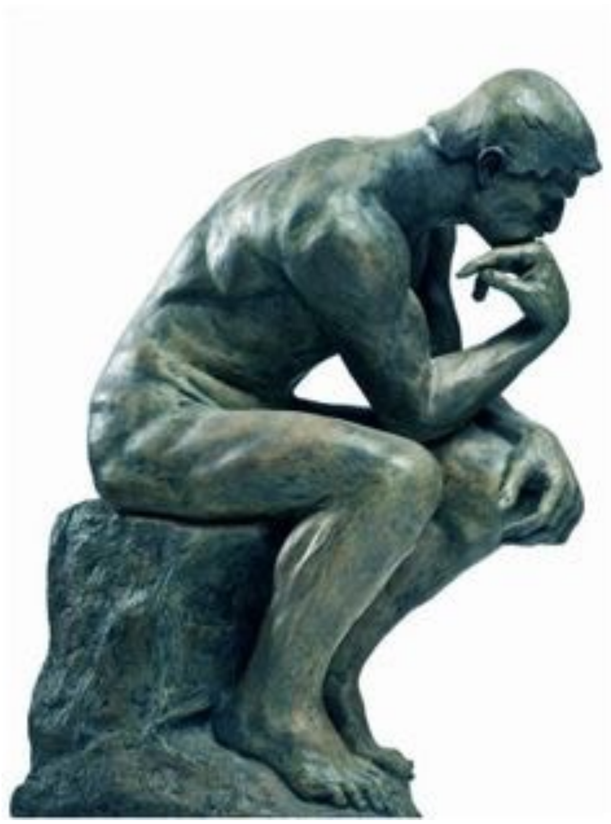
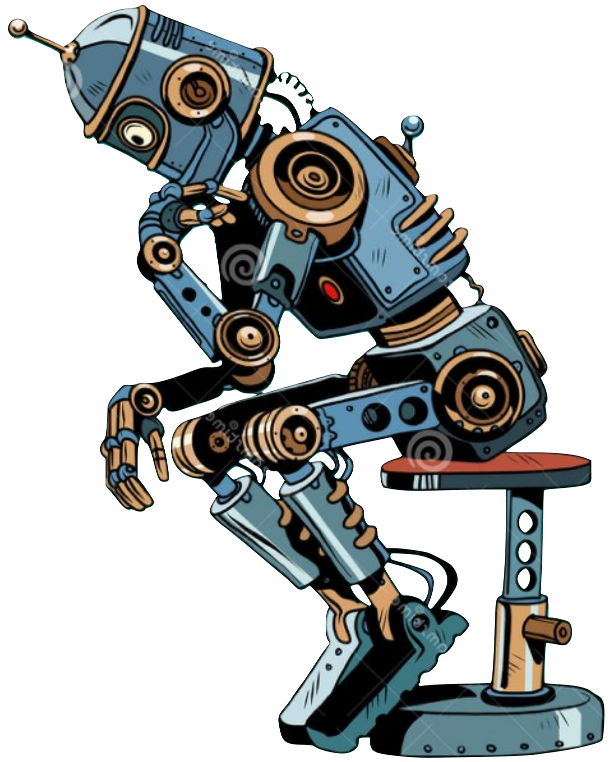
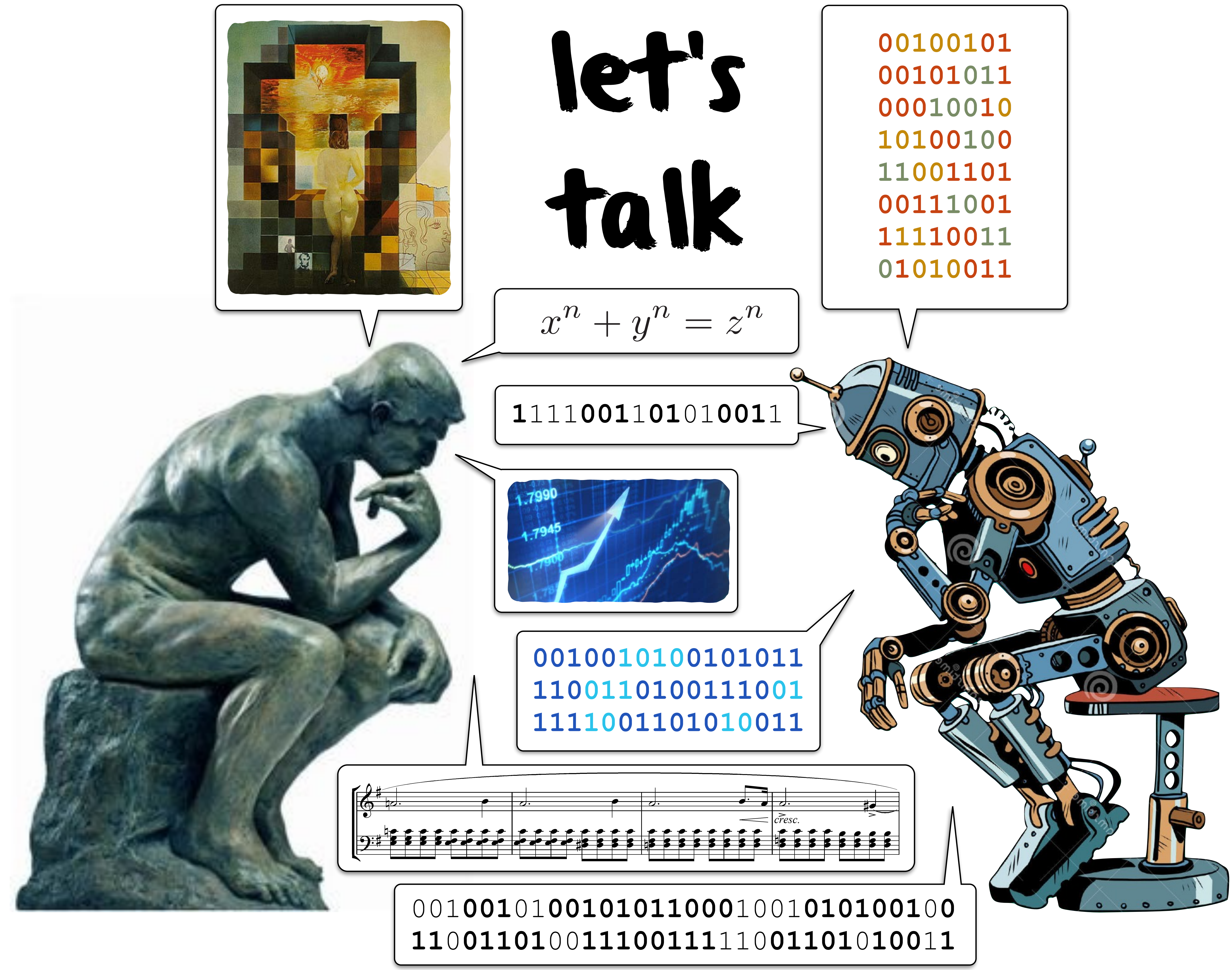
algorithms and computational thinking





course overview


let's talk
00100101
00101011
00010010
10100100
11001101
00111001
11110011
01010011
$x^n + y^n = z^n$
1111001101010011
1.7990
1.7945
0010010100101011
1100110100111001
1111001101010011
cresc.
00100101001010110001001010100100
11001101001110011111001101010011

problem

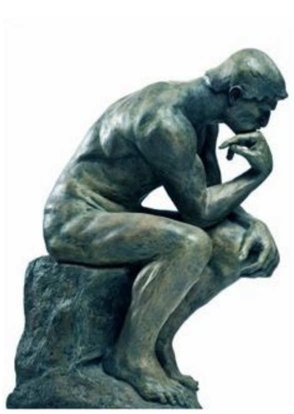
how do you tell the computer what to do, if you don't speak its language and it doesn't speak yours?


spark-1.3.1 — bash — 130×38
3ebc6a/userFiles-eadd4714-93e2-47d5-8a37-9b8d5cc7ccaf/spark-examples-1.3.1-hadoop2.4.0.jar to class loader
15/07/24 13:12:15 INFO Executor: Finished task 7.0 in stage 0.0 (TID 7). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Starting task 8.0 in stage 0.0 (TID 8, localhost, PROCESS_LOCAL, 1338 bytes)
15/07/24 13:12:15 INFO Executor: Finished task 3.0 in stage 0.0 (TID 3). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Starting task 9.0 in stage 0.0 (TID 9, localhost, PROCESS_LOCAL, 1338 bytes)
15/07/24 13:12:15 INFO Executor: Finished task 5.0 in stage 0.0 (TID 5). 736 bytes result sent to driver
15/07/24 13:12:15 INFO Executor: Running task 9.0 in stage 0.0 (TID 9)
15/07/24 13:12:15 INFO Executor: Running task 8.0 in stage 0.0 (TID 8)
15/07/24 13:12:15 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 803 ms on localhost (1/10)
15/07/24 13:12:15 INFO TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 805 ms on localhost (2/10)
15/07/24 13:12:15 INFO TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 809 ms on localhost (3/10)
15/07/24 13:12:15 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 826 ms on localhost (4/10)
15/07/24 13:12:15 INFO Executor: Finished task 4.0 in stage 0.0 (TID 4). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 4.0 in stage 0.0 (TID 4) in 818 ms on localhost (5/10)
15/07/24 13:12:15 INFO Executor: Finished task 1.0 in stage 0.0 (TID 1). 736 bytes result sent to driver
15/07/24 13:12:15 INFO Executor: Finished task 6.0 in stage 0.0 (TID 6). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 825 ms on localhost (6/10)
15/07/24 13:12:15 INFO TaskSetManager: Finished task 6.0 in stage 0.0 (TID 6) in 822 ms on localhost (7/10)
15/07/24 13:12:15 INFO Executor: Finished task 2.0 in stage 0.0 (TID 2). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 869 ms on localhost (8/10)
15/07/24 13:12:15 INFO Executor: Finished task 9.0 in stage 0.0 (TID 9). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 71 ms on localhost (9/10)
15/07/24 13:12:15 INFO Executor: Finished task 8.0 in stage 0.0 (TID 8). 736 bytes result sent to driver
15/07/24 13:12:15 INFO TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 78 ms on localhost (10/10)
15/07/24 13:12:15 INFO DAGScheduler: Stage 0 (reduce at SparkPi.scala:35) finished in 0.900 s
15/07/24 13:12:15 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/07/24 13:12:15 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:35, took 1.068825 s
Pi is roughly 3.140524
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
15/07/24 13:12:15 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
9:41 AM
Done
World Clock
Cupertino
New York
Detroit
London

approach of this course

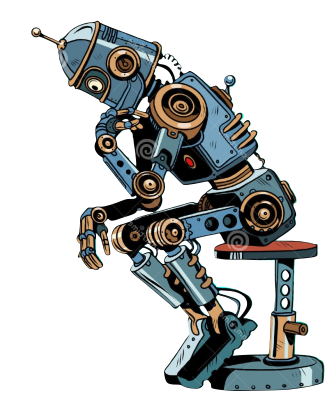
think precisely in computational terms

express problems formally and solutions as algorithms



use high-level programming languages

use tools to develop, and debug programs



computer science

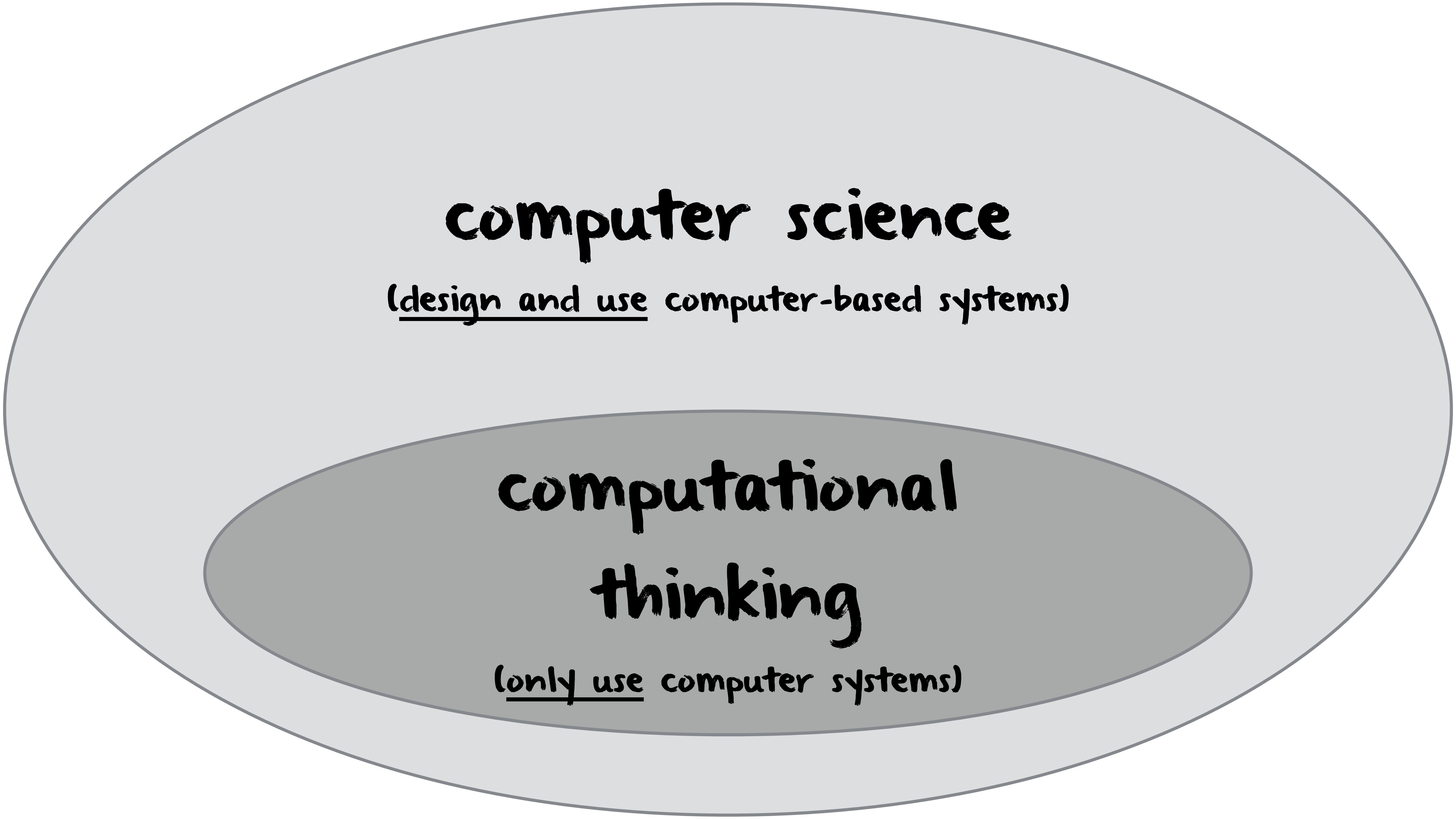
theoretical and practical study of how to design and use computer-based systems

computer science aims at devising automated algorithmic processes and computer-based systems that can run in a scalable manner

computational thinking

thought processes involved in formulating problems and expressing their solutions so that a computer can execute them

such solutions are expressed in terms of algorithms, which are in turn written in some programming language compiled and executed on some computer-based system



$$\textit{computational thinking} \subset \textit{computer science}$$

$$\textit{computer science} \not\subset \textit{computational thinking}$$

teaching staff

Benoît
Garbinato

professor

Arielle
Moro

Vaibhav
Kulkarni

assistants

Benoît Garbinato

PhD in ComputerScience

worked in the industry

Professor since 2004

Arielle
Moro

BSc in Business Computing

MSc in Information Systems

PhD student in Information Systems

Vaibhav Kulkarni

Contiki

The Open Source OS for the Internet of Things

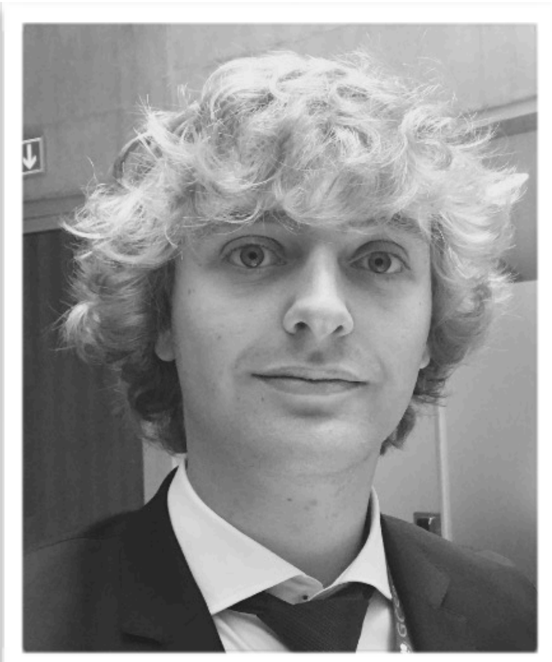
B. Eng. in Electronics & Telecommunication

MSc in Communication Technology

MSc in Embedded Systems

PhD student in Information Systems

student assistants


Adrian

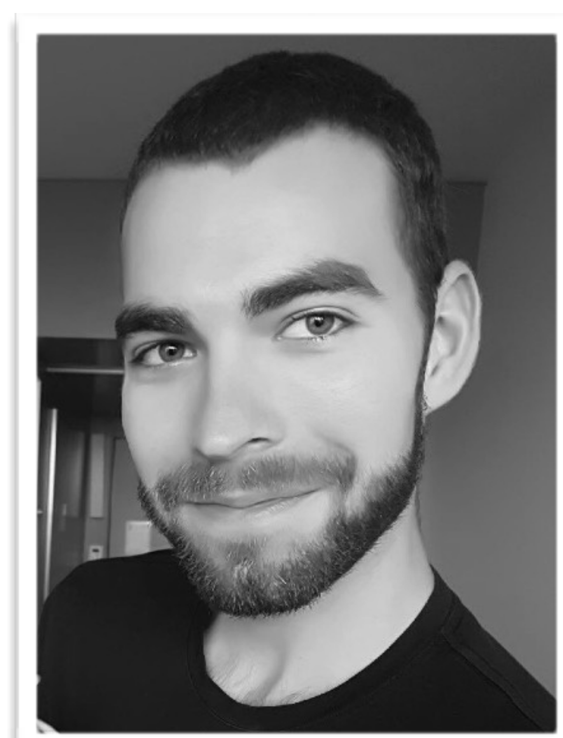

Arnaud


Francesco


Jean-Marc


Milena


Nomeny


Xavier


Yannick

course objective

learn a set of thinking skills and practical methods to formulate and solve problems using algorithms and computing devices

course objective

what's a computer?

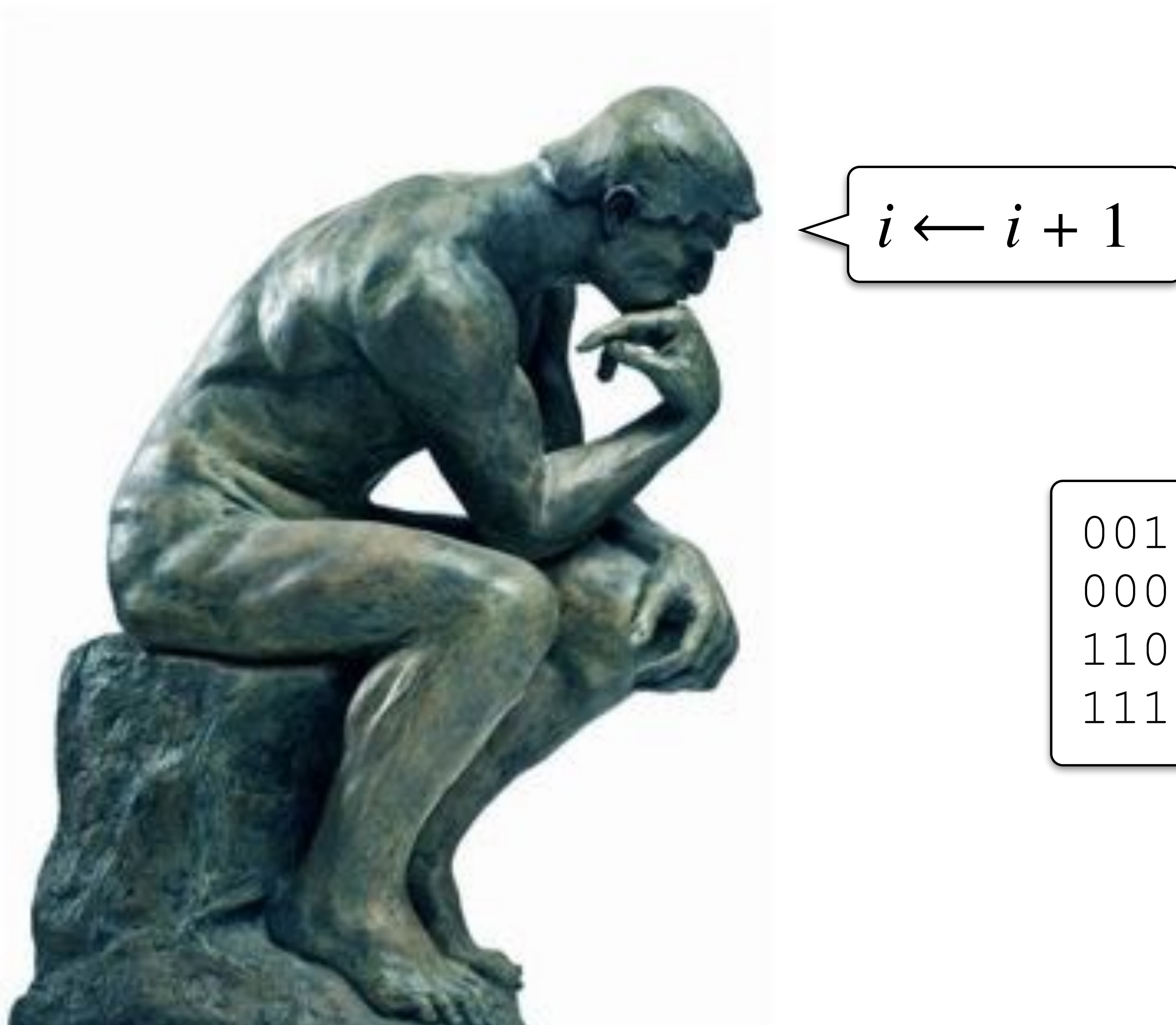
what's an algorithm?

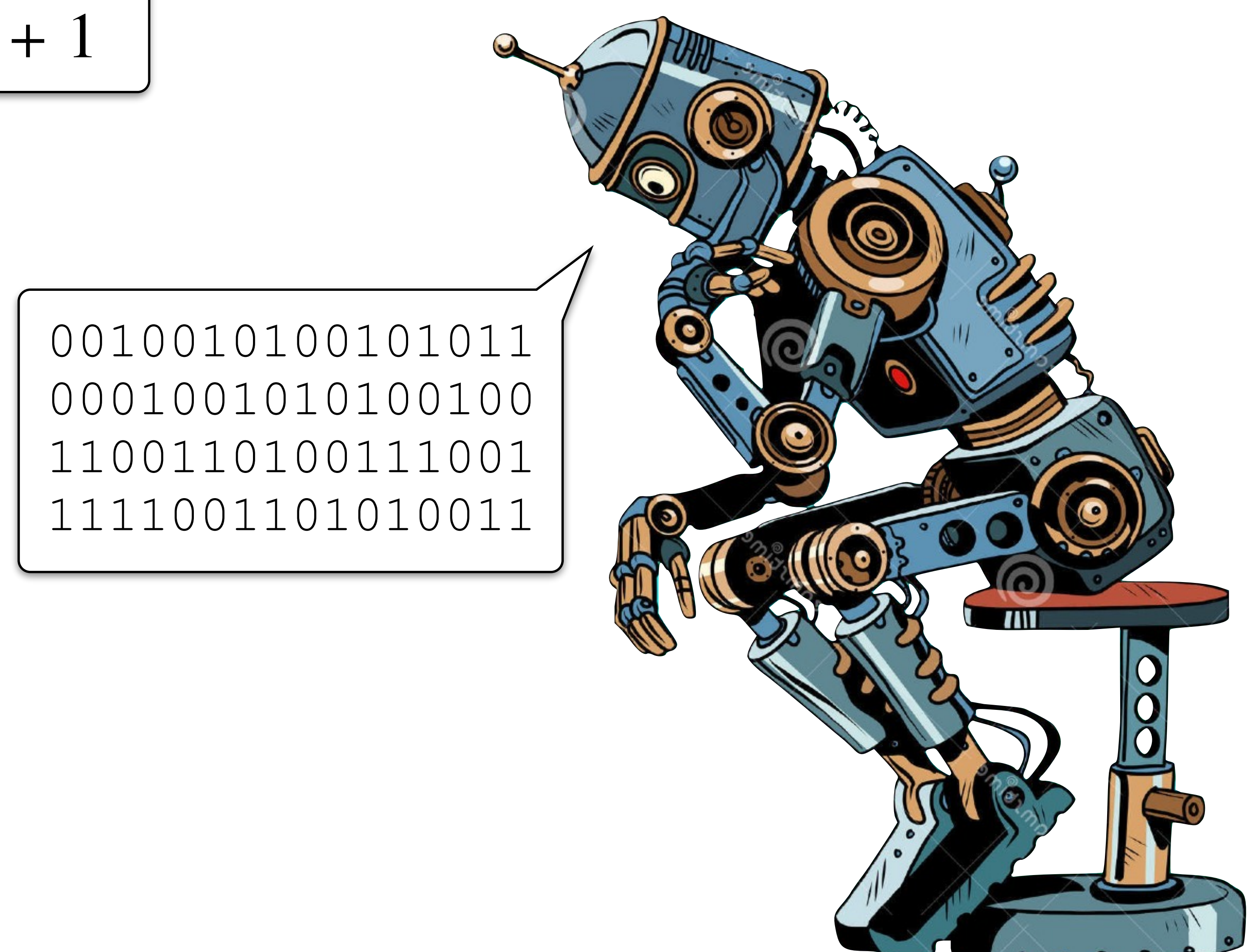
what's a compiler?

what's programming?

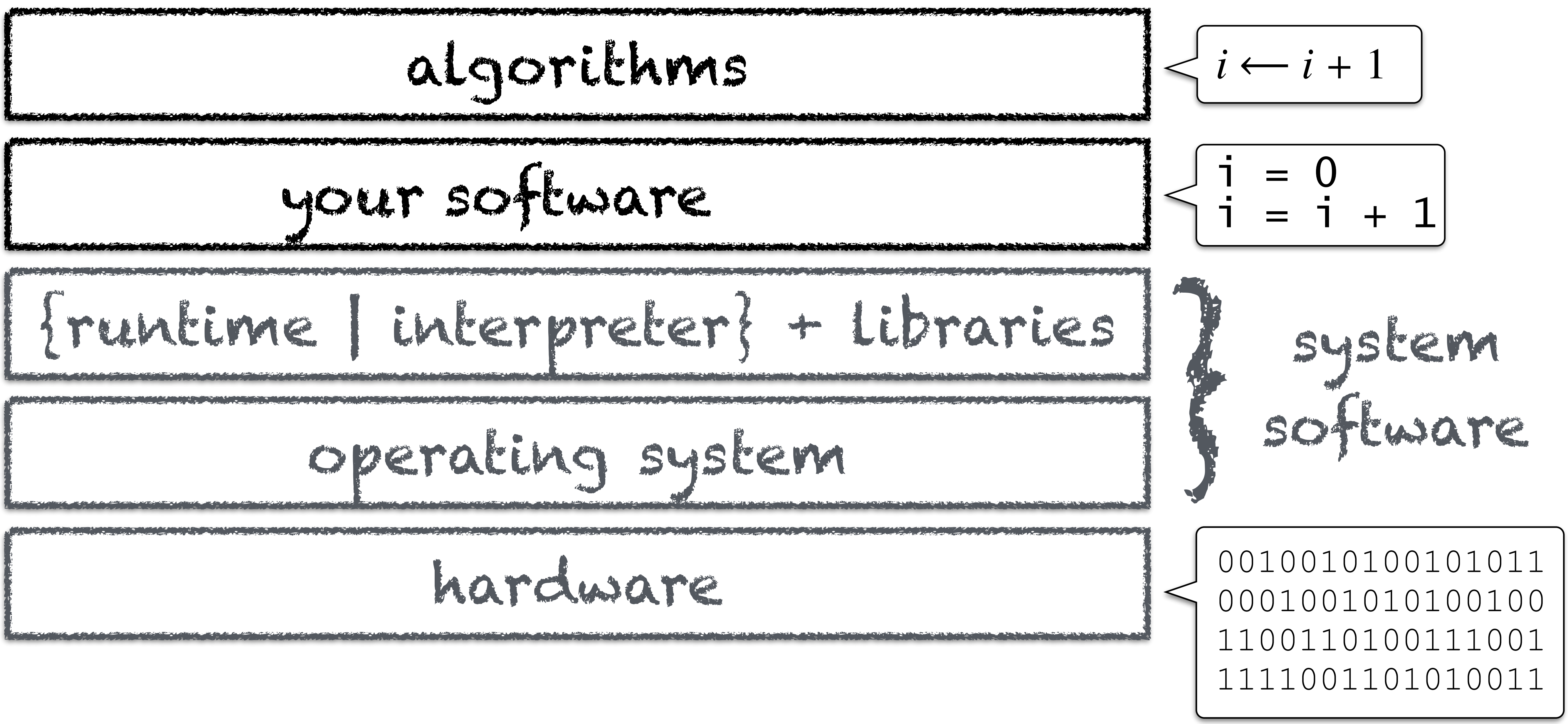
etc...

content & approach



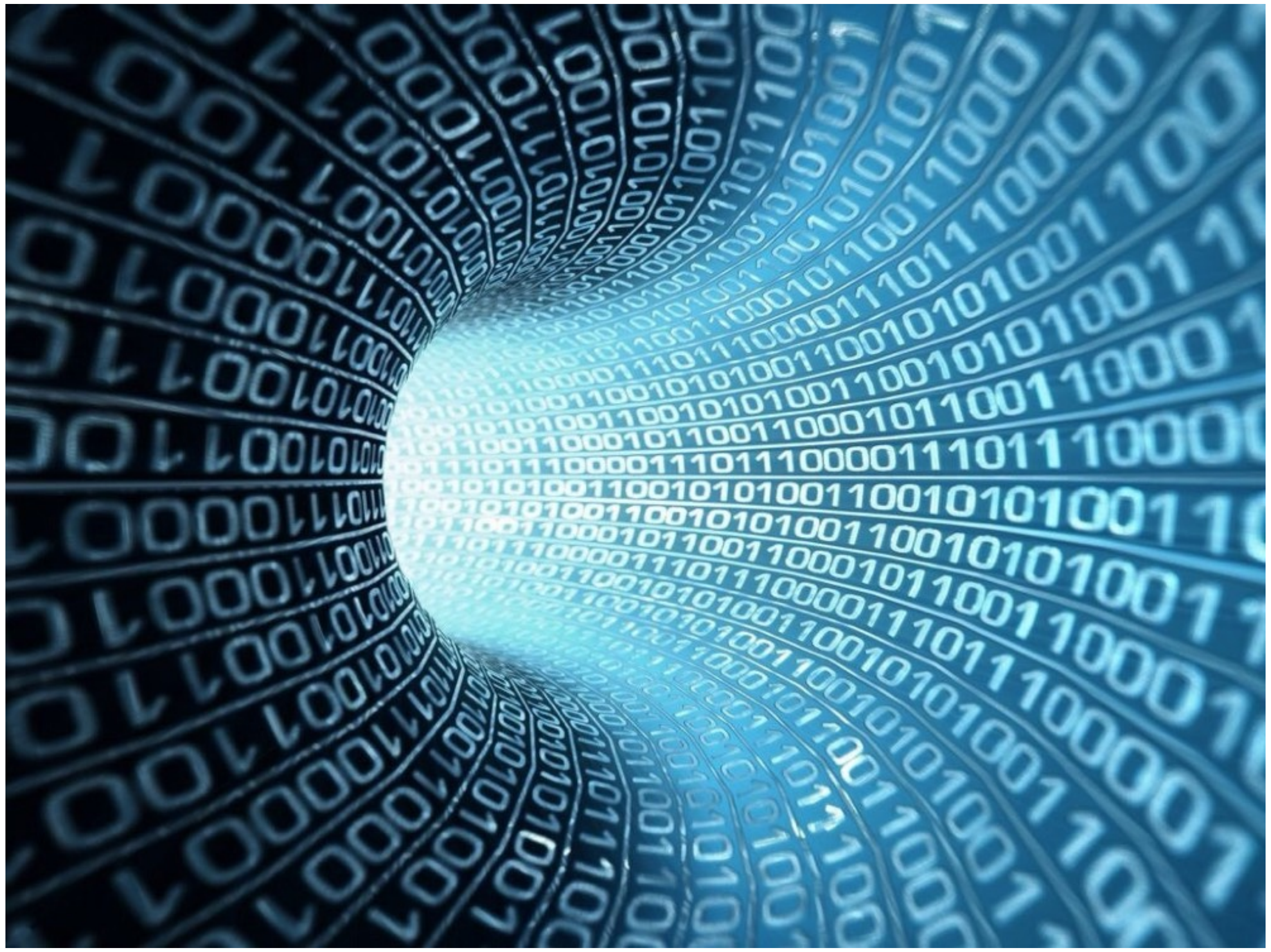


content & approach



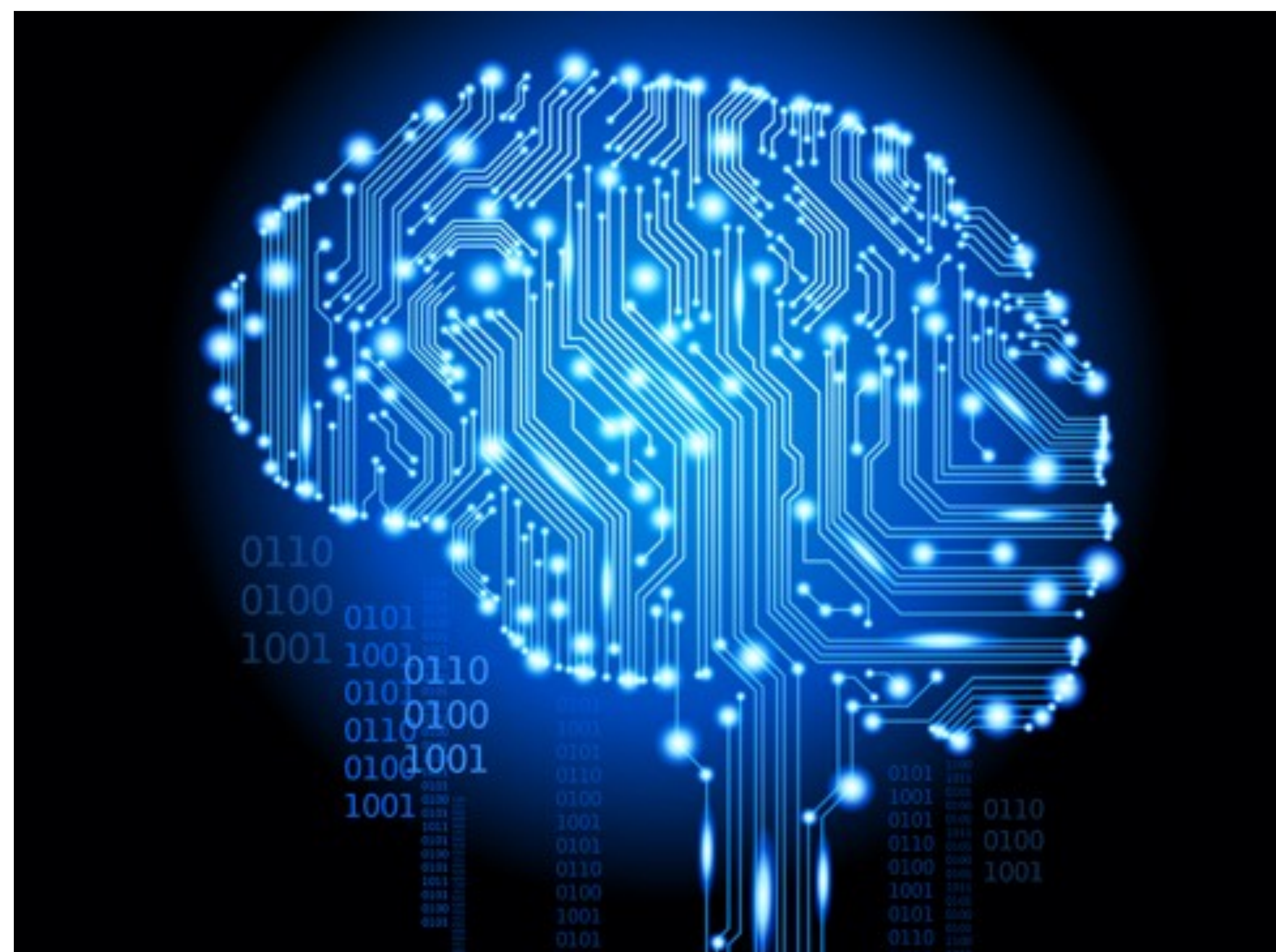
what are the benefits of this course?

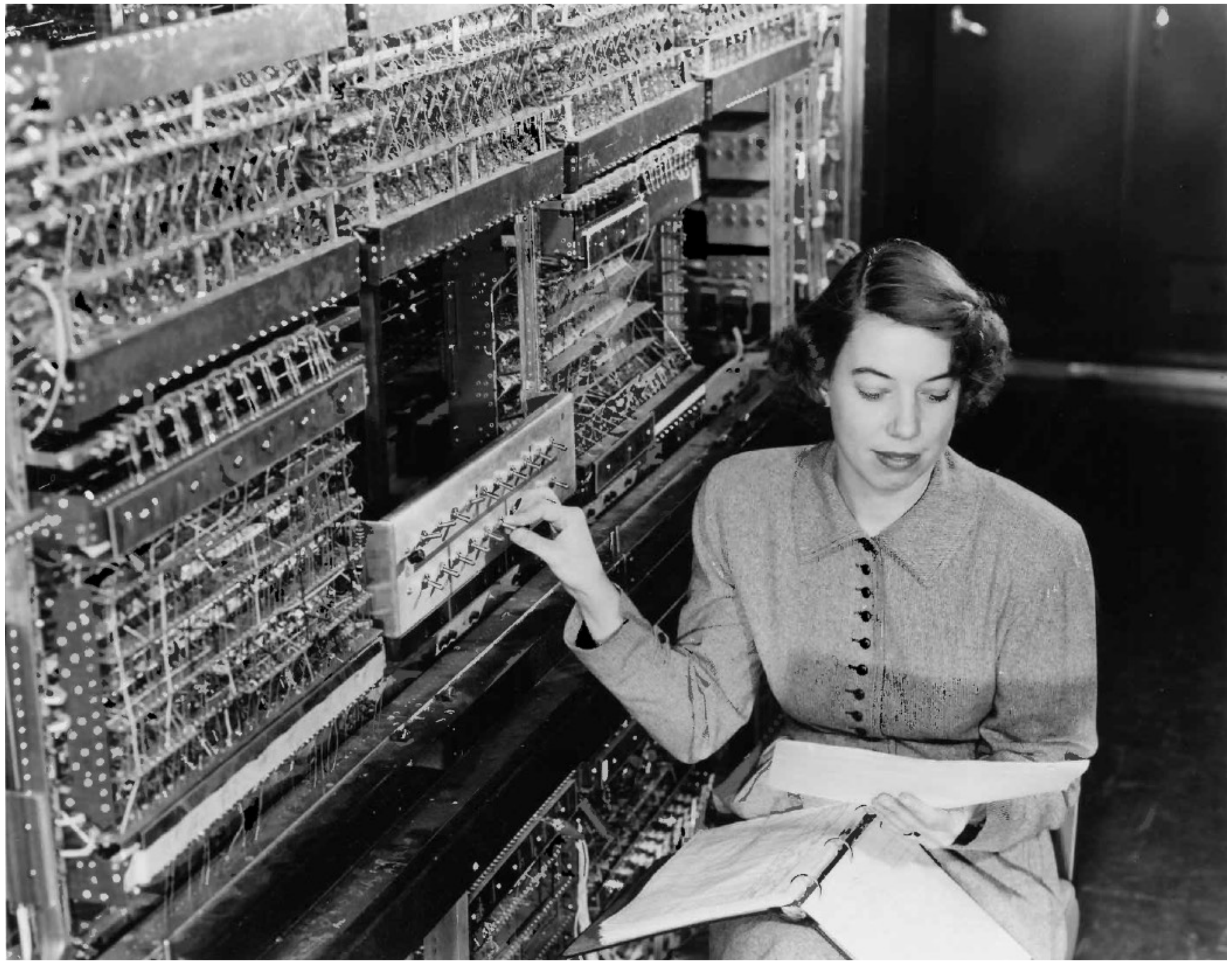
direct benefits



the ability to reason in algorithmic terms

and under computational constraints

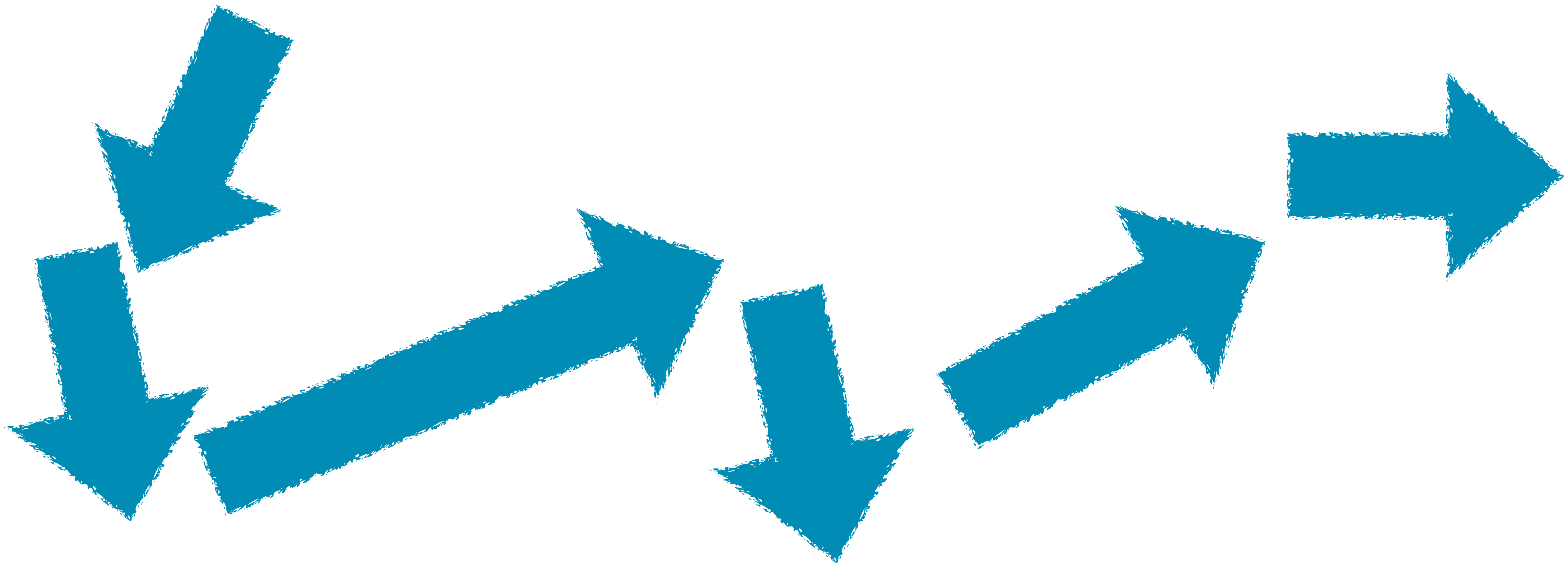




the ability to
precisely specify
various problems

and solve them by
writing computers
programs

indirect benefits




learn to think differently
learn to think creatively
experiment the design attitude
learn to navigate through different levels of abstraction
be ready for the digital transformation

decision attitude

assumes that the alternative courses of action are ready at hand, including the best one

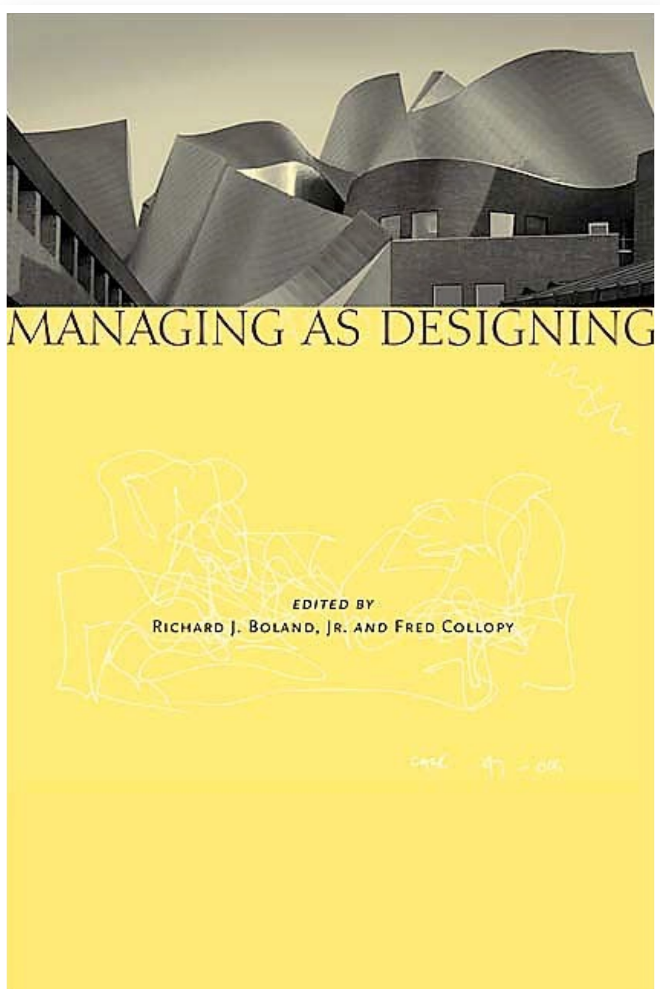
passive view of the decision maker as a problem solver

design attitude

a design attitude views each project as an opportunity for invention that includes a questioning of basic assumptions

designers relish the lack of predetermined outcomes

managing as designing

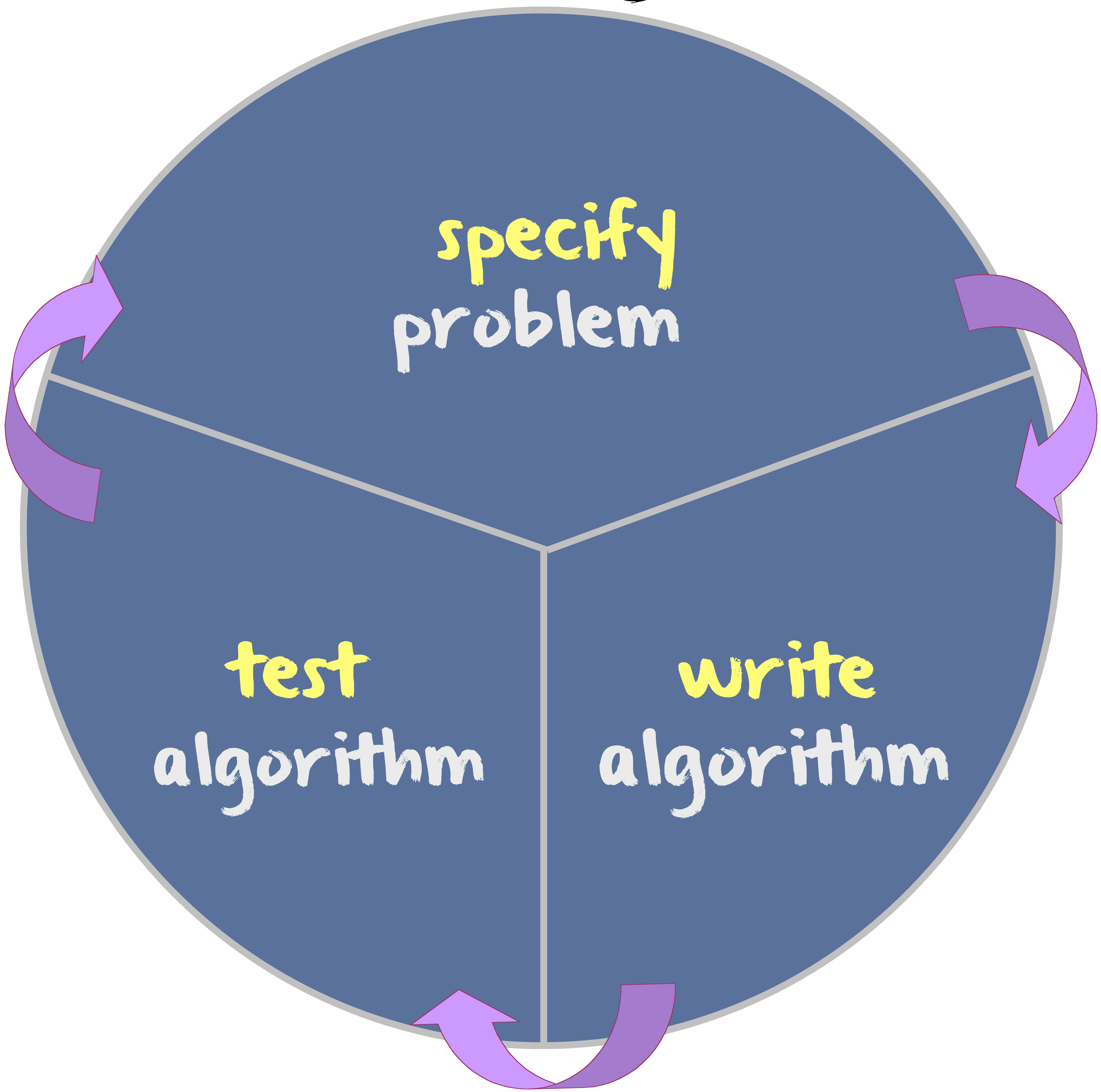


Managing as Designing
R. Boland, F. Collopy
Stanford Press

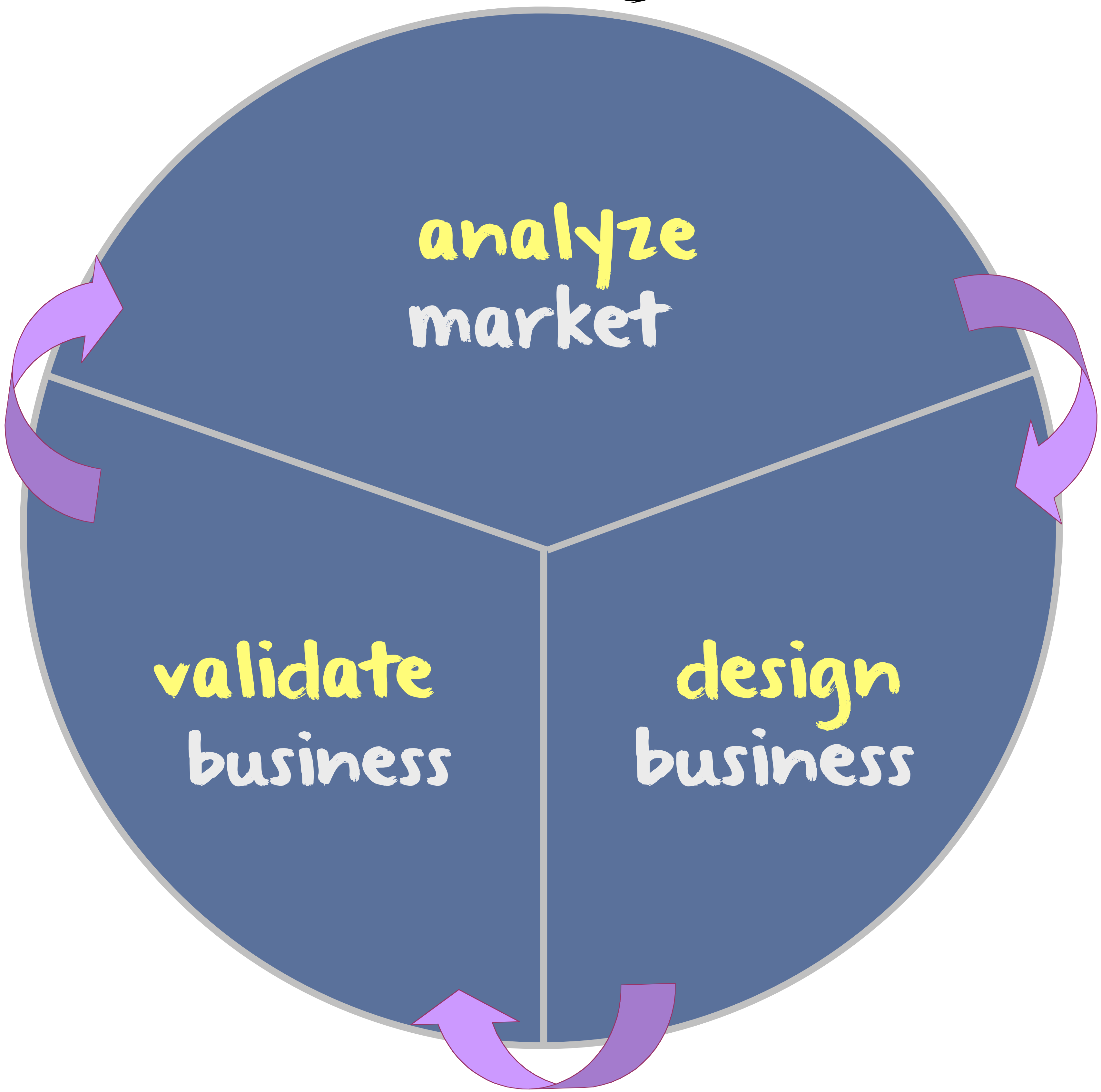
managers should act not only as decision makers, but also as designers

though decision and design are linked in management action, managers and scholars have emphasized the decision face over the design face.

typical design cycle



typical design cycle



think abstractions

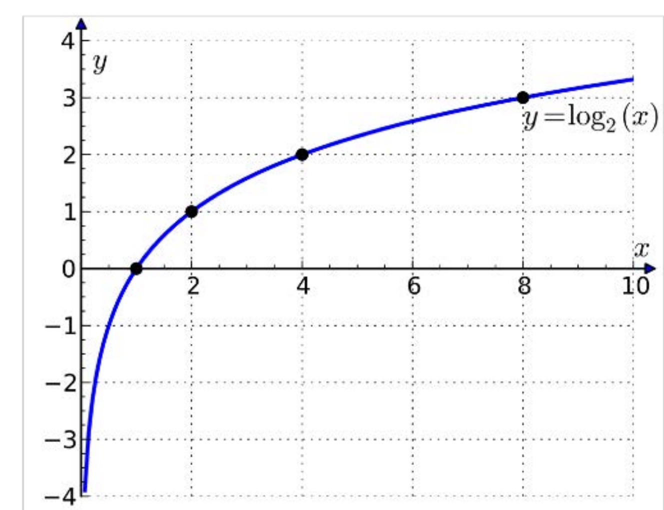
an abstraction is a set of common properties and laws extracted from several particular examples

examples:

$$\sum \vec{F} = m\vec{a}$$

Mutationem motus proportionalem esse vi motrici impressae, et fieri secundum lineam rectam qua vis illa imprimitur

The alteration of motion is ever proportional to the motive force impressed, and is made in the direction of the right line in which that force is impressed



mammals

$$f(x, y) = \sqrt{x^2 + y^2}$$

sphere

thinking in abstractions is one of the key traits in human being

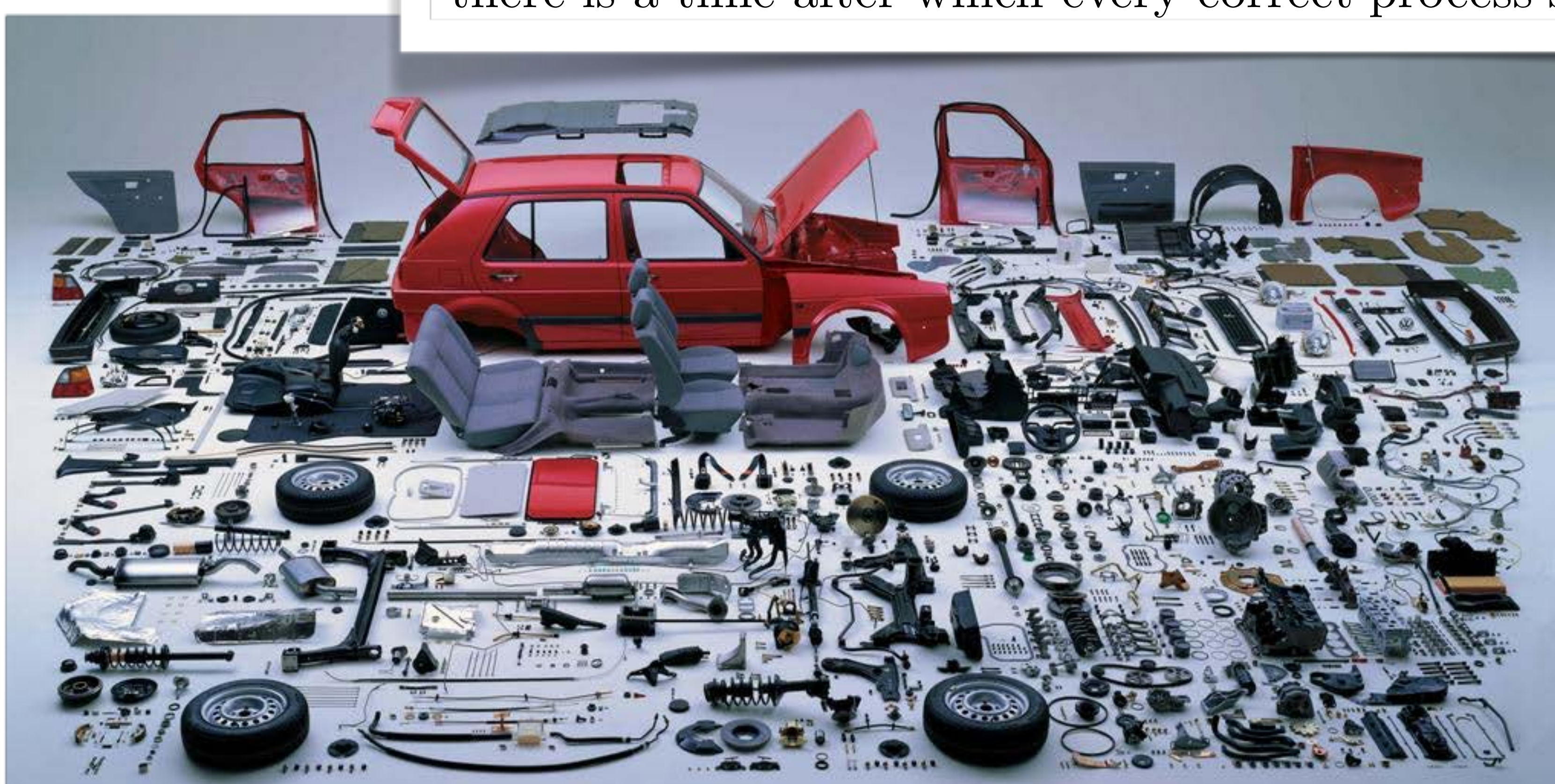
stacking abstractions

THEOREM 1. *$\mathcal{Q} \succeq \mathcal{P}$, $\mathcal{W} \succeq \mathcal{S}$, $\diamond\mathcal{Q} \succeq \diamond\mathcal{P}$, and $\diamond\mathcal{W} \succeq \diamond\mathcal{S}$.*

PROOF. Let $\mathcal{D}$ be any failure detector in $\mathcal{Q}$, $\mathcal{W}$, $\diamond\mathcal{Q}$, or $\diamond\mathcal{W}$. We show that $T_{\mathcal{D}\rightarrow\mathcal{D}'}$ transforms $\mathcal{D}$ into a failure detector $\mathcal{D}'$ in $\mathcal{P}$, $\mathcal{S}$, $\diamond\mathcal{P}$, or $\diamond\mathcal{S}$, respectively. Since $\mathcal{D}$ satisfies weak completeness, by Lemma 1, $\mathcal{D}'$ satisfies strong completeness.

LEMMA 1. *$T_{\mathcal{D}\rightarrow\mathcal{D}'}$ satisfies P1.*

PROOF. Let p be any process that crashes. Suppose that there is a time t after which some correct process q permanently suspects p in $H_{\mathcal{D}}$. We must show that there is a time after which every correct process suspects p in $output^R$.



- algorithms
- your software
- system software
- operating system
- hardware

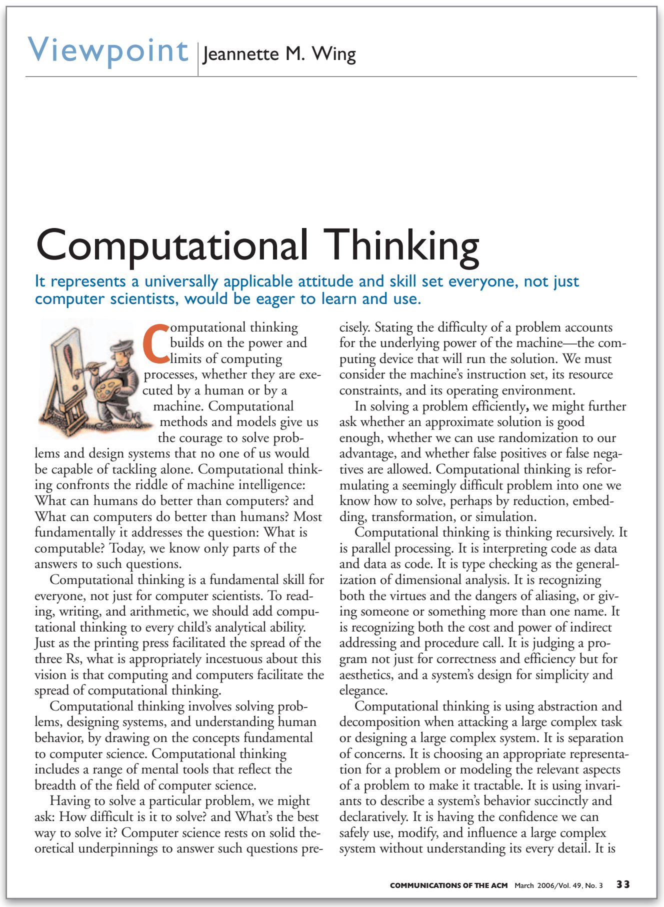
get ready for the digital transformation

digital transformation is the accelerating and profound transformation of all aspects of human society, including communication, business, learning, entertainment, etc., by the means of digital technologies

indeed, digital technologies are not longer being simply used as support for existing human activities but rather becoming the driver of profound changes in the way we do things and even the source of totally new activities

to be part of that movement, you have to understand the potential of digital technologies and learn to think algorithmically and computationally

books & papers


Viewpoint | Jeannette M. Wing

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

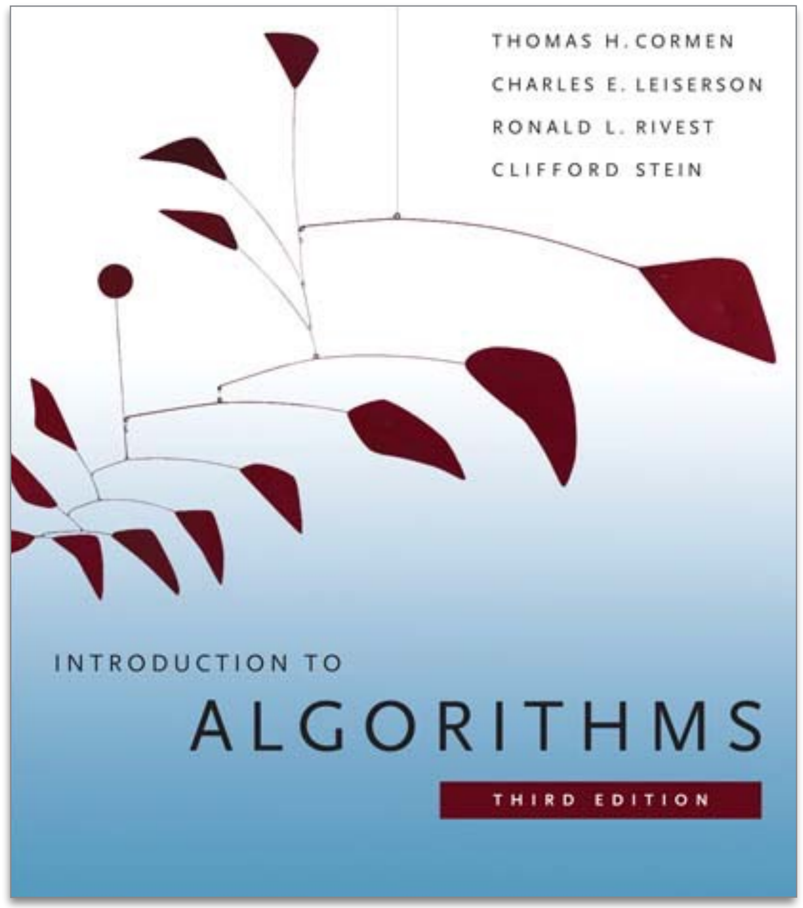
In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

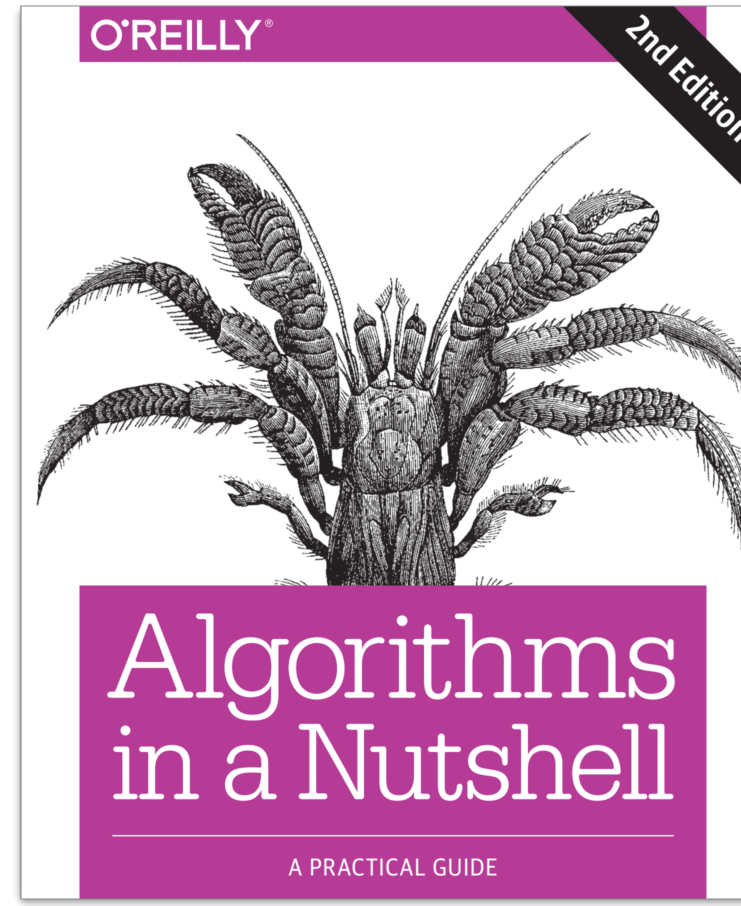
Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

COMMUNICATIONS OF THE ACM March 2006/Vol. 49, No. 3 33

Computational thinking. J.M. Wing. Communication of the ACM, 49(3):33–35, March 2006.



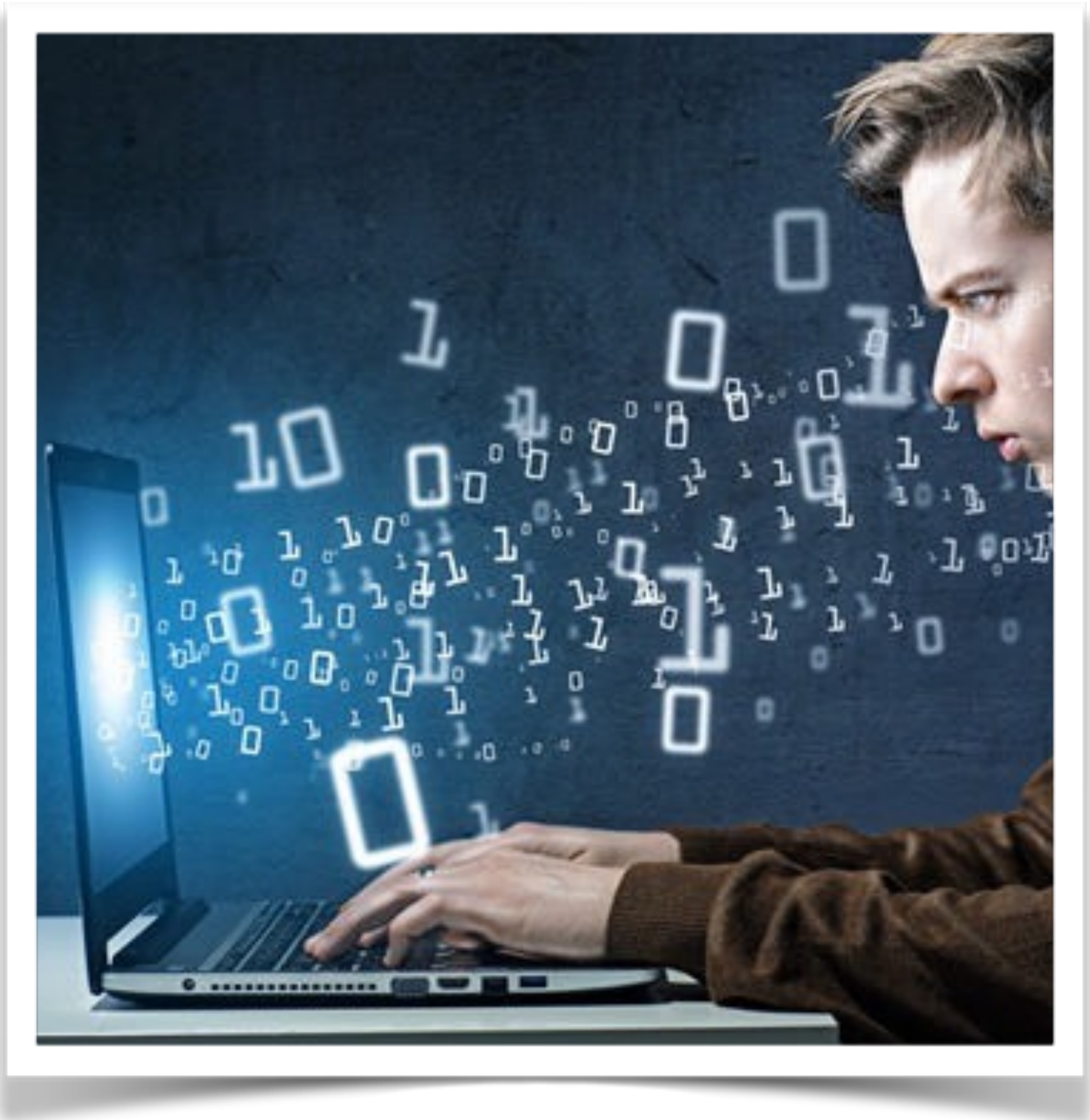
Introduction to Algorithms, 3rd Edition. T.H.Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. July 2009. MIT Press.



Algorithms in a Nutshell, 2nd Edition. G.T. Heineman, G. Pollice, S. Selkow. March 2016. O'Reilly.

and more to come during our journey together...

overview - week 1



computer architecture

overview - week 2

system software

overview - week 3

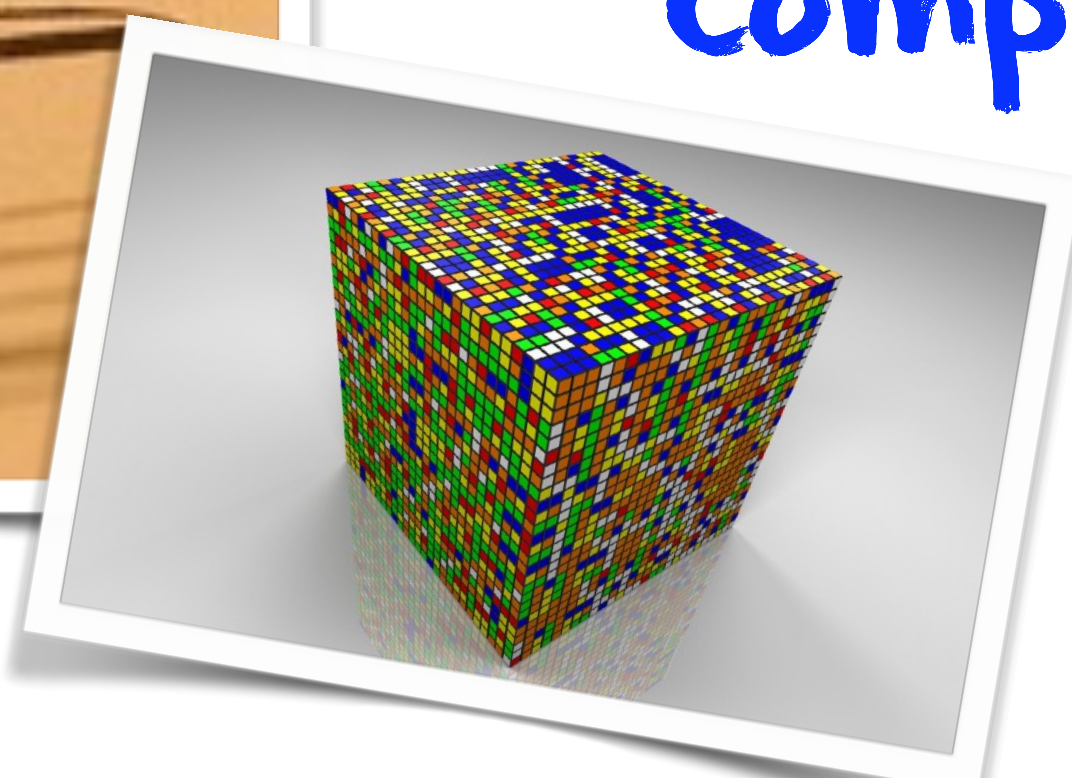
programming basics

overview - week 4

induction & recursion

overview - week 5

algorithms & computational complexity



overview - week 6

mid-term test

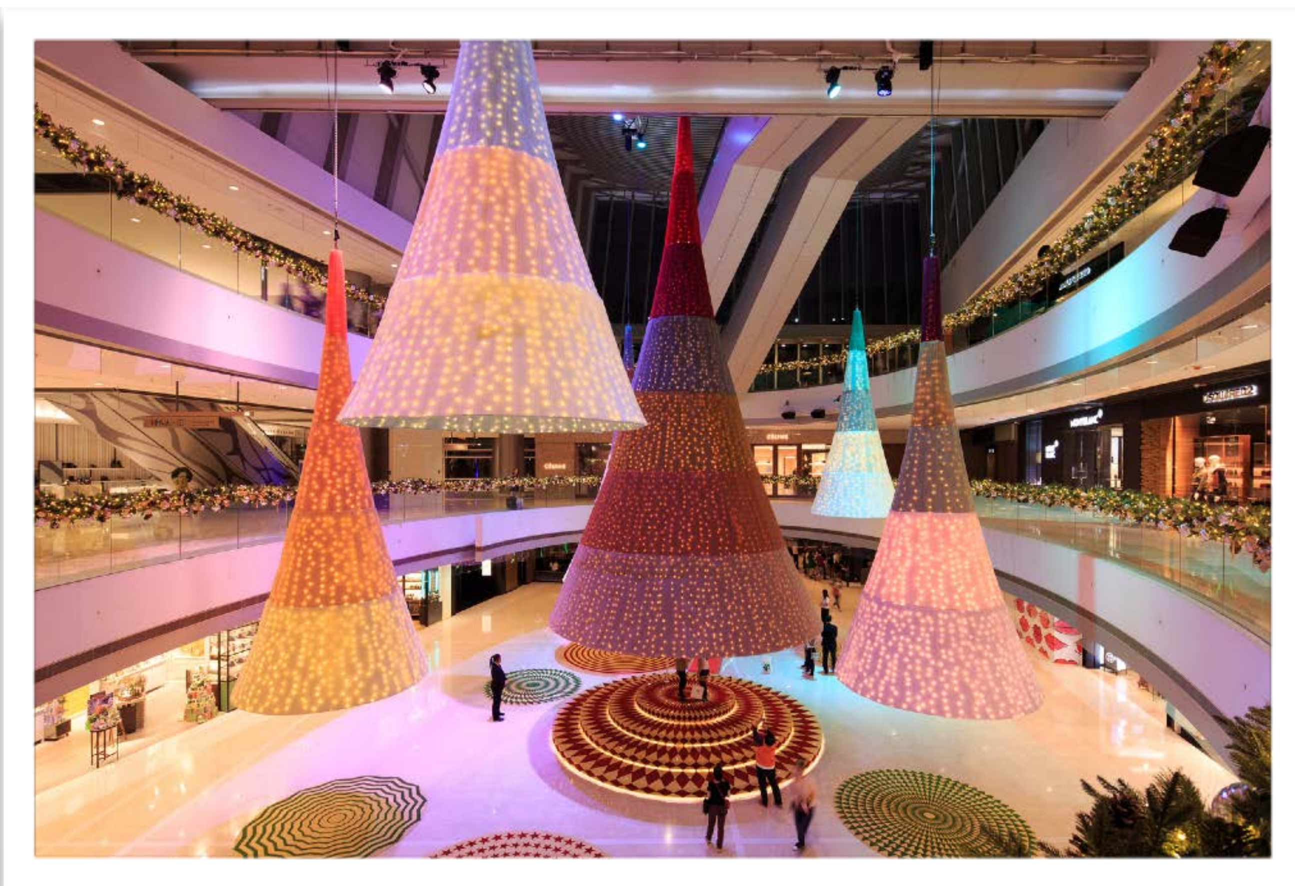
overview - week 7

searching algorithms

overview - week 8

graph algorithms

overview - week 9



spatial tree algorithms

overview - week 10

probabilistic algorithms

overview - week 11

classes,

objects &

interfaces

overview - week 12

inheritance &

polymorphism

overview - week 13

abstract classes & types

overview - week 14

functional programming

programming languages

python

scala

swift

(Java)

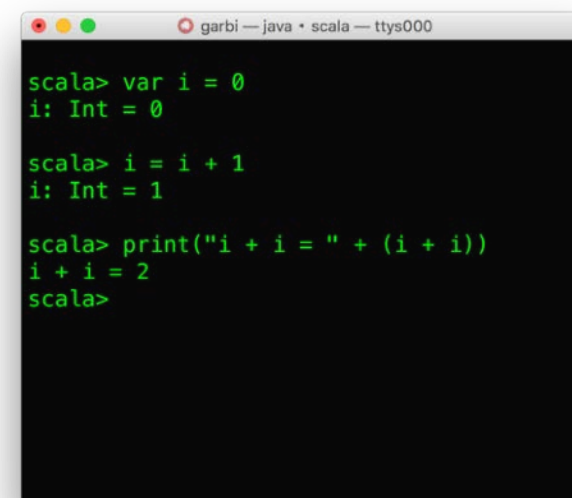
development tools

IntelliJ

(scala and python)

XCode

(swift)



the good old terminal

Calendar

theory

8:00-10:00

TUESDAY			WEEK
Sep 19	course overview	computer architecture	1
Sep 26	system software		2
Oct 03	basic programming		3
Oct 10	iteration and recursion		4
Oct 17	algorithms and computational complexity		5
Oct 24	mid-term test		6
Oct 31	searching algorithms		7
Nov 07	graph algorithms		8
Nov 14	spatial tree algorithms		9
Nov 21	probabilistic algorithms		10
Nov 28	classes, objects and interfaces		11
Dec 05	inheritance and polymorphism		12
Dec 12	abstract classes and types		13
Dec 19	functional programming		14

doplab.unil.ch/act

Calendar

practice

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:00 - 9:00		ACT - Theory ALL		ACT - Practice HEC	
9:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00			ACT - Practice ESC - Group B		
14:00 - 15:00		ACT - Practice ESC - Group A			
15:00 - 16:00			ACT - Practice ESC - Group A		
16:00 - 17:00		ACT - Practice ESC - Group B			
17:00 - 18:00					
18:00 - 19:00					

doplab.unil.ch/act

practical issues

hec students

lectures: Amphimax 351

exercises: Internef 143

if you consider taking this class register as soon as possible via the following webpage:

http://bit.ly/2cqPvDf

practical issues

esc students

lectures: Amphimax 351

exercises: Amphipôle 140 + 146

you don't get to choose whether you want to attend this course

😉

evaluation

the evaluation is based on

- an intermediate **test** during the semester
- a final **exam*** during the exam session

$$\text{grade} = 0.4 \times \textbf{test} + 0.6 \times \textbf{exam}$$

*the final exam is written in the regular session and oral in the retake session

warning

this course is given for
the first time...

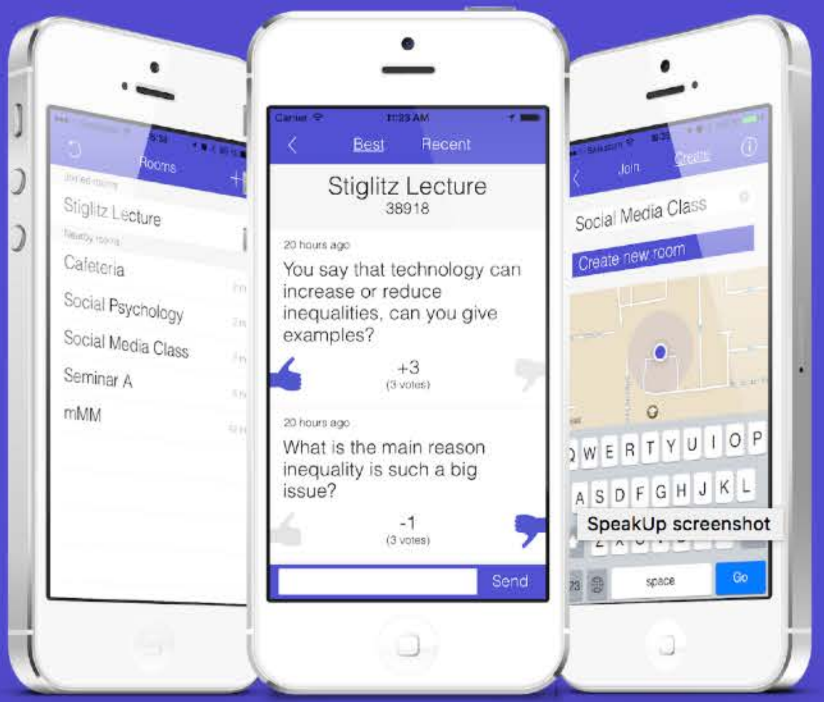
...in such a large audience

good news you will be able to say "I was among the first students to learn about computational thinking"

bad news you will serve as our guinea pigs 

SpeakUp

Let your audience anonymously share and rate each other's questions to answer the best ones.



Available on the iPhone App Store

ANDROID APP ON Google play

Available on the Web

http://speakup.info