Informatics Europe: 19th European Computer Science Summit
AI and the Future of Informatics Education Workshop
Session 2: **The impact of AI on Informatics curricula and educational practices**

Should we still teach "Programming"?

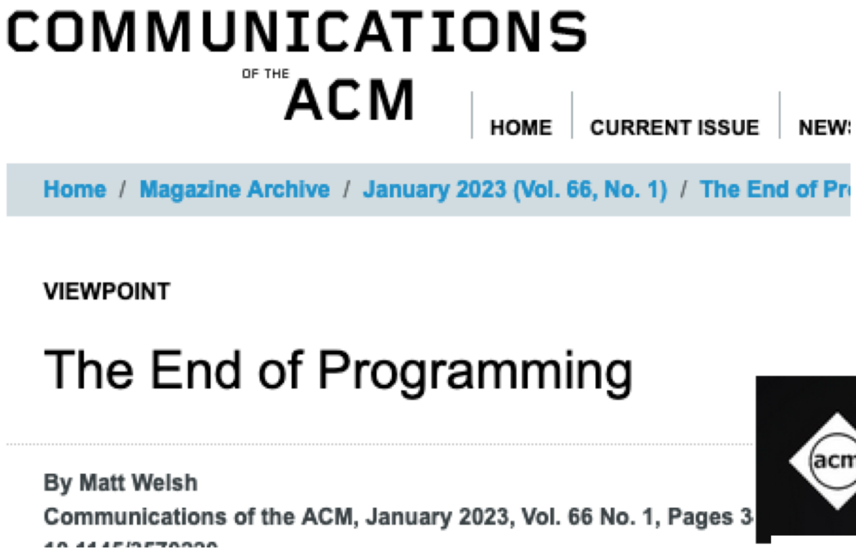
Thomas R. Gross

Department Informatik
ETH Zürich

Disclaimer

- **I do not know the answer(s).**
  - Maybe more than one answer
- **These comments are my personal view and not cleared with my institution or a (national) Informatics Association**
  - Based on research and teaching at several institutions
    - Design and implementation of compilers, programming tools, networking and computer systems
  - **Teaching the last 8 years "Introduction to Programming" to 1st semester computer science students at ETH Zurich**

You must have heard or seen ...



COMMUNICATIONS OF THE ACM

HOME | CURRENT ISSUE | NEW

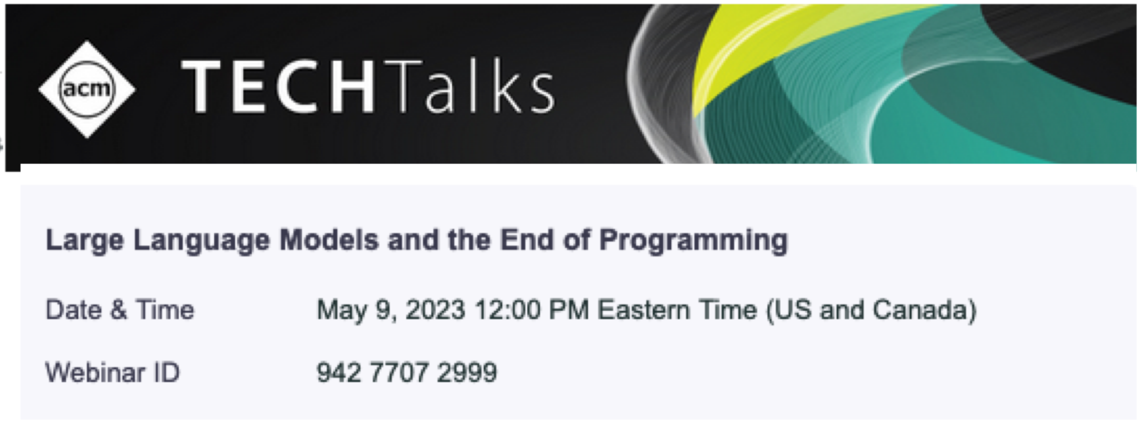
Home / Magazine Archive / January 2023 (Vol. 66, No. 1) / The End of Pr

VIEWPOINT

The End of Programming

By Matt Welsh

Communications of the ACM, January 2023, Vol. 66 No. 1, Pages 3

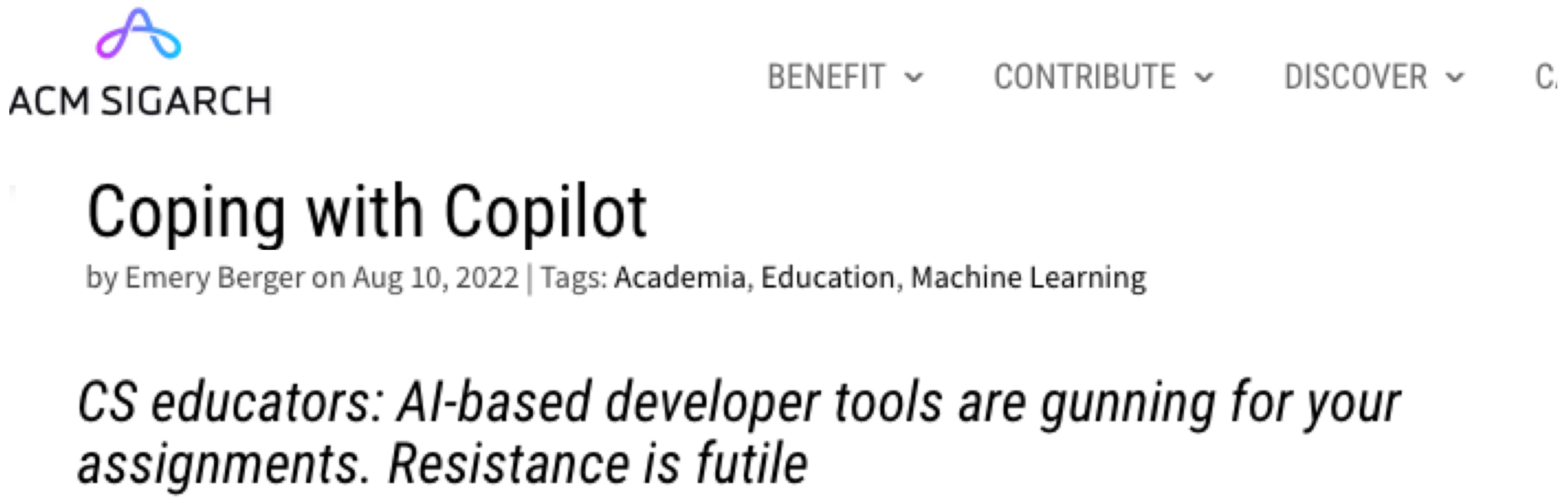


acm TECHTalks

Large Language Models and the End of Programming

Date & Time	May 9, 2023 12:00 PM Eastern Time (US and Canada)
Webinar ID	942 7707 2999

You must have heard or seen ...



Coping with Copilot

by Emery Berger on Aug 10, 2022 | Tags: Academia, Education, Machine Learning

CS educators: AI-based developer tools are gunning for your assignments. Resistance is futile

A number of minor issues

- How good/relevant are these tools for "Introduction to Programming"?

- Have we reached the end of programming?

- Do these tools make teaching programming {unnecessary | trivial}

Programming?

«Educators, generals, dieticians, psychologists, and parents program. Armies, students, and some societies are programmed.»

Alan Perlis

(Foreword to «Structure and Interpretation of Computer Programs», H. Abelson and G. J. Sussman, 1985)

Programming?

- Programming $\subset$ Software Engineering
- Software Engineering $\subset$ Programming

Programming?

- Related to "Computational Thinking"
  - Permeates different groups
  - Reflection
- High-school curricula
  - Scratch, Logo and other block programming languages
  - Other languages (from Python to Java to ....)
- Data analysis for domain experts
- (University-level) programming as foundation for system design

(University-level) programming and I2P

- I2P: Introduction to Programming
- Concepts, not syntax
- Foundation for various subfields
- Must be mastered to appreciate wide range of topics (AI, networking, security, program verification, computer vision, user interfaces, ...)

Programming (in the context of CS education)

«Programming as universal activity» by Vinton Cerf, CACM March 2016, vol 59(3) p 7

- Analyzing problems
- Breaking them down into manageable parts
- Finding solutions
- Integrating the results

Summary I2P@ETH

- Programming as foundation for system design
- Imperative programming (w/ class-based OOP)
- Practical slant ("Programming+")
    - Testing (JUnit)
    - Debugging, Design/partitioning
    - Leverage IDE
- Optional practice problems (some new each year)

Overview

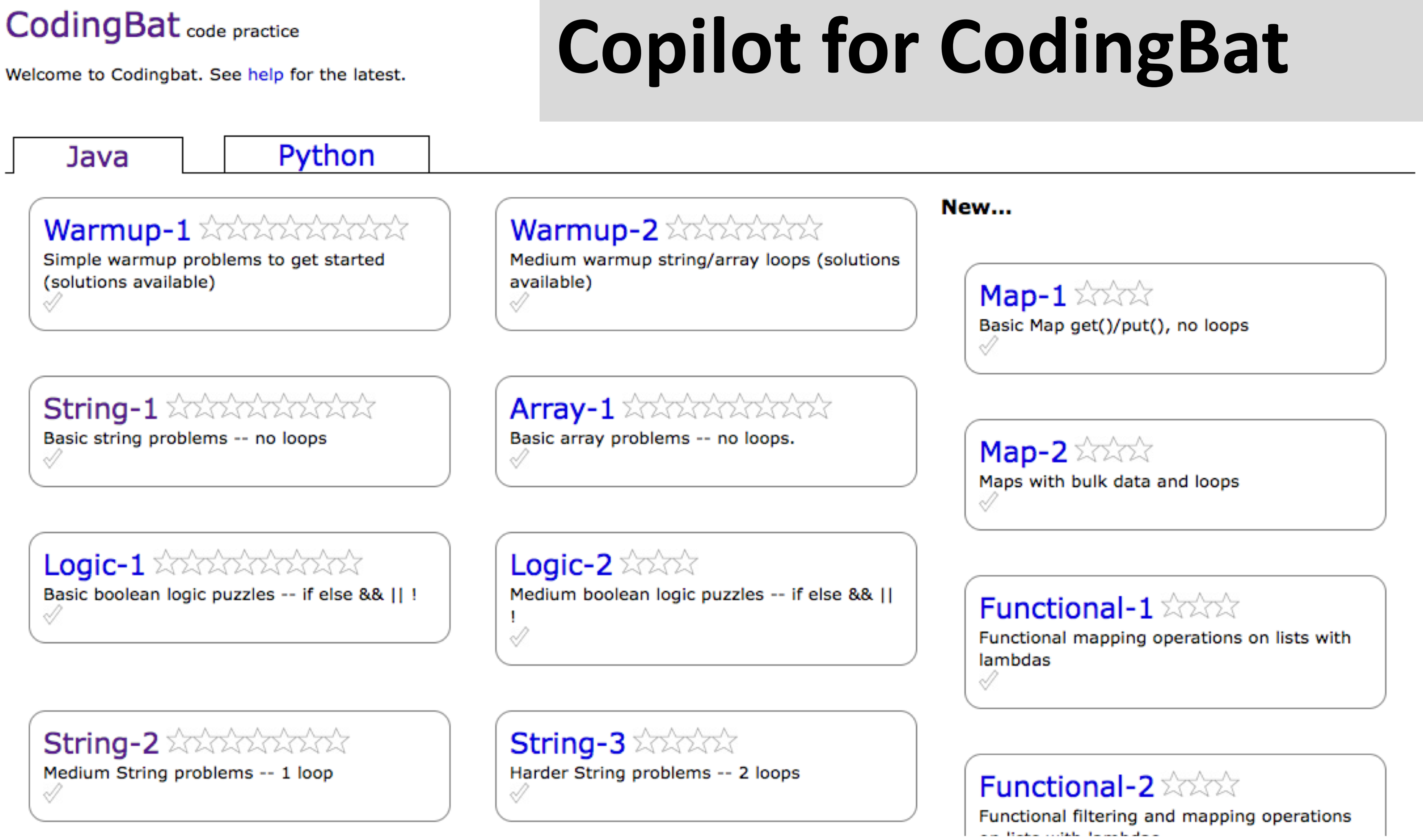
- How good/relevant are LLM tools (programmer assistants) for "Introduction to Programming"?
  - I2P as part of the ETH Computer Science B.Sc. program
  - Focus on one tool: GitHub Copilot

Copilot – friend or foe?

- General-purpose languages (e.g., Algol 60, FORTRAN IV) replaced assembly language programming
- (Compile-time checking of) Types raised programming level
- Object-oriented languages provide facilities to structure (larger) programs
- Parallel program generators avoid dealing with low-level details

- How can Copilot help?

Copilot for CodingBat



CodingBat code practice

Welcome to Codingbat. See help for the latest.

Java | Python

Warmup-1 Simple warmup problems to get started (solutions available)

Warmup-2 Medium warmup string/array loops (solutions available)

String-1 Basic string problems -- no loops

Array-1 Basic array problems -- no loops.

Logic-1 Basic boolean logic puzzles -- if else && || !

Logic-2 Medium boolean logic puzzles -- if else && || !

String-2 Medium String problems -- 1 loop

String-3 Harder String problems -- 2 loops

New...

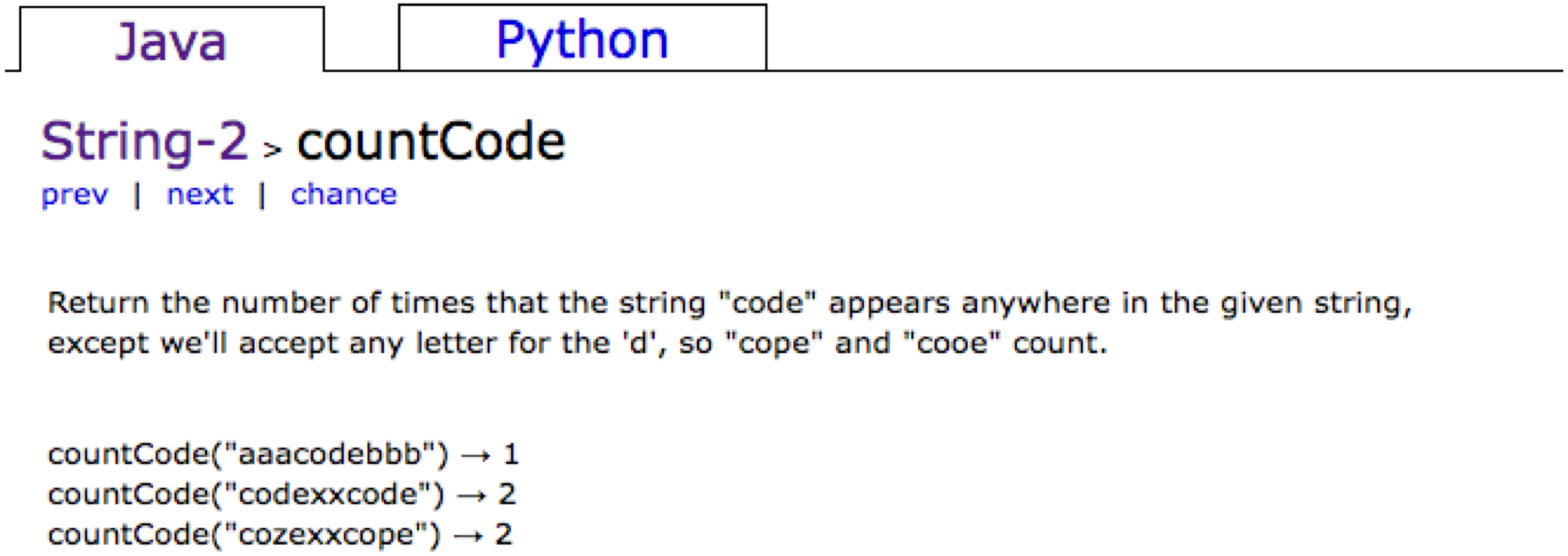
Map-1 Basic Map get()/put(), no loops

Map-2 Maps with bulk data and loops

Functional-1 Functional mapping operations on lists with lambdas

Functional-2 Functional filtering and mapping operations

countCode

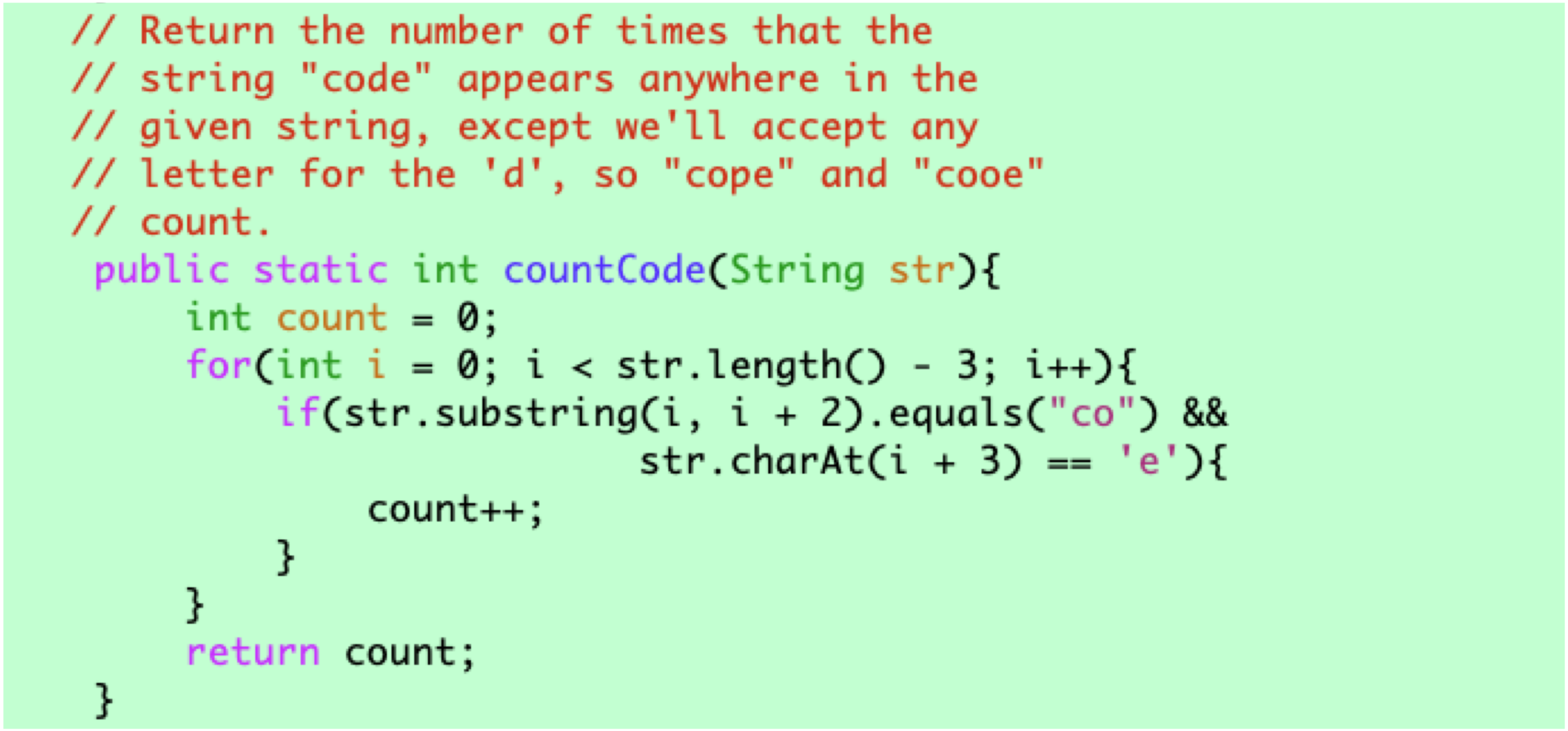


String-2 > countCode

Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.

```
countCode("aaacodebbb") → 1
countCode("codexxcode") → 2
countCode("cozexxcope") → 2
```

countCode

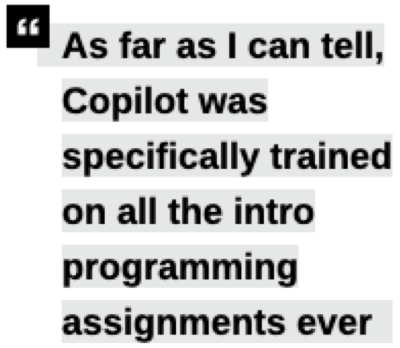


```
// Return the number of times that the
// string "code" appears anywhere in the
// given string, except we'll accept any
// letter for the 'd', so "cope" and "cooe"
// count.
 public static int countCode(String str){
     int count = 0;
     for(int i = 0; i < str.length() - 3; i++){
         if(str.substring(i, i + 2).equals("co") &&
                         str.charAt(i + 3) == 'e'){
             count++;
         }
     }
     return count;
 }
```

Copilot

- Picked good name for method
- Needed help with method parameter

- Summary: ✓



❝ As far as I can tell, Copilot was specifically trained on all the intro programming assignments ever

E Berger, Coping w/ Copilot

- **Others reported similar observation**
- **Copilot was trained on public repositories and popular collections of programming problems**
  - **Copilot will know the answers to these problems**
  - **Also textbook problems like TowersOfHanoi, Fibonacci, ...**

- **Copilot needed sometimes help with the structure**
  - **Method and method parameter type(s)**
  - **First suggestion often a "literal parameter"**

Copilot performance for other tasks

- **"New" (not from a repository or textbook)**
  - **One task/week, wit start in Week 4 or 5**
    - Comes with small (sample) test suite
    - Students should write tests
  - **Increasing difficulty/size**
  - **Optional for students but must be done within a week to be graded**

- **Step 1: Feed task description to Copilot**

Tasks 2018

Week	Size (Words)	Topic	Success
4	197	String-Addition: loops, arrays	0
5	456	Tool Rental: classes, arrays, iteration, JUnit	0
6	344	Valleys & Hills: arrays, data analysis, I/O	0
7	146	String Interleaving: recursion	0
8	151	List Reversal: references, working w/ classes	0
9	163	Class Puzzle: inheritance *(w/o classes or driver)*	0
10	600	Desk Calculator: inheritance, recursion	0
11	302	Data Analysis (FIFA): ArrayList<..>, Map<..>	0
12	194	Sublist Palindrome: Set<List<..>>, exceptions	100

Copilot

Your AI pair programmer

Copilot – hints and suggestions

- Hints and interactions
  - Informal classification
  - Not a strict hierarchy
  - Identify "highest level" for hint
- Rough classification
  - Experiments by G. Ponti
- Preliminary

Hints – summary

- Many tasks require hints
    - That is expected ...
- Appropriate hints require knowledge of programming concepts

Copilot – 2018 hints and suggestions

Week	Size (Words)	Topic	Total Required	Success
4	197	String-Addition: loops, arrays	4	100
5	456	Tool Rental: classes, arrays, iteration, JUnit	3	100
6	344	Valleys & Hills: arrays, data analysis, I/O	4	100
7	146	String Interleaving: recursion	2	0
8	151	List Reversal: references, working w/ classes	1	100
9	163	Class Puzzle: inheritance *(w/o classes or driver)*	12	100
10	600	Desk Calculator: inheritance, recursion	5	100
11	302	Data Analysis (FIFA): ArrayList<..>, Map<..>	3	100
12	194	Sublist Palindrome: Set<List<..>>, exceptions	0	100

Hints – summary

- Many tasks require hints
  - That is expected ...
- Appropriate hints require knowledge of programming concepts
  - There is no free lunch
  - Unless your task description matches completely

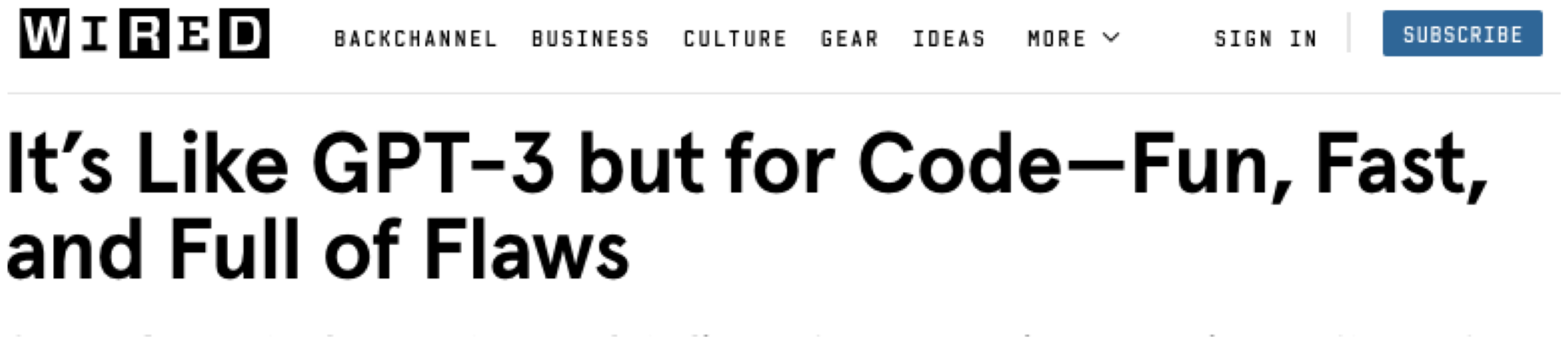
Acknowledgments

- Michael Faes, Gaurav Parthasarathy, Felix Wolf
  - Co-authors of programming problems

- Giacomo Ponti
  - Experiments with Copilot

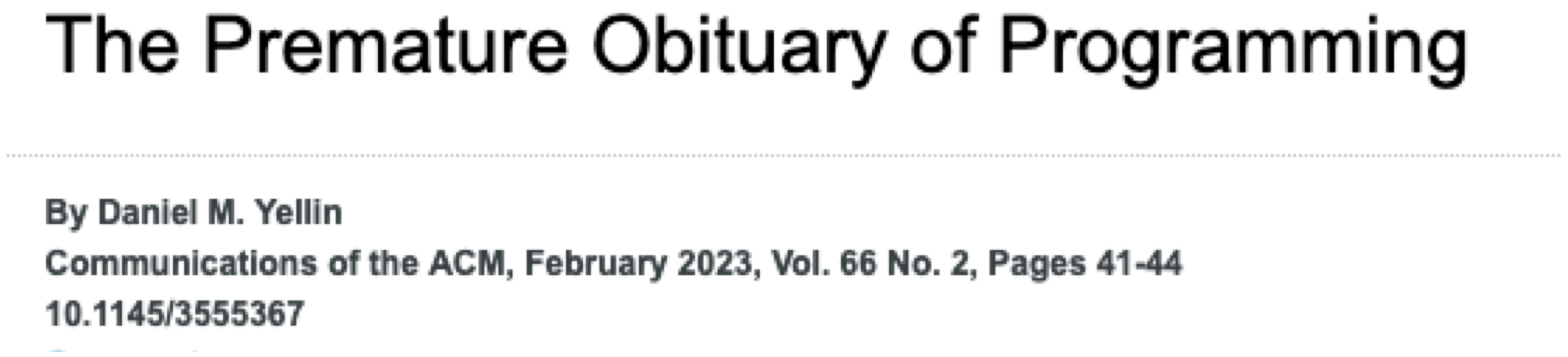
Discussion

- Copilot is not an automatic program generator
- Copilot seems to expect a fairly skilled pair programmer
  - May use advanced language features
  - Solution to early problem uses classes and generics
- Copilot encourages good programming practice
  - Comments, good examples in real-life setting
- Student must (still) understand unit tests, carefully read problem description
  - Text understanding crucial skill, adversary reading/analysis
  - Requirements analysis



It’s Like GPT-3 but for Code—Fun, Fast, and Full of Flaws

The end of programming?

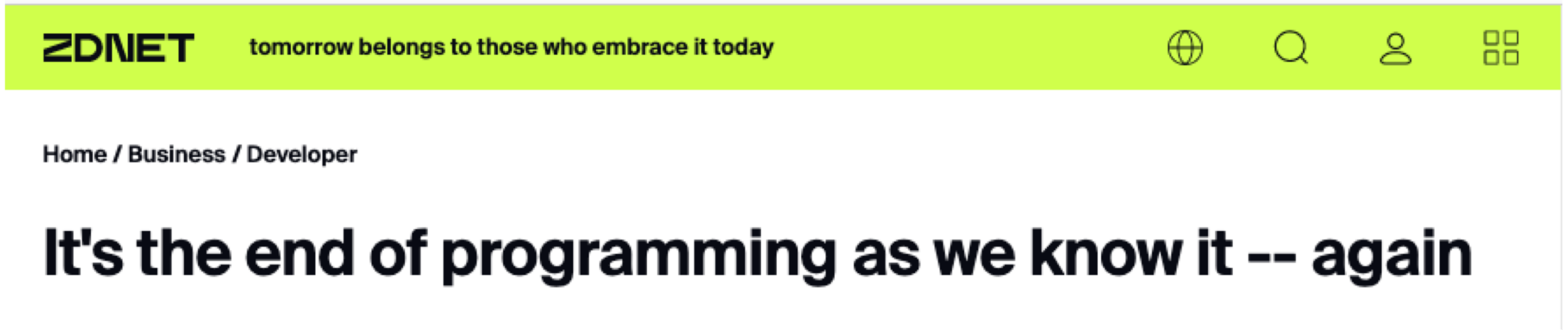


The Premature Obituary of Programming

By Daniel M. Yellin

Communications of the ACM, February 2023, Vol. 66 No. 2, Pages 41-44

10.1145/3555367



ZDNET tomorrow belongs to those who embrace it today

Home / Business / Developer

It's the end of programming as we know it -- again

Issues outside of scope

- Use of LLM tools to support teaching
  - Task development, individual feedback, grading, ...
- Security implications
- IP issues
- ... and many more

Should we teach programming? (My take)

- Depends on the student population/B.S. program design
- All students? High school? Kindergarden?
  - Programming the "Latin of the 21st century"?
  - Is Scratch/Logo/… the best way to educate (future) digital citizens?
- System designers and implementors
  - No alternative yet

Should we teach programming? (Your input)

- Should the answer depend on the student’s level
  - What are good levels?
  - K-12? Kindergarden? Secondary schools? Tertiary? University?
- For each group, what is the answer?
- What should be the focus of ”Programming” ?
  - Text understanding? Analytical/Critical thinking?
  - Testing
  - System design and analysis
  - Use of tools

Related aspects – Teaching Programming

- Student practice a big problem
  - Must synthesize solution
  - Partition problems into sub-problems, find solutions, integrate results
- Many students may underestimate importance of practice (and effort required)
  - Evidence that practice is crucial
  - Exam: 2 hrs live programming plus 40 min written test (foundations)

Concluding remarks

- LLM Tools (Copilot) – the end of programming?
  - Not really for B.S. CS students that design future systems
  - Assistance – helpful for skilled programmer, challenging for others
  - Students must still master concepts
- Copilot implications (for I2P)
  - Even more emphasis on testing & debugging & requirements analysis
- Challenges and research issues
  - Student engagement: may ignore practice (even more than now)
  - Interface (human) programmer <--> tool (programmer assistant)