotCanDoParkour.*

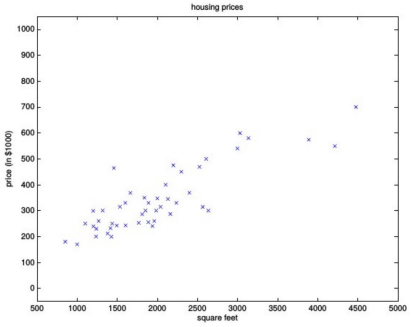
$\rightarrow$ *Handyman.*

Building an ML Algorithm

- Optimize a performance criterion using an example data or past experience.
- Role of Statistics: Inference from a sample.
- Role of computer science: Efficient algorithms to
  - Solve the optimization problem.
  - Representing and evaluating the model for inference.

Building an ML Algorithm (II)

Living area (feet ^{2})	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

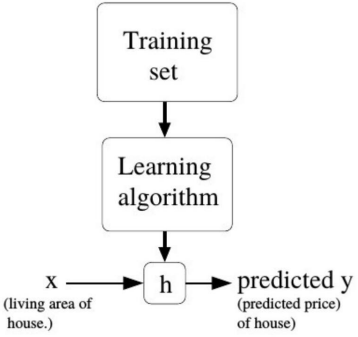


Given data like this, how can we learn to predict the prices of other houses as a function of the size of their living areas?

Building an ML Algorithm (III)

- $x(i)$: **"input" variables** (living area in this example), also called **input features**.
- $y(i)$: **"output" / target variable** that we are trying to predict (price).
- **Training example**: a pair $(x(i), y(i))$.
- **Training set**: a list of m training examples $(x(i), y(i)); i = 1, ..., m$.

To perform supervised learning, we must decide how we're going to represent **functions/hypotheses** h in a computer.



Building an ML Algorithm (IV)

- **Model/Hypothesis:**

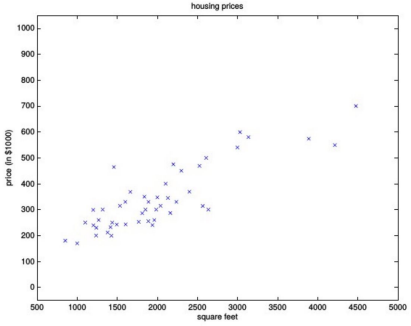
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ_i's: parameters

- **Cost Function:**

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$\Longrightarrow$ Minimize $J(\theta)$ in order to obtain the coefficients θ.



Building an ML Algorithm (V)

In general, Machine Learning in 3 Steps:

- Choose a model $h(x \mid \theta)$.
- Define a cost function $J(\theta \mid x)$.
- Optimization procedure to find $\theta*$ that minimizes $J(\theta)$.

Computationally, we need data, linear algebra, statistics tools, and optimization routines.

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

— Edsger Dijkstra, 1972 Turing Award Lecture

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.
- If you have to **commute**, the traffic flows are monitored by computers in a (usually vain) attempt to avoid traffic jams.

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.
- If you have to **commute**, the traffic flows are monitored by computers in a (usually vain) attempt to avoid traffic jams.
- You prefer to take the **train**? That train will also be computerized; → some even operate without a driver, and the train's subsystems, such as announcements, braking, and ticketing, involve lots of computers.

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.
- If you have to **commute**, the traffic flows are monitored by computers in a (usually vain) attempt to avoid traffic jams.
- You prefer to take the **train**? That train will also be computerized; → some even operate without a driver, and the train's subsystems, such as announcements, braking, and ticketing, involve lots of computers.
- Today's **entertainment industry** (music, movies, television, stage shows) is among the largest users of computers. Even non-cartoon movies use (computer) animation heavily.

Our Civilization Runs on Software

- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.
- If you have to **commute**, the traffic flows are monitored by computers in a (usually vain) attempt to avoid traffic jams.
- You prefer to take the **train**? That train will also be computerized; → some even operate without a driver, and the train's subsystems, such as announcements, braking, and ticketing, involve lots of computers.
- Today's **entertainment industry** (music, movies, television, stage shows) is among the largest users of computers. Even non-cartoon movies use (computer) animation heavily.
- **Music and photography are also digital** (i.e., using computers) for both recording and delivery.

Our Civilization Runs on Software

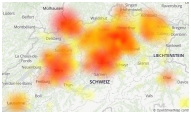
- How many computers do you depend on for what you do during a day?
- You eat: getting the food to you is a major effort requiring minor miracles of planning, transport, and storage.
  - The management of the distribution networks is of course computerized, as are the communication systems that stitch them all together.
- If you have to **commute**, the traffic flows are monitored by computers in a (usually vain) attempt to avoid traffic jams.
- You prefer to take the **train**? That train will also be computerized; → some even operate without a driver, and the train's subsystems, such as announcements, braking, and ticketing, involve lots of computers.
- Today's **entertainment industry** (music, movies, television, stage shows) is among the largest users of computers. Even non-cartoon movies use (computer) animation heavily.
- **Music and photography are also digital** (i.e., using computers) for both recording and delivery.
- **Should you become ill**, the tests your doctor's diagnosis will involve computers.

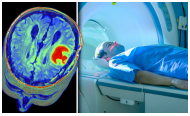
Software in Our Daily Life

- Software is a **collection of programs running on some computer.**
- Sometimes, we can see the computer.
- Often, we can see only something that contains the computer, such as a telephone, a camera, a bread maker, a car, or a wind turbine.

Software in Our Daily Life

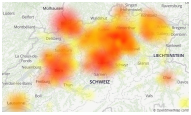
- Software is a **collection of programs running on some computer.**
- Sometimes, we can see the computer.
- Often, we can see only something that contains the computer, such as a telephone, a camera, a bread maker, a car, or a wind turbine.
- We can see what that software does.
- We can be annoyed or hurt if it doesn't do what it is supposed to do.
- Research, i.e., science itself relies heavily on computers.

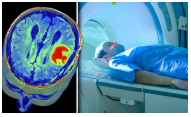




Software in Our Daily Life

- Software is a **collection of programs running on some computer.**
- Sometimes, we can see the computer.
- Often, we can see only something that contains the computer, such as a telephone, a camera, a bread maker, a car, or a wind turbine.
- We can see what that software does.
- We can be annoyed or hurt if it doesn't do what it is supposed to do.
- Research, i.e., science itself relies heavily on computers.
- Improving software and finding new uses for software are two of the ways an individual can help improve the lives of many.





→ **Programming plays an essential role in all of that.**

A few quiz questions to get an overview of your knowledge

1. How are your programming skills?
   - 1.1 I have never programmed at all
   - 1.2 I have never programmed in Python, C nor C++
   - 1.3 I know some basic Python
   - 1.4 I know some basic C
   - 1.5 I know some basic C++
   - 1.6 I know C++ well
   - 1.7 I am a C++ guru

A few quiz questions to get an overview of your knowledge

1. What operating system are you using (for programming)?
   - 1.1 I have no idea
   - 1.2 Windows
   - 1.3 Linux
   - 1.4 macOS (my computer looks pretty and has some bitten apple on it)
   - 1.5 Other

A few quiz questions to get an overview of your knowledge

1. What compiler do you use?
   - 1.1 None, I don't know what it is
   - 1.2 Whatever the compile button in my IDE (integrated dev. env.) uses
   - 1.3 GNU Compiler Collection
   - 1.4 Clang
   - 1.5 MinGW
   - 1.6 My own

A few quiz questions to get an overview of your knowledge

1. Do you know build systems?
   - 1.1 I have never heard about it
   - 1.2 I have used Automake
   - 1.3 I have used Lego
   - 1.4 I have used CMake
   - 1.5 I have used Scons
   - 1.6 I know it well
   - 1.7 I am a guru

A few quiz questions to get an overview of your knowledge

1. Do you know version control?
   - 1.1 I have never heard about it
   - 1.2 I have used CVS
   - 1.3 I have used SVN
   - 1.4 I have used Git
   - 1.5 I have used Copy&Paste
   - 1.6 I know it well
   - 1.7 I am a guru

A few quiz questions to get an overview of your knowledge

1. How is the integer value $+1$ represented in binary in a 16-bit integer
   - 1.1 0000000000000000
   - 1.2 0000000000000001
   - 1.3 1000000000000000
   - 1.4 1111111111111111
   - 1.5 1000000000000001
   - 1.6 1111111111111110

A few quiz questions to get an overview of your knowledge

1. What is the size of the string “Hello” in Python, i.e. the result of ``len(``Hello'')``
   1.1 1
   1.2 5
   1.3 6
   1.4 7
   1.5 8

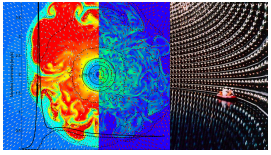
Computer Science

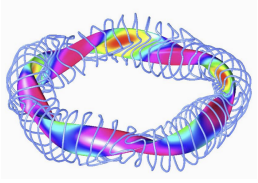
- **Computer science** is the discipline that seeks to build a scientific foundation for such topics as
  - computer design
  - computer programming
  - information processing
  - algorithmic solutions of problems
  - the algorithmic process itself
- It provides the **underpinnings** for
  - today's computer applications as well as
  - the foundations for tomorrow's computing infrastructure.

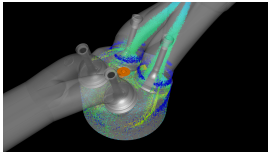
Computational Science

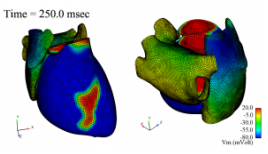
- Computational science: a rapidly growing multidisciplinary field that uses advanced computing capabilities to understand and solve complex problems.
- It is an area of science which spans many disciplines (comp. finance, comp. econ, comp. physics, comp. biology,...).
- $\rightarrow$ At its core it involves the development of models and simulations to understand complex systems.

Computational science aims to make the complexity of those systems tractable.

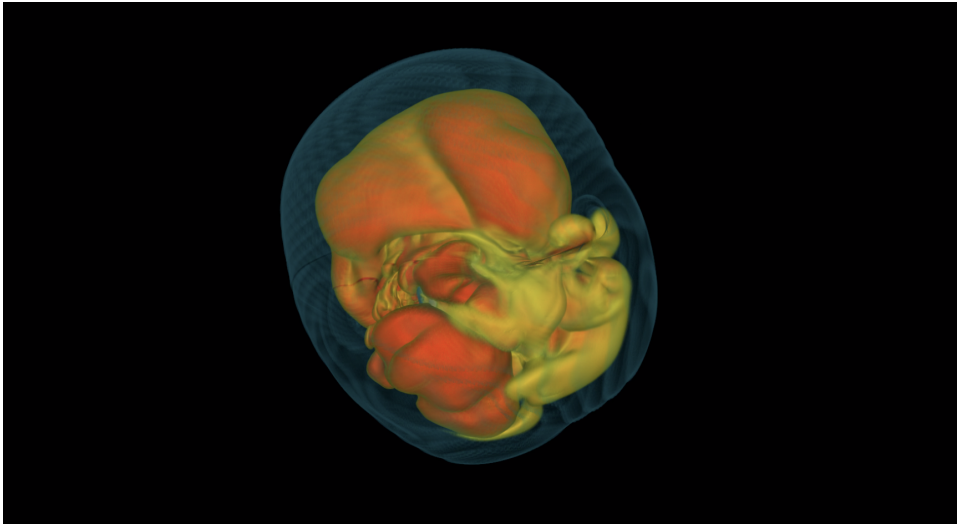








Example: A star explosion



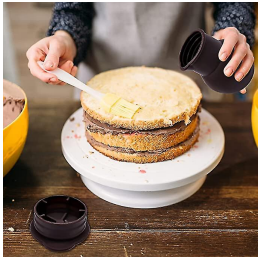
https://wwwmpa.mpa-garching.mpg.de/ccsnarchive/movies/s20_3Ds_HD.mp4

The Role of Algorithms

- We begin with the most fundamental concept of computer science - that of an **algorithm**.

> Informally, an algorithm is a set of steps that defines how a task is performed.

- For example, there are
  - algorithms for **cooking** (called **recipes**),
  - for finding your way through a strange city (more commonly called **directions**).



Example of an Algorithm

- The study of algorithms began as a subject in mathematics.
- The search for algorithms was a significant activity of mathematicianslong before the development of today's computers. The goal was to find a single set of instructions that described how all problems of a particular type could be solved.
- One of the best known examples of this early research is the long division algorithm for finding the quotient of two multiple-digit numbers.

Description: This algorithm assumes that its input consists of two positive integers and proceeds to compute the greatest common divisor of these two values.

Procedure:

Step 1. Assign M and N the value of the larger and smaller of the two input values, respectively.

Step 2. Divide M by N, and call the remainder R.

Step 3. If R is not 0, then assign M the value of N, assign N the value of R, and return to step 2; otherwise, the greatest common divisor is the value currently assigned to N.

The Euclidean algorithm for finding the greatest common divisor of two positive integers.

Euclidean Algorithm

1. Compute greatest common divisor of $(48, 18)$.
2. Divide 48 by 18 to get a quotient of 2 remainder of 12.
3. Divide 18 by 12 to get a quotient of 1 and a remainder of 6.
4. Divide 12 by 6 to get a remainder of 0.

$\rightarrow$ This means that 6 is the "gcd".

We ignored the quotient in each step except to notice when the remainder reached 0, indicating that we had arrived at the answer.

Algorithm → Program

- Before a machine such as a computer can perform a task, an algorithm for performing that task must be discovered and represented in a form that is compatible with the machine.

→ A representation of an algorithm is called a program.

- For the convenience of humans, computer programs are usually printed on paper or displayed on computer screens.
- For the convenience of machines, programs are encoded in a manner compatible with the technology of the machine.

Programming: The process of developing a program, encoding it in machine-compatible form, and inserting it into a machine is called programming.

- **Programs, and the algorithms they represent**, are collectively referred to as **software**, in contrast to the **machinery** itself, which is known as **hardware**.

Programmer and User

- **Programmer** — the person who solves the problem and writes the instructions for the computer.
- **User** — any person who uses the program written by the programmer.

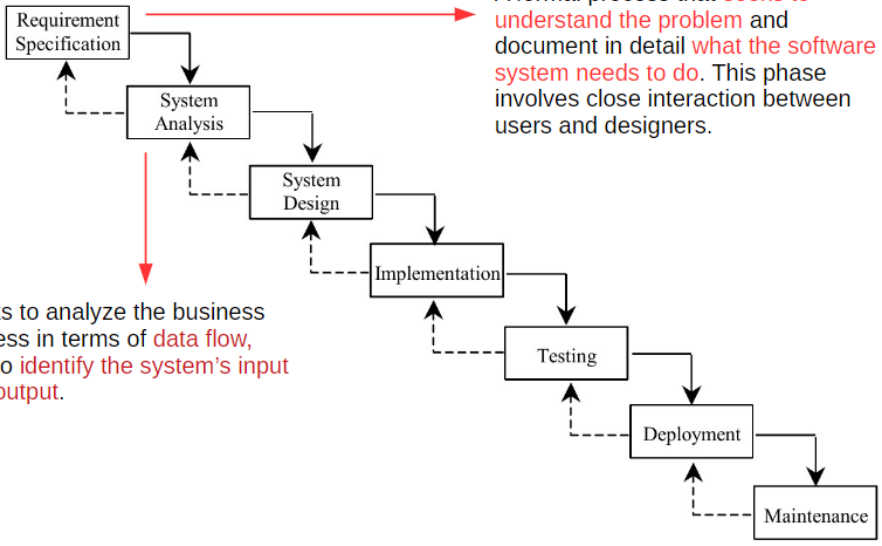
Program Planning

- Always have a plan before trying to write a program.
- The more complicated the problem, the more complex the plan must be.
- Planning before coding saves time.

Program Development Cycle

1. **Analyze**: Define the problem.
2. **Design**: Plan the solution to the problem.
3. **Choose the interface**: Select the objects (text boxes, buttons, etc.).
4. **Code**: Translate the algorithm into a programming language.
5. **Test and debug**: Locate and remove any errors in the program.
6. **Complete the documentation**: Organize all the materials that describe the program.

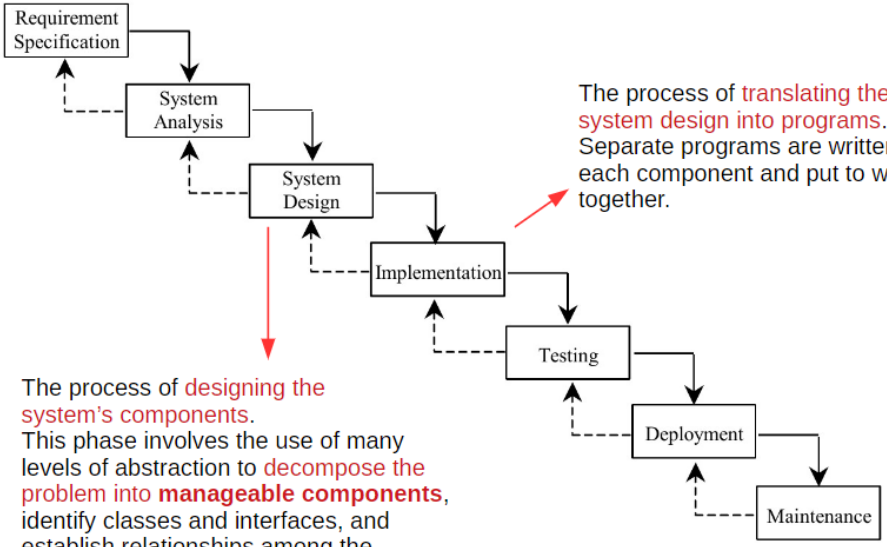
Software Development Process



A formal process that seeks to understand the problem and document in detail what the software system needs to do. This phase involves close interaction between users and designers.

Seeks to analyze the business process in terms of data flow, and to identify the system's input and output.

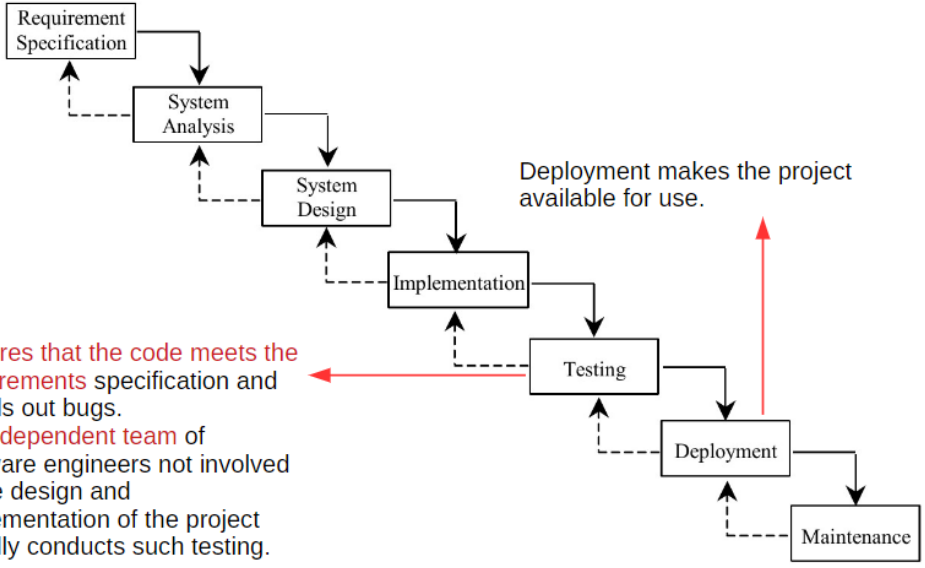
System Design & Implementation



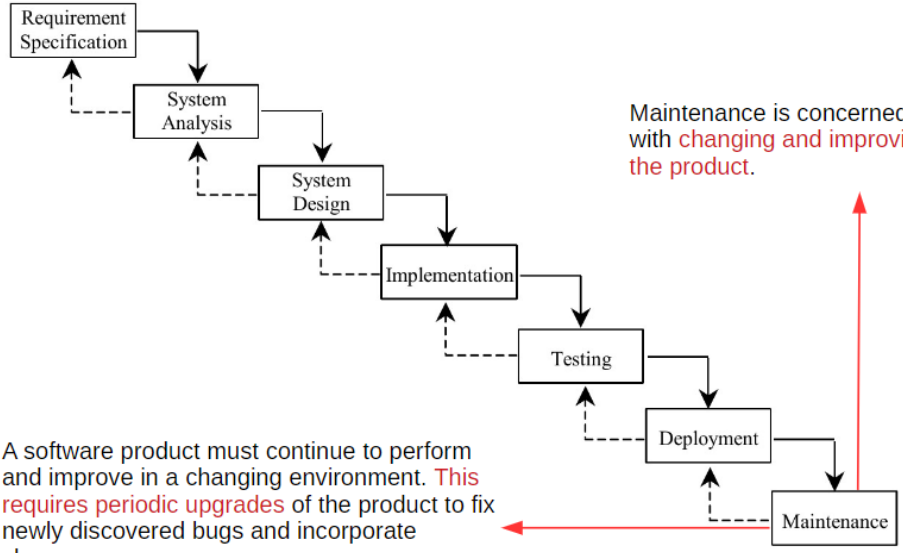
The process of translating the system design into programs. Separate programs are written for each component and put to work together.

The process of designing the system's components. This phase involves the use of many levels of abstraction to decompose the problem into **manageable components**, identify classes and interfaces, and establish relationships among the classes and interfaces.

Testing & Deployment



Maintainence



Maintenance is concerned with changing and improving the product.

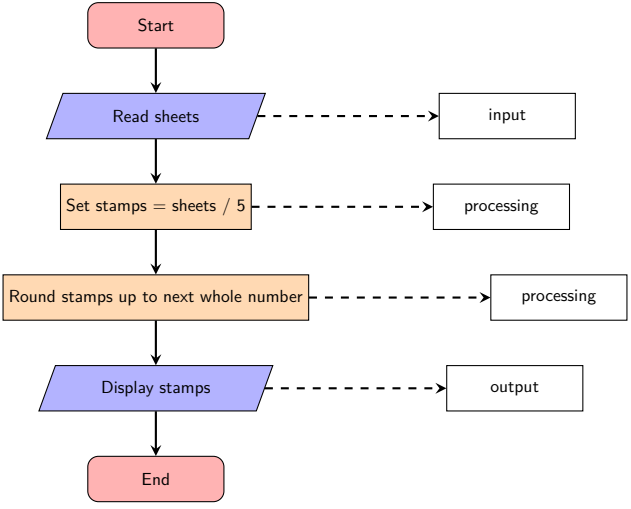
A software product must continue to perform and improve in a changing environment. This requires periodic upgrades of the product to fix newly discovered bugs and incorporate changes.

Three tools are used to convert algorithms into computer programs:

- **Flowchart**: Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- **Pseudo-code**: Uses English-like phrases, e.g., with some Python terms to outline the program.
- **Hierarchy chart**: Shows how the different parts of a program relate to each other.

Flowchart Example

Example: Determine the proper number of stamps for a letter.



Pseudocode Example

Pseudocode uses English-like phrases with some Python/C++ terms to outline the task.

Determine the proper number of stamps for a letter

Read Sheets ***(input)***

Set the number of stamps to Sheets / 5 ***(processing)***

Round the number of stamps up to the next whole number ***(processing)***

Display the number of stamps ***(output)***

A Computer

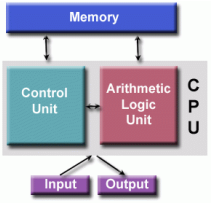
A computer is a machine that can:

- **Accept input**. Input could be entered by a human typing at a keyboard, received over a network, or provided automatically by sensors attached to the computer.
- **Execute a (mechanical) procedure**, that is, a procedure where each step can be executed without any thought.
- **Produce output.** Output could be data displayed to a human, but it could also be anything that effects the world outside the computer such as electrical signals that control how a device operates.

Basics: von Neumann Architecture

https://computing.llnl.gov/tutorials/parallel_comp

- Virtually all computers have followed this basic design. Comprised of **four main components**: Memory, Control Unit, Arithmetic Logic Unit, Input/Output.
- **Read/write, random access memory** is used to store both program instructions and data:
  - Program instructions are coded data which tell the computer to do something.
  - Data is simply information to be used by the program.
- **Control unit**
  - fetches instructions/data from memory, decodes the instructions and then sequentially coordinates operations to accomplish the programmed task.
- **Arithmetic unit**
  - performs basic arithmetic operations.
- **Input/Output**
  - interface to the human operator.



From a programming language to hardware

http://cse-lab.ethz.ch/index.php/teaching/42-teaching/classes/577-hpcsei

- A computer is a "stupid" device, only understands "on" and "off".
- The symbols for these states are 0 and 1 (binary).
- First programmers communicated in 0 and 1.
- Later programs where developed to translate from symbolic notation to binary. The first was called "**assembly**".

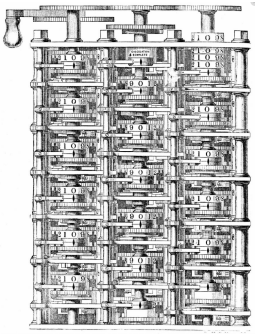
```
> add A, B (programmer writes in assembly language)
>1000110010100000 (assembly translates to machine language)
```

Advanced programming languages are better than "assembly":

- programmer thinks in a more natural language.
- productivity of software development.
- portability.

A Brief History of Computing

- Today's computers have an extensive genealogy.
- One of the earlier computing devices was the abacus.
- A few inventors began to experiment with the technology of gears.
- Among these were Blaise Pascal (1623-1662), Gottfried Wilhelm Leibniz (1646-1716), and Charles Babbage (1792-1871).
- These machines represented data through gear positioning, with data being input mechanically by establishing initial gear positions.
- Output from Pascal's and Leibniz's machines was achieved by observing the final gear positions.
- Babbage envisioned machines that would print results of computations on paper so that the possibility of transcription errors would be eliminated.

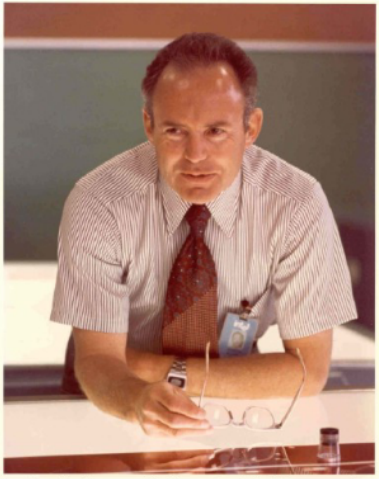


A Brief History of Computing (II)

- Examples of this progress include the electromechanical machine Mark I, completed in 1944 at Harvard University by Howard Aiken and a group of IBM engineers.
- These machines made heavy use of electronically controlled mechanical relays.
- A major step toward popularizing computing was the development of desktop computers.
- The origins of these machines can be traced to the computer hobbyists who built homemade computers from combinations of chips.
- It was within this "underground" of hobby activity that Steve Jobs and Stephen Wozniak built a commercially viable home computer and, in 1976, established Apple Computer, Inc. (now Apple Inc.) to manufacture and market their products.

Moore's Law

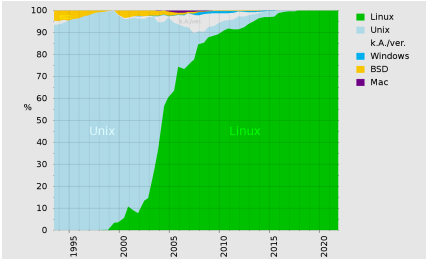
- The number of transistors on a chip doubles every 18 months (April 19, 1965).
  - More transistors means smaller transistors
  - Smaller transistors → shorter distances → faster signals.
  - Smaller transistors → fewer charges → faster switching
  - Thus, the CPU speed increases exponentially.
- Has worked for the past 50 years!
- How long will it continue?
  - Prototype chips at 10 GHz.
  - Insulating layers only 4 atoms thick!
  - , and we still reduce the size??
  - **Moore's law will probably stop working in this decade**.
  - **Software optimization will become more important**.



Gordon E. Moore, Co-founder, Intel Corporation.

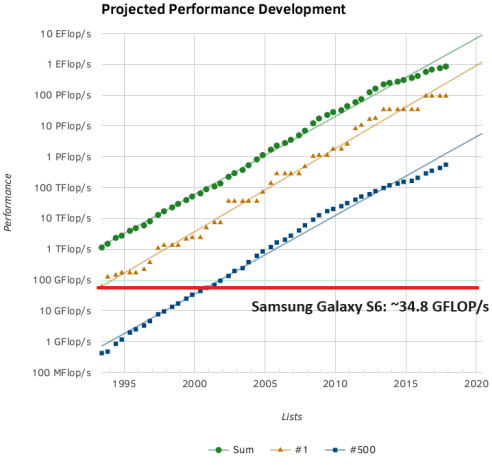
"Frankly I didn't expect to be so precise"

Moore's Law for Supercomputers



- HPC is the "third" pillar of science — nowadays is the dawn of an era (in physics, chemistry, biology, ..., next to experiment and theory).
- "In Silico" experiments.
- Modern Supercomputers (e.g. world's number 7 Summit at Oak Ridge National Lab) 200×10^{15} Flops/Sec. $\rightarrow$ 1 **day vs. Laptop:** $\sim 30,000$ **years.**

Moore's Law for Supercomputers



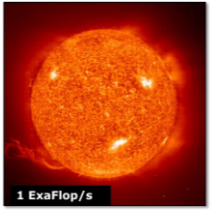
- $\rightarrow$ Move away from stylized models towards "realistically-sized" problems.
- **Creating "good" HPC software can be very difficult...**

Petascale Computers — How can we tap them?

*Slide adjusted from O. Schenk

Modern Supercomputers (Summit) 200×10^{15} Flops/Sec. $\rightarrow$ 1 day
vs. Laptop: $\sim 30,000y$.

Let us say you can print:
10^{18} numbers (Exaflop) = 100,000,000 km
(distance to the sun) stack printed per second
(Exaflop/s)



https://top500.org/lists/top500/2025/06/ (June 2025)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	793.40	930.00	13,088
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
6	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
7	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
8	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124

Data Representation in a Computer

- Computer memory is composed of billions of individual switches, each of which can be ON or OFF, but not at a state in between.

→ Each switch represents one **b**inary dig**it** (also called a *bit*)

Data Representation in a Computer

- Computer memory is composed of billions of individual switches, each of which can be ON or OFF, but not at a state in between.

→ Each switch represents one **b**inary dig**it** (also called a *bit*)

- The **ON** state is interpreted as a binary **1**, and the OFF state is interpreted as a binary **0** .

Data Representation in a Computer

- Computer memory is composed of billions of individual switches, each of which can be ON or OFF, but not at a state in between.

→ Each switch represents one **b**inary dig**it** (also called a *bit*)

- The **ON** state is interpreted as a binary **1**, and the OFF state is interpreted as a binary **0** .
- Taken by itself, a single switch can only represent the numbers 0 and 1.
- → Since we obviously need to work with numbers other than 0 and 1 , a number of bits are grouped together to represent each number used in a computer.

When several bits are grouped together, they can be used to represent numbers in the binary (base 2) number system.

Bit — Byte — Word

- The smallest common grouping of bits is called a byte.
- A byte is a group of 8 bits that are used together to represent a binary number. The byte is the fundamental unit used to measure the capacity of a computer's memory.

Bit — Byte — Word

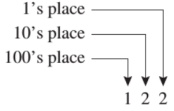
- The smallest common grouping of bits is called a byte.
- A byte is a group of 8 bits that are used together to represent a binary number. The byte is the fundamental unit used to measure the capacity of a computer's memory.
- For example, the personal computer on which I am writing these words has a main memory of 24 gigabytes (24,000,000,000 bytes) and a secondary memory (disk drive) with a storage of 2 tera-bytes (2,000,000,000,000 bytes).
- The next larger grouping of bits in a computer is called a word.
- A word consists of 2, 4, or more consecutive bytes that are used to represent a single number in memory.

Bit — Byte — Word

- The smallest common grouping of bits is called a byte.
- A byte is a group of 8 bits that are used together to represent a binary number. The byte is the fundamental unit used to measure the capacity of a computer's memory.
- For example, the personal computer on which I am writing these words has a main memory of 24 gigabytes ($24,000,000,000$ bytes) and a secondary memory (disk drive) with a storage of 2 tera-bytes ($2,000,000,000,000$ bytes).
- The next larger grouping of bits in a computer is called a word.
- A word consists of 2, 4, or more consecutive bytes that are used to represent a single number in memory.
- The size of a word varies from computer to computer $\rightarrow$ so words are not a particularly good way to judge the size of computer memories.
- Modern CPUs tend to use words with lengths of either 32 or 64 bits.

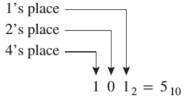
The Decimal Number System

- In the familiar base 10 number system, the smallest (rightmost) digit of a number is the ones place (10^0).
- The next digit is in the tens place (10^1), and the next one is in the hundreds place (10^2), etc.
- Thus, the number 122_{10} is really $(1 \times 10^2) + (2 \times 10^1) + (2 \times 10^0)$.
- Each digit is worth a power of 10 more than the digit to the right of it in the base 10 system.



The Binary Number System

- Similarly, in the binary number system, the smallest (rightmost) digit is the ones place (2^0).
- The next digit is in the twos place (2^1), and the next one is in the fours place (2^2), etc. Each digit is worth a power of 2 more than the digit to the right of it in the base 2 system. For example, the binary number 101_2 is really $(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 5$.



Representation of numbers

- Note that three binary digits can be used to representeight possible values: $0\,(=000_2)$ to $7\,(=111_2)$.

Representation of numbers

- Note that three binary digits can be used to representeight possible values: $0 (= 000_2)$ to $7 (= 111_2)$.
- In general, if n bits are grouped together to form a binary number, then they can represent 2^n possible values.

Representation of numbers

- Note that three binary digits can be used to representeight possible values: $0 (= 000_2)$ to $7 (= 111_2)$.
- In general, if n bits are grouped together to form a binary number, then they can represent 2^n possible values.
- Thus, a group of 8 bits (1 byte) can represent 256 possible values, and a group of 16 bits (2 bytes) can be used to represent 65,536 possible values.

Representation of numbers

- Note that three binary digits can be used to representeight possible values: $0 (= 000_2)$ to $7 (= 111_2)$.
- In general, if n bits are grouped together to form a binary number, then they can represent 2^n possible values.
- Thus, a group of 8 bits (1 byte) can represent 256 possible values, and a group of 16 bits (2 bytes) can be used to represent 65,536 possible values.
- A group of 32 bits (4 bytes) can be used to represent 4,294,967,296 possible values.
  - In a typical implementation, half of all possible values are reserved for representing negative numbers, and half of the values are reserved for representing zero plus the positive numbers.
  - Thus, a group of 8 bits (1 byte) is usually used to represent numbers between -128 and +127 , including 0 , and a group of 16 bits (2 bytes) is usually used to represent numbers between $-32,768$ and $+32,767$, including 0 .

Types of Data Stored in Memory

Three common types of data are stored in a computer's memory:

- **character data**
- **integer data**
- **real data** (numbers with a decimal point).

Each type of data has different characteristics, and takes up a different amount of memory in the computer.

Character Data

- The character data type consists of characters and symbols.
- A typical system for representing character data in a non-Oriental language must include the following symbols:
  1. The **26 uppercase letters A through Z**
  2. The **26 lowercase letters a through z**
  3. The **10 digits 0 through 9**
  4. **Miscellaneous common symbols**, such as ", 0, {}, [], !, ~, @, #, $, %, ^, &, and *.
  5. **Any special letters or symbols** required by the language, such as à, ç, ë, and .
- Since the total number of characters and symbols required to write western languages is less than 256, it is customary to use 1 byte of memory to store each character.

Therefore, 10,000 characters would occupy 10,000 bytes of the computer's memory.

Integer Data

- The integer data type consists of the positive integers, the negative integers, and zero.
- The amount of memory devoted to storing an integer will vary from computer to computer but will usually be $1, 2, 4$, or 8 bytes.

Integer Data

- The integer data type consists of the positive integers, the negative integers, and zero.
- The amount of memory devoted to storing an integer will vary from computer to computer but will usually be $1, 2, 4$, or 8 bytes.
- Four-byte integers (4 bytes) are the most common type in modern computers.

Integer Data

- The integer data type consists of the positive integers, the negative integers, and zero.
- The amount of memory devoted to storing an integer will vary from computer to computer but will usually be $1, 2, 4$, or 8 bytes.
- Four-byte integers (4 bytes) are the most common type in modern computers.
- Since a finite number of bits are used to store each value, **only integers that fall within a certain range can be represented on a computer.**
- Usually, the smallest number that can be stored in an n-bit integer is

$$\text{Smallest integer value } = -2^{n-1}$$

- and the largest number that can be stored in an n-bit integer is

$$\text{Largest integer value } = 2^{n-1} - 1$$

- For a 4-byte integer, the smallest and largest possible values are $-2,147,483,648$ and 2,147,483,647, respectively.

The Real Data Type

The integer data type has two fundamental limitations:

1. It is not possible to represent numbers with fractional parts (0.25, 1.5, 3.14159 , etc.) as integer data.
2. It is not possible to represent very large positive integers or very small negative integers, because there are not enough bits available to represent the value. The largest and smallest possible integers that can be stored in a given memory location will be given by equations from the previous slide.

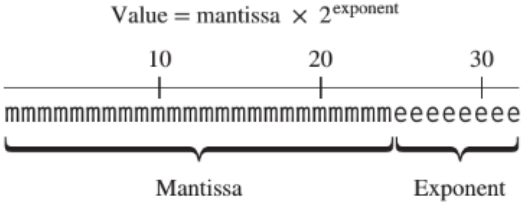
$\rightarrow$ To get around these limitations, computers include a **real or floating-point data type.**

The Real Data Type (II)

- The real numbers in a computer are similar to the scientific notation above, except that a computer works in the base 2 system instead of the base 10 system.
- Real numbers can e.g. occupy 32 bits (4 bytes) of computer memory, divided into two components:
  - a **24-bit mantissa and an 8-bit exponent.**
  - The mantissa contains a number between -1.0 and 1.0.
  - The exponent contains the power of 2 required to scale the number to its actual value.

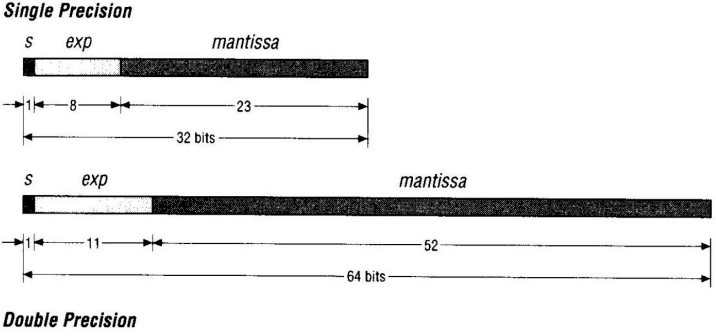
The Real Data Type (II)

- The real numbers in a computer are similar to the scientific notation above, except that a computer works in the base 2 system instead of the base 10 system.
- Real numbers can e.g. occupy 32 bits (4 bytes) of computer memory, divided into two components:
  - a **24-bit mantissa and an 8-bit exponent.**
  - The mantissa contains a number between -1.0 and 1.0.
  - The exponent contains the power of 2 required to scale the number to its actual value.



This floating-point number includes a 24-bit mantissa and an 8-bit exponent.

IEEE Floating Point Representation



Type	Exponent	Mantissa	Smallest	Largest	Base 10 accuracy
float	8	23	1.2E-38	3.4E+38	6-9
double	11	52	2.2E-308	1.8E+308	15-17

Precision and Range

- **Real numbers** are characterized by two quantities: **precision** and **range**.
  - **Precision**: is the number of significant digits that can be preserved in a number
  - Range: is the difference between the largest and smallest numbers that can be represented.

Precision and Range

- **Real numbers** are characterized by two quantities: **precision** and **range**.
  - **Precision**: is the number of significant digits that can be preserved in a number
  - Range: is the difference between the largest and smallest numbers that can be represented.
  - The precision of a real number depends on the number of bits in its mantissa
  - The range of the number depends on the number of bits in its exponent.
  - $\rightarrow$ A 24-bit mantissa can represent approximately $\pm 2^{23}$ numbers, or about **seven significant decimal digits**, so the precision of real numbers is about seven significant digits.

Precision and Range

- **Real numbers** are characterized by two quantities: **precision** and **range**.
  - **Precision**: is the number of significant digits that can be preserved in a number
  - Range: is the difference between the largest and smallest numbers that can be represented.
  - The precision of a real number depends on the number of bits in its mantissa
  - The range of the number depends on the number of bits in its exponent.
  - $\rightarrow$ A 24-bit mantissa can represent approximately $\pm 2^{23}$ numbers, or about **seven significant decimal digits**, so the precision of real numbers is about seven significant digits.
- An 8-bit exponent can represent multipliers between 2^{-128} and 2^{127}, so the range of real numbers is from about 10^{-38} to 10^{38}.

$\rightarrow$ Note that the real data type can represent numbers much larger or much smaller than integers can, but only with seven significant digits of precision.

Round-off Error

- When a value with more than seven digits of precision is stored in a real variable, only the most significant 7 bits of the number will be preserved.
- $\rightarrow$ The remaining information will be lost forever.

Round-off Error

- When a value with more than seven digits of precision is stored in a real variable, only the most significant 7 bits of the number will be preserved.
- $\rightarrow$ The remaining information will be lost forever.
- For example, if the value $12,345,679.9$ is stored in a real variable on a PC, it will be rounded off to $12,345,680.0$. This difference between the original value and the number stored in the computer is known as round-off error.

Round-off Error

- When a value with more than seven digits of precision is stored in a real variable, only the most significant 7 bits of the number will be preserved.
- $\rightarrow$ The remaining information will be lost forever.
- For example, if the value $12,345,679.9$ is stored in a real variable on a PC, it will be rounded off to $12,345,680.0$. This difference between the original value and the number stored in the computer is known as round-off error.
- You will use the real data type in many places throughout this lecture and in your programs after you finish this course. It is quite useful, but you must always remember the limitations associated with round-off error, or your programs might give you an unpleasant surprise.

Round-off Error

- When a value with more than seven digits of precision is stored in a real variable, only the most significant 7 bits of the number will be preserved.
- $\rightarrow$ The remaining information will be lost forever.
- For example, if the value $12,345,679.9$ is stored in a real variable on a PC, it will be rounded off to $12,345,680.0$. This difference between the original value and the number stored in the computer is known as round-off error.
- You will use the real data type in many places throughout this lecture and in your programs after you finish this course. It is quite useful, but you must always remember the limitations associated with round-off error, or your programs might give you an unpleasant surprise.
- For example, if your program must be able to distinguish between the numbers $1,000,000.0$ and $1,000,000.1$, then you cannot use the standard real data type.
- It simply does not have enough precision to tell the difference between these two numbers!

Floating Point Arithmetic

- Truncation can happen because of finite precision

$$\begin{array}{r} 1.00000 \\ 0.0000123 \\ 1.00001 \end{array}$$

- **Machine precision** ε is smallest number such that $1 + \varepsilon \neq 1$
- Be very careful about roundoff.

How to speak to a Computer

- When a computer executes a program, it executes a string of very simple operations such as load, store, add, subtract, multiply, and so on.

How to speak to a Computer

- When a computer executes a program, it executes a string of very simple operations such as load, store, add, subtract, multiply, and so on.
- Each such operation has a unique binary pattern called an **operation code (op code)** to specify it.

How to speak to a Computer

- When a computer executes a program, it executes a string of very simple operations such as load, store, add, subtract, multiply, and so on.
- Each such operation has a unique binary pattern called an **operation code (op code)** to specify it.
- The program that a computer executes is just a string of op codes (and the data associated with the op codes) in the order necessary to achieve a purpose.

How to speak to a Computer

- When a computer executes a program, it executes a string of very simple operations such as load, store, add, subtract, multiply, and so on.
- Each such operation has a unique binary pattern called an **operation code (op code)** to specify it.
- The program that a computer executes is just a string of op codes (and the data associated with the op codes) in the order necessary to achieve a purpose.
- Op codes are collectively called machine language, since they are the actual language that a computer recognizes and executes.

How to speak to a Computer

- When a computer executes a program, it executes a string of very simple operations such as load, store, add, subtract, multiply, and so on.
- Each such operation has a unique binary pattern called an **operation code (op code)** to specify it.
- The program that a computer executes is just a string of op codes (and the data associated with the op codes) in the order necessary to achieve a purpose.
- Op codes are collectively called machine language, since they are the actual language that a computer recognizes and executes.
- **Unfortunately, we humans find machine language very hard to work with.**
- We prefer to work with English-like statements and algebraic equations that are expressed in forms familiar to us, instead of arbitrary patterns of zeros and ones. We like to program computers with high-level languages.

$\rightarrow$ We write out our instructions in a high-level language and then use special programs called compilers and linkers to convert the instructions into the machine language that the computer understands.

Programming Languages

- There are many different high-level languages, with different characteristics.
- Some of them are designed to work well for business problems, while others are designed for general scientific use.
- Still others are especially suited for applications like operating systems programming.

→ It is important to pick a proper language to match the problem that you are trying to solve.

Aspects of Languages

Primitive constructs:

- English: words
- Programming language: numbers, strings, simple operators

Syntax:

- English:
  - "cat dog boy" $\rightarrow$ not syntactically valid
  - "cat hugs boy" $\rightarrow$ syntactically valid
- Programming language:
  - "hi"5 $\rightarrow$ not syntactically valid
  - $3.2 * 5 \rightarrow$ syntactically valid

Aspects of Languages

- Semantics is the meaning associated with a syntactically correct string of symbols with no static semantic errors*.
- English: can have many meanings, such as "Flying planes can be dangerous"
- Programming languages: have only one meaning but may not be what programmer intended.

*For compiled languages, static semantics essentially include those semantic rules that can be checked at compile time. Examples include checking that every identifier is declared before it is used (in languages that require such declarations) or that the labels on the arms of a case statement are distinct.

Where things go wrong

- **syntactic errors**
  - common and easily caught
- **static semantic errors**
  - some languages check for these before running program can cause unpredictable behavior
- **no semantic errors but different meaning than what programmer intended**
  - program crashes, stops running
  - program runs forever
  - program gives an answer but different than expected

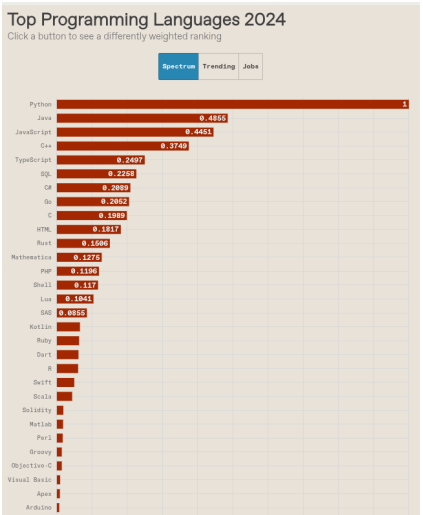
There are only two kinds of (programming) languages: the ones people complain about and the ones nobody uses.

— Bjarne Stroustrup (designer of C++)

Top Programming Languages 2024

https://spectrum.ieee.org/top-programming-languages-2024

- Python is one of the most popular programming languages worldwide.
- Python is a major tool for scientific computing, accounting for a rapidly rising share of scientific work around the globe.



Why should **YOU** program?

- Nearly all of the most exciting and important technologies, arts, and sciences of today and tomorrow are driven by computing.
- Understanding computing illuminates deep insights and questions into the nature of our minds, our culture, or even our universe.

Let's complain about...

What is Python?

- Python is a **general purpose** programming language conceived in 1989 by Dutch programmer Guido van Rossum.
- Python is **free and open source**, with development coordinated through the **Python Software Foundation** https://www.python.org/psf/).
- Python has experienced rapid adoption in the last decade, and is now one of the most popular programming languages.

Common uses

Python is a general purpose language used in almost all application domains:

- communications
- web development
- graphical user interfaces
- games, multimedia, data processing, security, etc.
- Machine Learning, Artificial Intelligence

Used extensively by Internet service and high tech companies such as

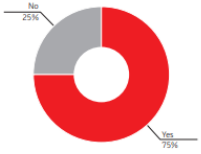
- Google
- Dropbox
- Reddit
- YouTube
- Walt Disney Animation,...

$\rightarrow$ **Often used to teach computer science and programming**

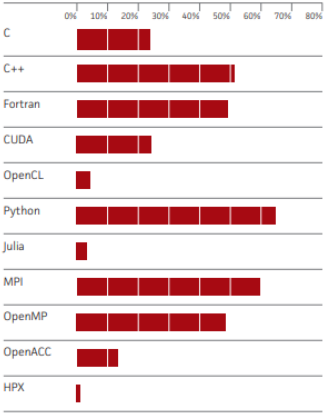
$\rightarrow$ **For reasons we will discuss, Python is particularly popular within the scientific community**

Statistics from supercomputer users

Do you develop and maintain application codes?



Which programming languages and parallelization paradigms are you using primarily?



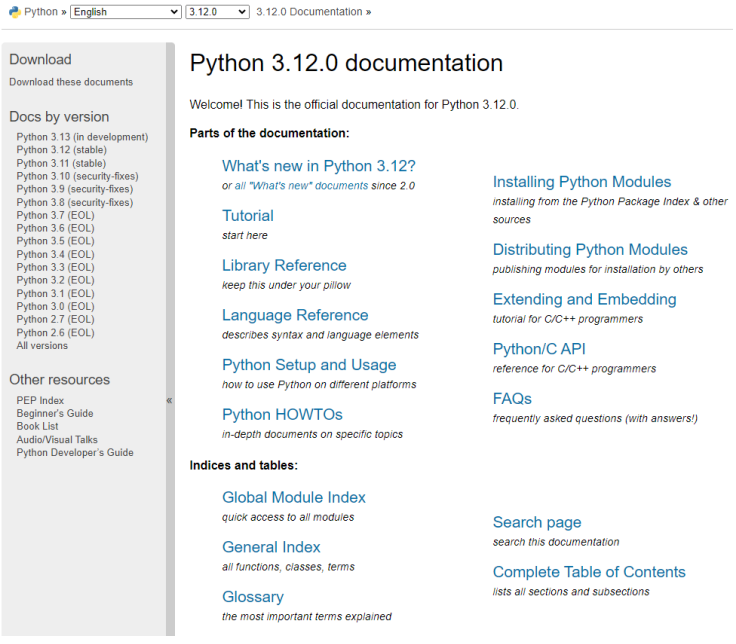
Some features

- Python is a high level language suitable for rapid development.
- It has a relatively small core language **supported by many libraries.**
- A multi-paradigm language, in that multiple programming styles are supported (procedural, object-oriented, functional,...).
- **Interpreted rather than compiled.**

Syntax and Design

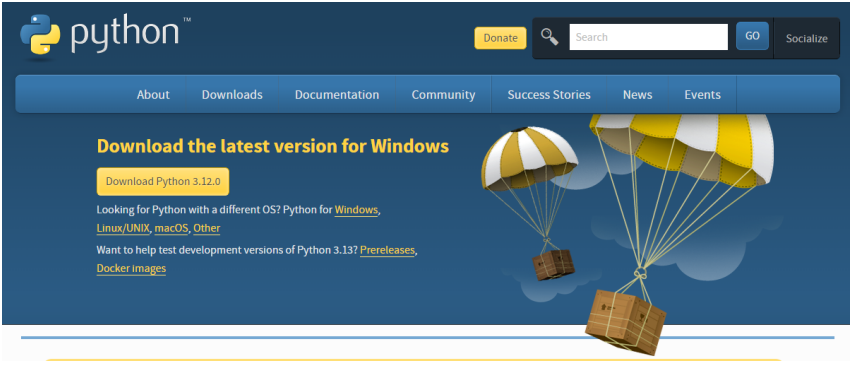
- One nice feature of Python is its **elegant syntax** — we'll see many examples later on.
- **Elegant code** might sound superfluous but in fact it's highly beneficial because it makes the syntax **easy to read and easy to remember.**
- **Remembering** how to read from files, sort dictionaries and other such routine tasks means that you don't need to break your flow in order to hunt down correct syntax.
- Closely related to **elegant syntax** is **elegant design.**
- Features like iterators, generators, list comprehensions, etc. make Python highly expressive, **allowing you to get more done with less code.**

https://docs.python.org/



Install Python

https://www.python.org/downloads/



Find the installation you need (Linux, MacOS, Windows)

Get Python

- You can download and install Python directly from https://www.python.org.
- Since we're going to use several libraries for numerical computation (numpy), data analysis (pandas), machine learning (scikit-learn), and visualization (matplotlib), it is easier to install Anaconda, which bundles all things required https://www.continuum.io/downloads.

Computational Infrastructure

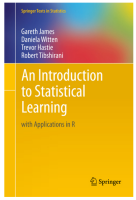
- You don't need to install Python.
- All you need is a device with a browser.
- $\rightarrow$ We use a cloud server that has all the necessary libraries and compilers installed (see the crash course later today, or the TA session).
- $\rightarrow$ We use `nuvolos.cloud` (see later today and the TA session!).

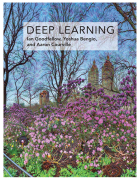
A Microsoft Azure Center

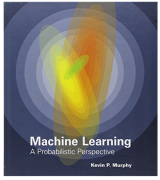
Some source material

I will not follow a textbook, but will point in my slides to the relevant literature.
Some useful textbooks:

- ***Machine Learning: a Probabilistic Perspective***
  K. Murphy, MIT Press, 2012. https://www.cs.ubc.ca/~murphyk/MLbook/index.html
- ***An Introduction to Statistical Learning***
  Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani; Springer, 8th editon, 2017.
  https://www-bcf.usc.edu/~gareth/ISL/
- ***Deep Learning***
  Ian Goodfellow and Yoshua Bengio and Aaron Courville; MIT Press 2016.
  http://www.deeplearningbook.org







Some source material

- ***Pattern Recognition and Machine Learning***
  C. M Bishop; Springer 2006. (pdf freely available)
- ***Python Machine Learning***
  S. Raschka; PACKT Publishing 2017
- ***Introduction to Machine Learning***
  Ethem Alpaydin; MIT Press 2014.
- ***A Primer on Scientific Programming with Python***
  Hans Petter Langtangen; Springer 2016.
- ***Mathematics for Machine Learning***
  Deisenroth, A. Aldo Faisal, and Cheng Soon Ong; Cambridge University Press 2020.
- ***Introduction to Computatiton and Programming using Python***
  J. Guttag. MIT Press, 2013

https://www.cs.ubc.ca/murphyk/MLbook/index.html