ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI
INFORMATICA - SCIENZA E INGEGNERIA

Teaching programming in the age of generative AI

Simone Martini

Dipartimento di Informatica-Scienza e Ingegneria

Future visions of computing and programming, then and now
Lille, June 12, 2023

Programming

Programming is the essence of computing/informatics. Indeed, computing is much more than programming, but programming [. . .] is essential to computing.

[Caspersen, Teaching Programming. In Computer Science Education, Bloomsbury Academic. 2018]

Programming

Programming was considered by many to be a uniquely intellectual activity, a black art that relied on individual ability and idiosyncratic style. [. . .]

By the early 1960s, the "problem of programming" had eclipsed all other aspects of commercial computer development.

[Ensmenger, The Computer Boys Take Over, MIT Press, 2010, p. 29]

Programming

Through programming:

- algorithms
- languages
- methods: modeling and simulation, analysis and evaluation
- processes: abstraction, problem decomposition, modularization
- even systems and networks

Programming languages

From the end of the 50s

- Tool of the trade
- Object of study
- Meta-language

Our programming languages are also (a huge part of) the metalanguage in which we express the discipline.

“Programming” languages

What we insist in calling programming languages

Are powerful tools to organize, make coherent, and model reality

- data models
- procedural models
- interaction models
- synchronization models
- organization models
- ...

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

An informal journey

With two fellow travellers, of almost the same age

Then and Now

Then: Automatic programming

Saul Gorn, 1912-1992

1940s: Saul Gorn

Is Automatic Programming Feasible?

A (formerly) classified paper for the Aberdeen Proving Ground

According to D.L. Parnas:

The automatic programming system was an assembler in today's terminology.
[One would write a symbolic] code and the computer would automatically punch the proper holes in the tape.

[Parnas, Sw aspects of strategic defence systems, CACM 28(12), 1985]

Corrado Böhm, 1923-2017

1952: Corrado Böhm

La codification automatique

La codification se fait pendant deux phases consécutives :
- le programme est placé sur l'entrée de la machine ; la calculatrice exécute une série de calculs qui fournissent le même programme, mais enregistré sous forme d'instructions codifiées, propres à être interprétées automatiquement ;
- la calculatrice réalise les calculs, d'après les instructions codifiées.

[Böhm, Calculatrices digitales. PhD thesis, ETH Zürich, 1952]

Alick Glennie, 1925-2003

1952: Alick Glennie

Autocode for Manchester Mark I

```
1    c@VA t@IC x@½C y@RC z@NC
2    INTEGERS +5 →c
3          →t
4       +t      TESTA Z
5       -t
6               ENTRY Z
7    SUBROUTINE 6 →z
8      +tt →y →x
9      +tx →y →x
10   +z+cx    CLOSE WRITE 1

11   a@½ b@MA c@GA d@OA e@PA f@HA i@VE x@ME
12   INTEGERS +20 →b +10 →c +400 →d +999 →e +1 →f
13   LOOP 10n
14      n →x
15   +b-x →x
16      x →q
17   SUBROUTINE 5 →aq
18   REPEAT n
19     +c →i
20   LOOP 10n
21    +an SUBROUTINE 1 →y
22    +d-y TESTA Z
```

1988: Rich and Waters

We will be saying much the same thing about automatic programming in 1998 that we said in 1958:
that it has improved programmer productivity dramatically and has further reduced the distinction between programmers and end users.

[Rich and Waters, Automatic Programming: Myths and Prospects, IEEE Computer 21(8), 1988]

Automatic programming in-the-large

2005: Generative Programming

GP is an attempt to manufacture software components in an automated way by developing programs that synthesize other programs.

[Cointe, Towards Generative Programming, Unconventional Programming Paradigms, 2005]

David Parnas, 1941-

Summarizing: David Parnas

Automatic programming always has been a euphemism for programming with a higher-level language than was then available to the programmer.
Research in automatic programming is simply research in the implementation of higher-level programming languages.

[Parnas, Sw aspects of strategic defence systems, CACM 28(12), 1985]

A brief intermezzo

Two mantras. . .

A first mantra: in CS Education

As the level of abstraction in computing education [. . .] steadily arose, the credo among computing cognoscenti became that one needs to be familiar with at least one abstraction level below that at which one is working.

[Tedre et al., Teaching Machine Learning in K–12 Classroom, IEEE Access, vol. 9, 2021]

1950 underlying electronics

1960 octal machine code for assembly

1970 assembly language for high-level languages

19xx data structures for highly optimized class libraries

A first mantra: in CS Education

As the level of abstraction in computing education [. . .] steadily arose, the credo among computing cognoscenti became that one needs to be familiar with at least one abstraction level below that at which one is working.

[Tedre et al., Teaching Machine Learning in K–12 Classroom, IEEE Access, vol. 9, 2021]

1950 underlying electronics

1960 octal machine code for assembly

1970 assembly language for high-level languages

19xx data structures for highly optimized class libraries

A second mantra: in Programming Languages

Programs are not text; they are hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint.

[Teitelbaum, The Cornell Program Synthesizer. CACM. 24(9), 1981]

My “ontological” interpretation

Programs are not text; they are hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint.

There is no programming, if they are not created, understood, and modified under this viewpoint.

Sed contra

- Phrases as trees has been the mantra of NLP for decades
- And it didn't work

However

- Meaning in NL is fluid and multidimensional
- String corpora are always up-to-date and usable,
  as opposed as NL semantic trees, that need to be updated
- Hybrid models: graph neural network,
  neuro-symbolic architectures, knowledge integration

Then and Now

Now: Generative AI produces (good?) code

Should we still teach programmming?
Should we teach machine learning?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ChatGPT as a problem solver

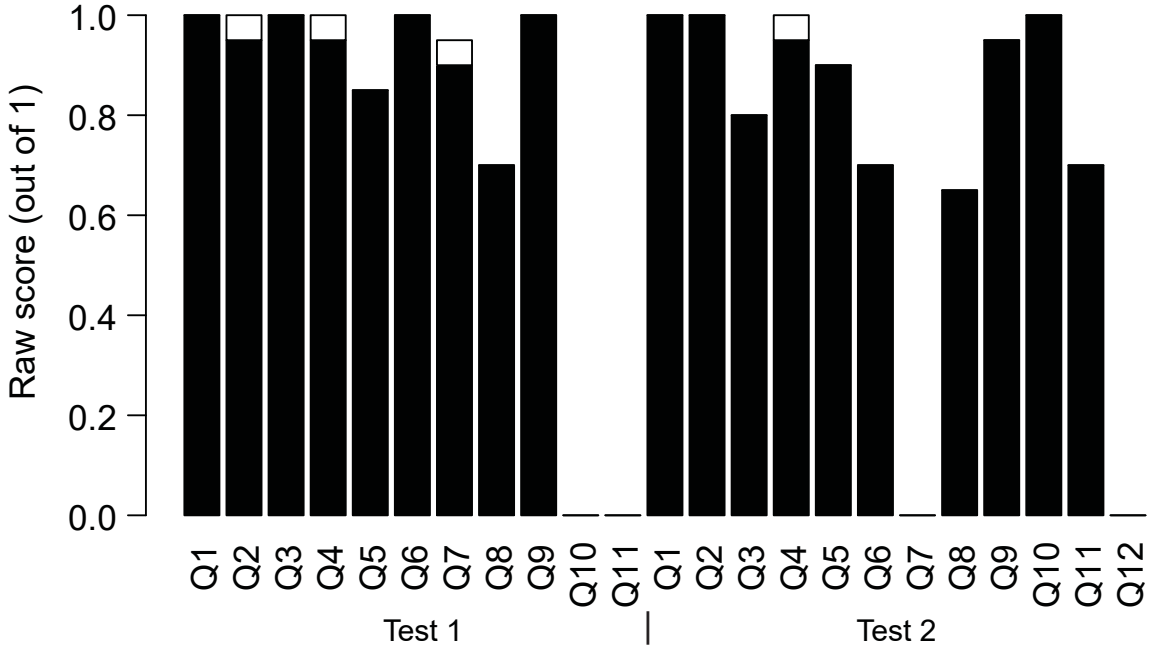
Elementary problems: DaVinci on Bebras

- correct 1/3 of the times; a random solver: 1/4 of the times; the poor performance confirms previous results
- erratic answers; there is no guarantee that they are correct
- better suited for tasks that describe a procedure
  synthesis tasks are harder
- remarkable linguistic fluency
  scrutiny needed to recognize inconsistencies, errors, gaps, that occur frequently

[Bellettini et al., DaVinci goes to Bebras, CSEDU, 2023]

OpenAI Codex as a programmer

Test problems for CS1



[Finnie-Ansley et al. The Robots Are Coming: Exploring. . . . ACE '22, 2022.]

OpenAI Codex as a programmer

Test problems for CS1

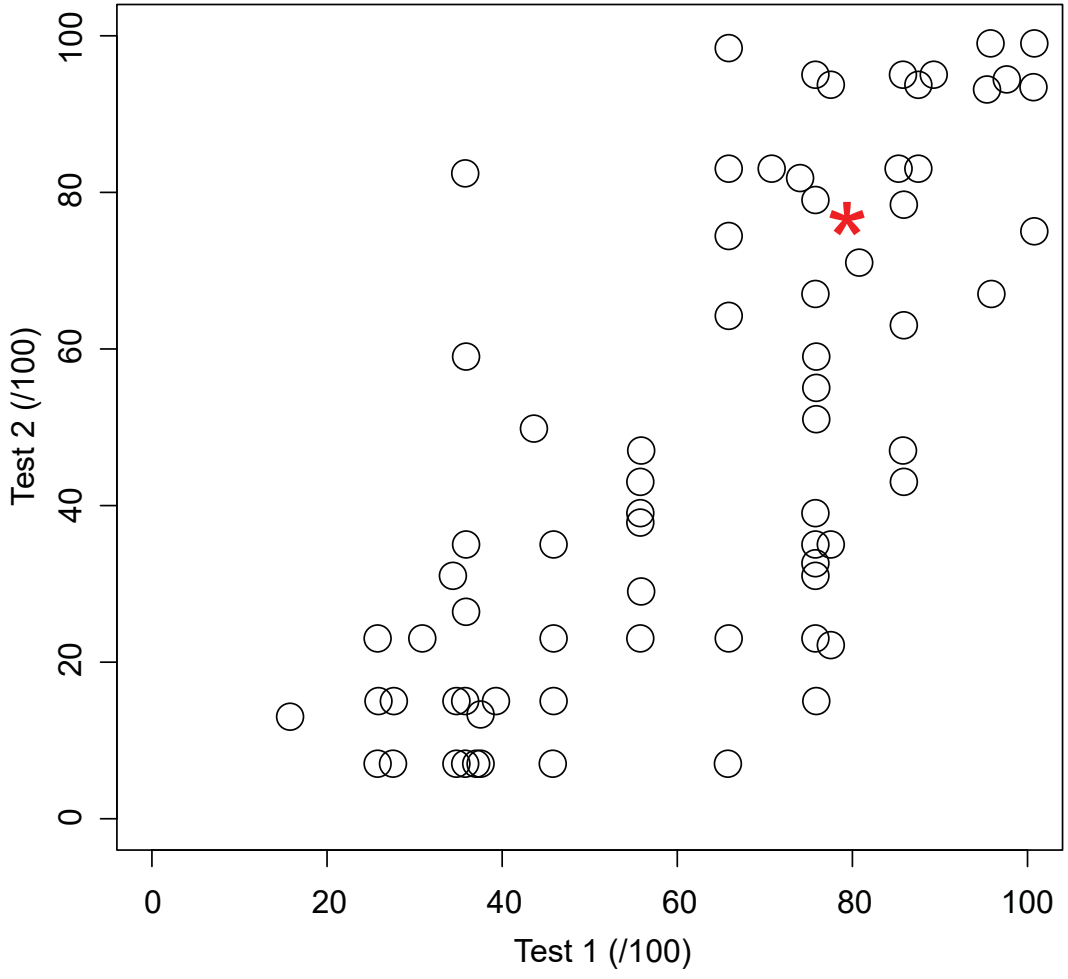
Failed tests:
restrictions on what features could be used

In some cases: correct solutions, violating the constraints
In other cases: incorrect solutions, meeting the requirements

[Finnie-Ansley et al. The Robots Are Coming: Exploring. . . . ACE '22, 2022.]

OpenAI Codex as a programmer

Student vs Codex performance



[Finnie-Ansley et al. The Robots Are Coming: Exploring. . . . ACE '22, 2022.]

Codex on the Rainfall problem

Algorithmic variations

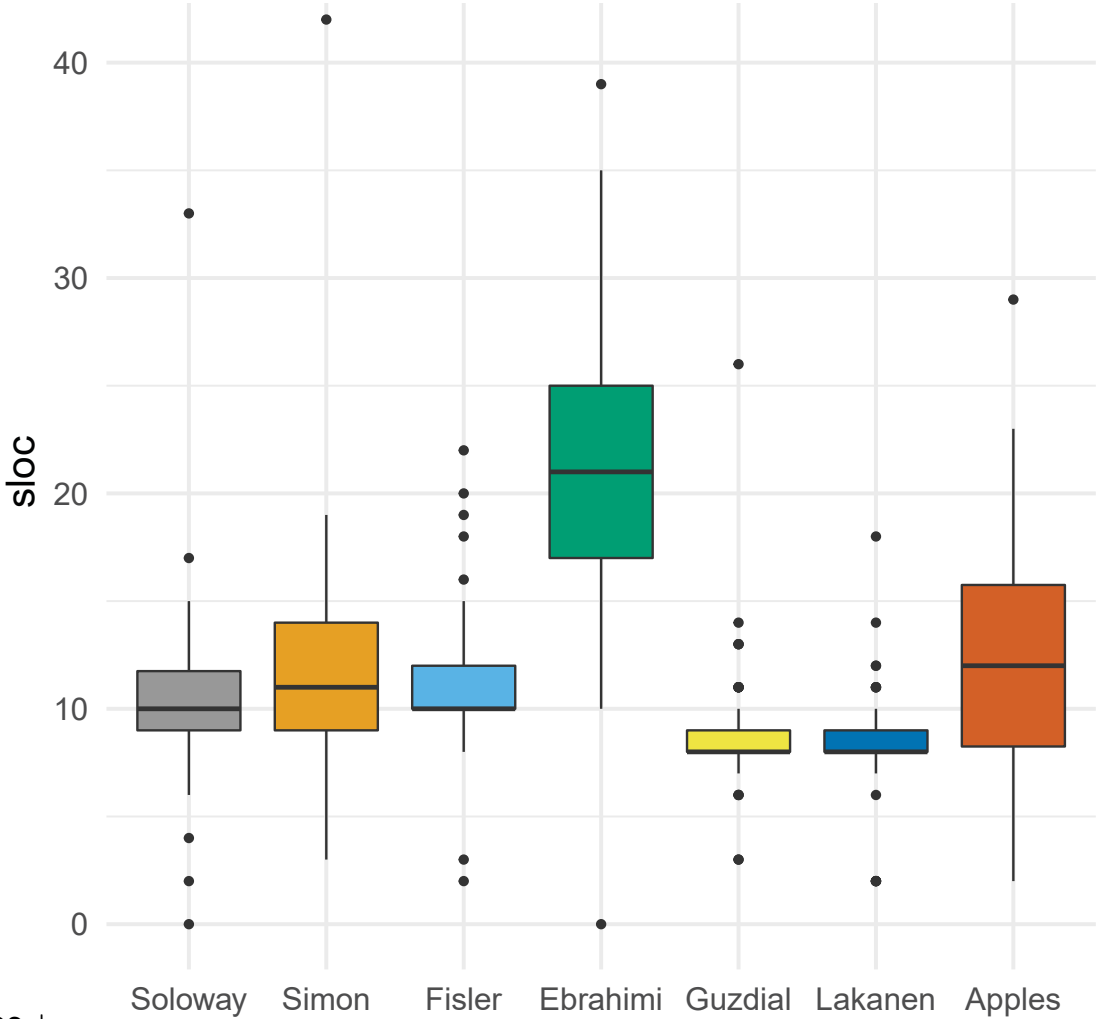
Variant	One while	One for	Sum / Len	Two Pass	Other
Soloway [39]	41	1	0	5	3
Simon [37]	5	36	1	8	0
Fisler [10]	3	42	1	3	1
Ebrahimi [9]	36	3	0	10	1
Guzdial et al. [16]	0	35	1	13	1
Lakanen et al. [18]	2	37	2	5	4
apples	4	23	3	16	4

[Finnie-Ansley et al. The Robots Are Coming: Exploring.... ACE '22, 2022.]

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Codex on the Rainfall problem

Source lines of code (sloc)



[Finnie-Ansley et al. The Robots Are Coming: Exploring. . . . ACE '22, 2022.]

OpenAI Codex as a Python programmer

Lab problems for first-year Math students

- Very good on well-described, analytic texts (as in exam texts)
- Much less good under constraints
- Much less good on more generic texts:
  erratic answers, "*very well*" (albeit incorrectly) justified

[Lodi, SM, Sbaraglia. Preliminary results. 2023]

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Notional machines

The aims of introductory courses are students' development of notional machines for reasoning about how a computer executes a program, and the development of the pragmatic skills for writing and debugging programs that computers can execute.

[Shapiro et al., How Machine Learning Impacts the Undergraduate Computing Curriculum. CACM 61(11), 2018]

Students will learn early on about two kinds of notional machine—that of the classical logical computer and that of the statistical model.

Notional machines

The aims of introductory courses are students' development of notional machines for reasoning about how a computer executes a program, and the development of the pragmatic skills for writing and debugging programs that computers can execute.

[Shapiro et al., How Machine Learning Impacts the Undergraduate Computing Curriculum. CACM 61(11), 2018]

Students will learn early on about two kinds of notional machine—that of the classical logical computer and that of the statistical model.

Shifts

- from rule-driven to data-driven thinking
- change in the role of syntax and semantics
- a shift away from algorithmic steps
- higher level of abstraction and black-boxed mechanisms
- new notional machines
- access to bodily and natural language interaction
- ...

[Tedre et al., Teaching Machine Learning in K–12 Classroom, IEEE Access, vol. 9, 2021]

From deterministic compilers to variability on produced code

Shifts

- from rule-driven to data-driven thinking
- change in the role of syntax and semantics
- a shift away from algorithmic steps
- higher level of abstraction and black-boxed mechanisms
- new notional machines
- access to bodily and natural language interaction
- ...

[Tedre et al., Teaching Machine Learning in K–12 Classroom, IEEE Access, vol. 9, 2021]

From deterministic compilers to variability on produced code

Teaching programming

In a holistic curriculum:

e.g., elementary school, non-vocational high-school ("liceo")

vs

In a professional curriculum:

e.g., vocational high-school, CS university degree

In a holistic curriculum:

Disciplines are lenses to understand reality

CS to understand the algorithmic fabric of our society

In a vocational curriculum

Disciplines are tools to modify reality

CS to weave the algorithmic fabric of our society

In a holistic curriculum:

Disciplines are lenses to understand reality

CS to understand the algorithmic fabric of our society

In a vocational curriculum

Disciplines are tools to modify reality

CS to weave the algorithmic fabric of our society

Concepts

- Collect and verify data
- Data representation
- Problem decomposition
- Abstraction
- Generalisation and pattern recognition
- Algorithms
- Automation
- Simulation, test, debug
- Parallelization
- Computational complexity

- And how these concepts are related to their linguistic expression

Concepts

- Collect and verify data
- Data representation
- Problem decomposition
- Abstraction
- Generalisation and pattern recognition
- Algorithms
- Automation
- Simulation, test, debug
- Parallelization
- Computational complexity

- And how these concepts are related to their linguistic expression

Generative AI calls for deeper integration with (other) STEAM subjects

Gilbert Simondon, 1924-1989

Gilbert Simondon

The human being is among and with machines

His/her relation to technical objects is not explained
in terms of instrumentality
but expressed in terms of being-with

[Lindberg, Being with Technique—Technique as being-with, Continental Philosophy Review 52, 2019]

Technical objects

A technical object is something that...

- is distinct, can be carried along and manipulated
- corresponds to the forces of the human body
- can be lost, and found again
- exists regardless of its use
- but is designed for a specific use, included in a sequence of gestures that make it meaningful

Open and closed technical objects

Gilbert Simondon interviewed by Yves Deforge, 1967

[*Le point sur la technologie*, de J. Jahan et Y. Deforge, 1967]

Open and closed

Open technical object

- its user knows how it works, and how it could be repaired
- "being" instead of "appearing" (être et ne pas paraître)
- a lesson of reality, of veracity, of respect for the past (because it shows the trace of its own evolution)

Closed technical object

- its user does not understand how and why it works
- it cannot be repaired
- it is unmodifiable
- it recalls the sacred, the untouchable

Closed objects: alienation

Les objets techniques qui produisent le plus d'aliénation sont ceux qui sont destinés à des utilisateurs ignorants.

De tels objets se dégradent progressivement : neufs pendant peu de temps, ils se dévaluent en perdant ce caractère, parce qu'ils ne peuvent que s'éloigner de leurs conditions de perfection initiale.

G. Simondon, Du mode d'existence des objets techniques [1958], Paris, Aubier, 2012. p. 250-251

The technical objects that produce the most alienation are those intended for ignorant users.
Such objects are gradually degraded: new for a short time, they then lose their character, because they can only move away from their initial conditions of perfection.

Closed objects: alienation

Les objets techniques qui produisent le plus d'aliénation sont ceux qui sont destinés à des utilisateurs ignorants.

De tels objets se dégradent progressivement : neufs pendant peu de temps, ils se dévaluent en perdant ce caractère, parce qu'ils ne peuvent que s'éloigner de leurs conditions de perfection initiale.

G. Simondon, Du mode d'existence des objets techniques [1958], Paris, Aubier, 2012. p. 250-251

The technical objects that produce the most alienation are those intended for ignorant users.
Such objects are gradually degraded: new for a short time, they then lose their character, because they can only move away from their initial conditions of perfection.

Therefore:

Yes, we should keep teaching (some) programming
To reduce alienation

Moreover:

The understanding and manipulation of programs as "hierarchical compositions of computational structures" is cognitively important

Programming is not (only) the production of a string
Managing abstractions *inside a program*

In a vocational curriculum

Disciplines are tools to modify reality
CS to weave the algorithmic fabric of our society

In a vocational curriculum

As the level of abstraction in computing education across educational levels steadily arose, the credo among computing cognoscenti became that one needs to be familiar with at least one abstraction level below that at which one is working.

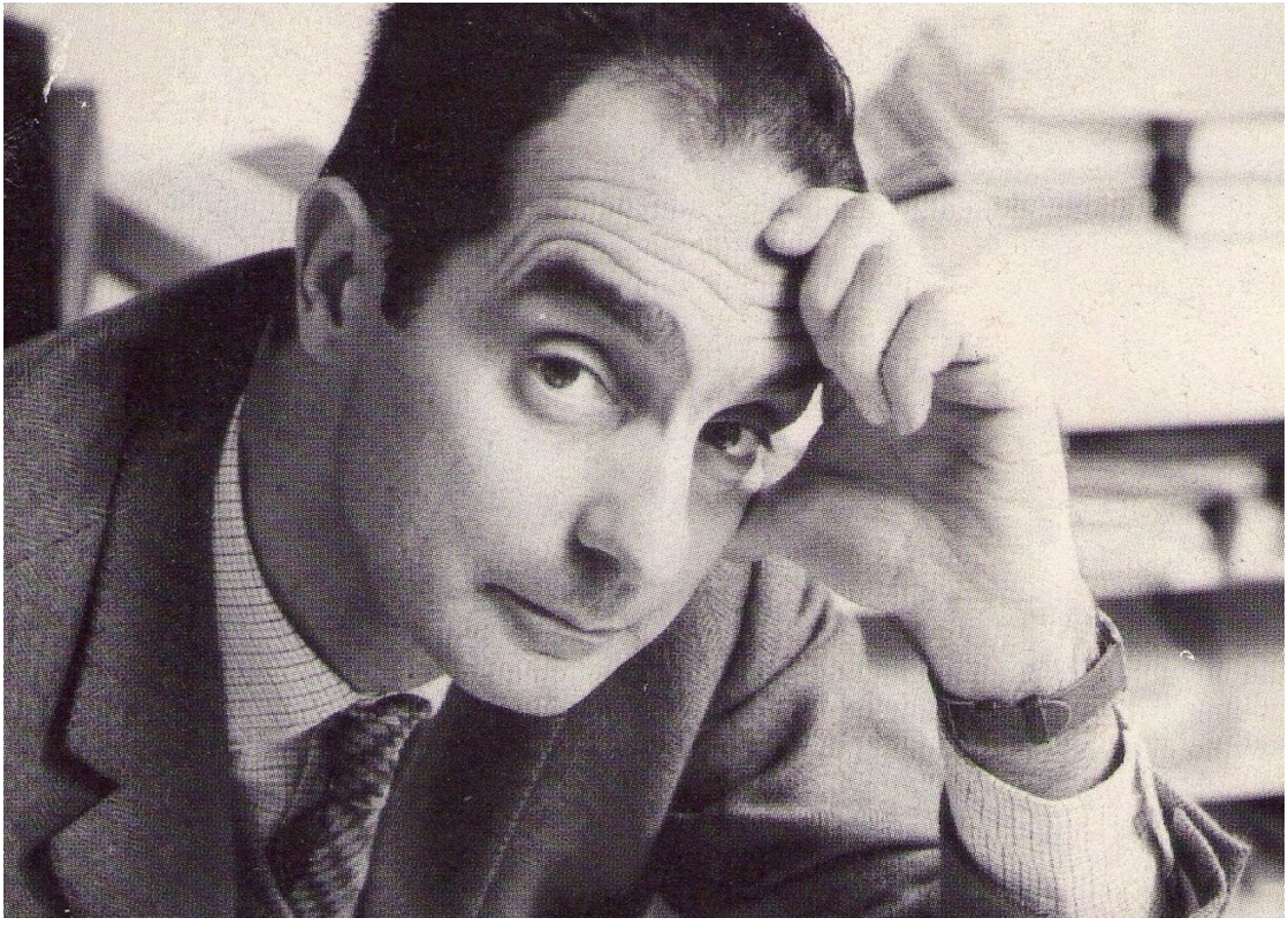
[Tedre et al., Teaching Machine Learning in K–12 Classroom, IEEE Access, vol. 9, 2021]

In a vocational curriculum

Being able to produce code in a programming language will lose importance

"Co-pilot"

Italo Calvino, 1923-1985



1967: Italo Calvino

Cybernetics and Ghosts

Lecture delivered in several Italian and European cities, November 1967.

[in Calvino, The Uses of Literature. Harcourt & Co., 1986. Translated by Patrick Creagh]

Calvino: Automatic writing

Will we have a machine capable of replacing the poet and the author?

I am thinking of a writing machine that would bring to the page all those things that we are accustomed to consider as the most jealously guarded attributes of our psychological life, of our daily experience, our unpredictable changes of mood and inner elations, despairs and moments of illumination.

Calvino: Automatic writing

What are these if not so many linguistic "fields," for which we might well succeed in establishing the vocabulary, grammar, syntax, and properties of permutation?

Developments in cybernetics lean toward machines capable of learning, of changing their own programs [. . .]

[We imagine] a literature-machine that at a certain point feels unsatisfied with its own traditionalism and starts to propose new ways of writing, turning its own codes completely upside down.

Calvino: The author disappears

Writers [. . .] are already writing machines [. . .]

What [we call] genius or talent or inspiration or intuition is nothing other than finding the right road empirically, following one's nose, taking short cuts.

Calvino: But something remains

Once we have dismantled and reassembled the process of literary composition, the decisive moment of literary life will be that of

Calvino: But something remains

Once we have dismantled and reassembled the process of literary composition, the decisive moment of literary life will be that of

reading.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Calvino: the reader

Let the author—this enfant gâté *of unawareness—disappear, to give place to a more thoughtful person, a person who will know that the author is a machine,*
and will know how this machine works.

Cf Simondon's open technical objects

Reading and correctness, 1

Compiler:
- rules based: deterministic output
- "easy" to understand and prove correct
- open technical object

Code generator:
- deep learning based: "non-deterministic" output
- opaque for our understanding
- closed technical object

Reading and correctness, 2

A reader to understand and convince themself
that the code is correct

Reading and maintenance

We cannot maintain a program changing its “high level” description for ChatGPT
Read, understand, and modify the (old style) code

Reading and code review

Automatically generated solutions may provide students with models that they can use for learning

Many benefits arise from looking at a variety of solutions, even when the code is flawed

More emphasis on code review, or evaluation of code

[Finnie-Ansley et al. The Robots Are Coming: Exploring. . . . ACE '22, 2022.]

Programs are meant to be read by humans
and only incidentally for computers to execute.

[Abelson & Sussman, Structure and Interpretation of Computer Programs, MIT Press, 1984]