Computing Education in the Era of Generative AI

AUTHORED BY: PAUL DENNY, JAMES PRATHER, BRETT A. BECKER, JAMES FINNIE-ANSLEY, ARTO HELLAS, JUHO LEINONEN, ANDREW LUXTON-REILLY, BRENT N. REEVES, EDDIE ANTONIO SANTOS, AND SAMI SARSA

PRESENTATION BY:
ASAD MIRZA

Overview of the Paper

This paper's goal is to..

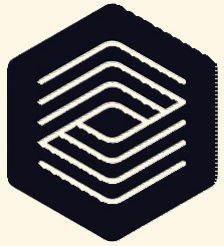
- Go over the use of LLM's in teaching environments.
- Conduct multiple case studies, showing how LLM's such as Codex GPT-4 perform on first year programming exams.
- Give multiple examples of how LLM's can be utilized by teachers.

Large Language Models and Code

Github Copilot

A LLM extension applied to your IDE such that it can both autofill and generate code based on instructions

Codex



A descendant of GPT-3 trained on more than 50 million public Github repositories – now deprecated

GPT-4

Released in March 2023, it was the most advanced LLM produced up until GPT-4o was revealed.

Challenges Ahead

Various questions emerge for educators

- How proficient are these tools with coding?
- Can Student's with no programming experience pass coding assessments with these tools?
- Do we need to change our teaching methods?

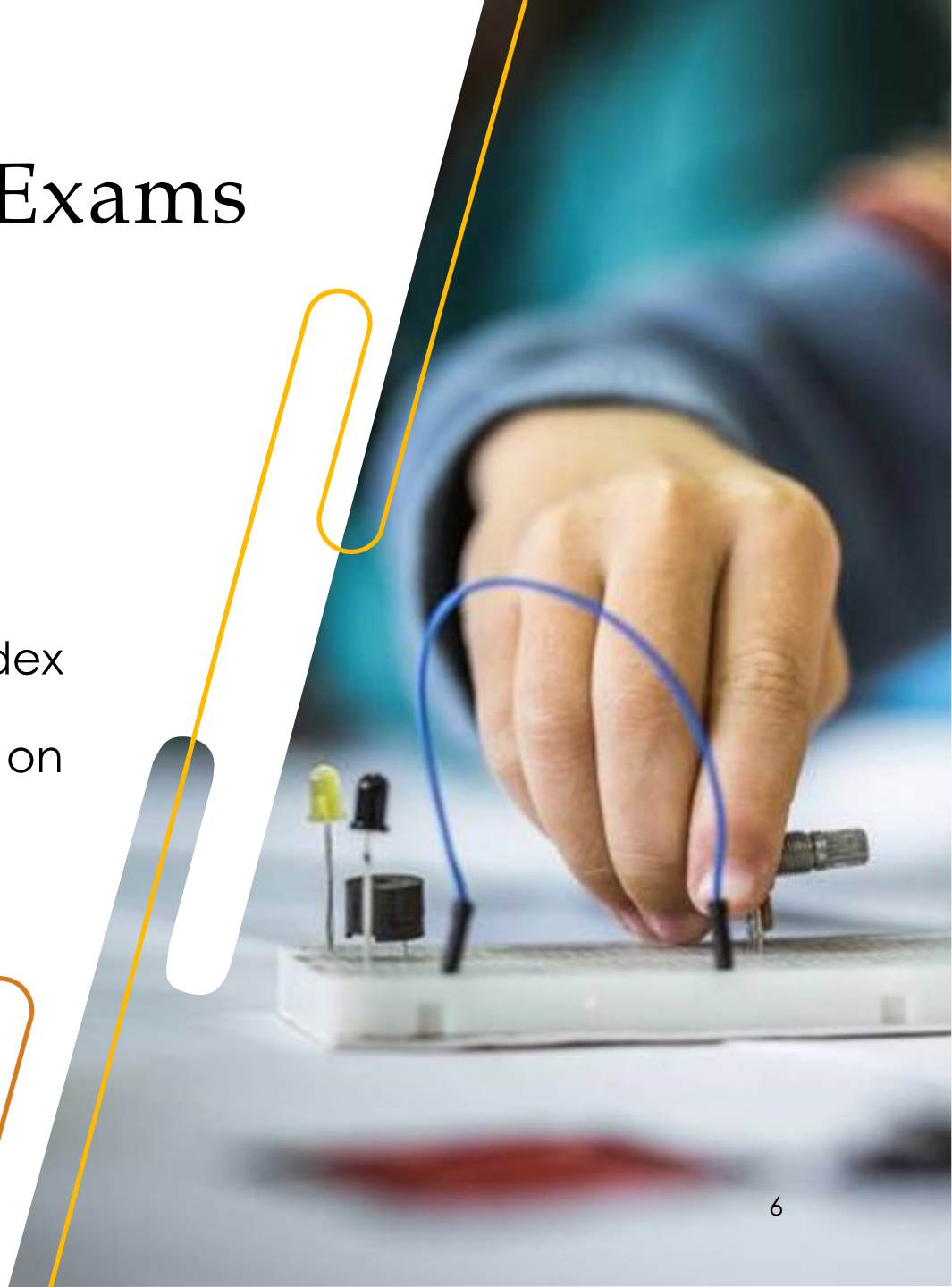
Let's take a look at some case studies outlined by the paper.

Case Study

Codex/GPT-4 against real exam questions

Codex vs CS1 Programming Exams

- Exams were general python exams
- Questions ranged from simple calculations to more complex data manipulations
- Questions were copy and pasted verbatim into Codex
- Codex scored 15.7/20 on the first exam and 19.5/25 on the second exam.
- Codex mainly struggled when tasked to not use language specific tools



make ascii of a dog

Creating an ASCII art representation of a dog can be a bit more complex. Here's a simple version:

javascript Copy code

```
 / \__
(    @\___
 /         O
/   (_____/
/_____/  U
```

This is a straightforward ASCII illustration of a dog. I hope you like it!

make ascii of a horse

Here's an ASCII art representation of a horse:

go Copy code

```
          ,~,
     ,~~, )  )~\
  ( )   \~ , )  )
   )    , )~~) ~\
  (   ) ' '  )  )
   `- ) ) ~~~)~
    ,~J `\  )'
  ()' `- ,~~
        (~~
         `)
```

This simple depiction captures the basic outline of a horse. Enjoy!

I apologize for that. Let's try a more detailed ASCII art of a horse:

lua Copy code

This version aims to better capture the overall shape and details of a horse. I hope you find this one more accurate!

The Rainfall problem

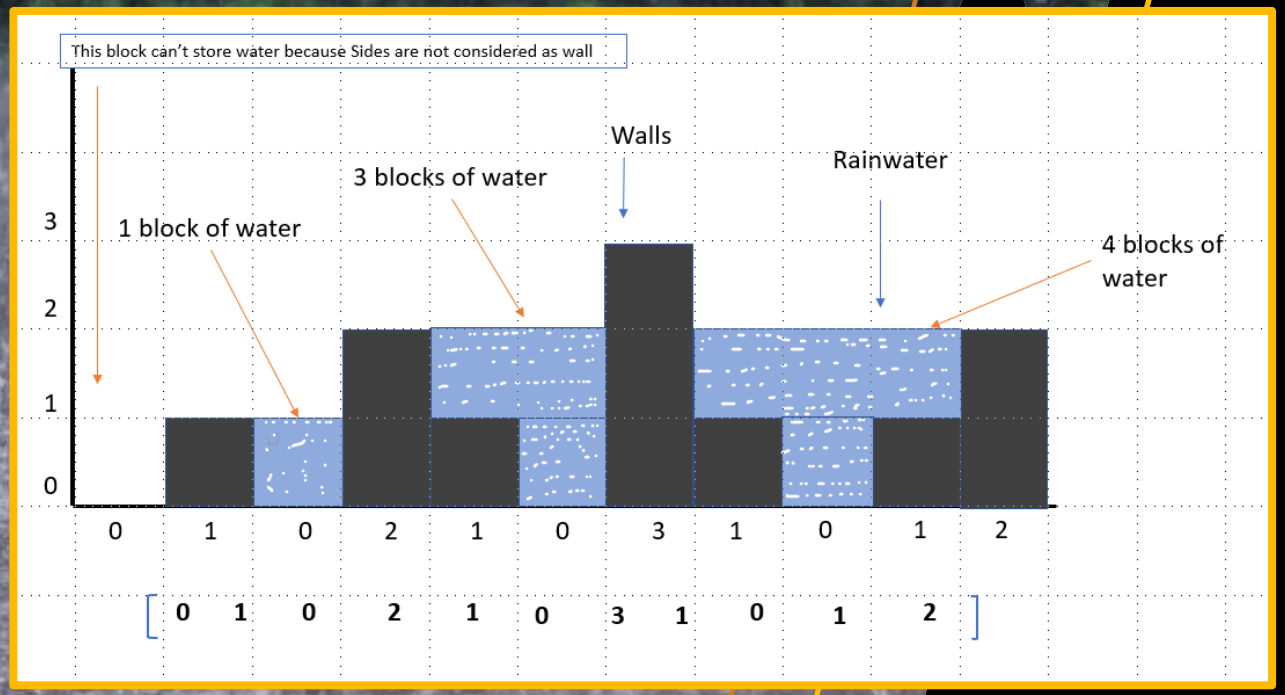
Codex's struggles

Definition

"Given an array of integers where each element represents the height of a bar in an elevation map, compute how much water can be trapped between these bars after it rains.".

Codex struggled heavily with this problem, resulting in an average score of 50% after 3500 evaluations.

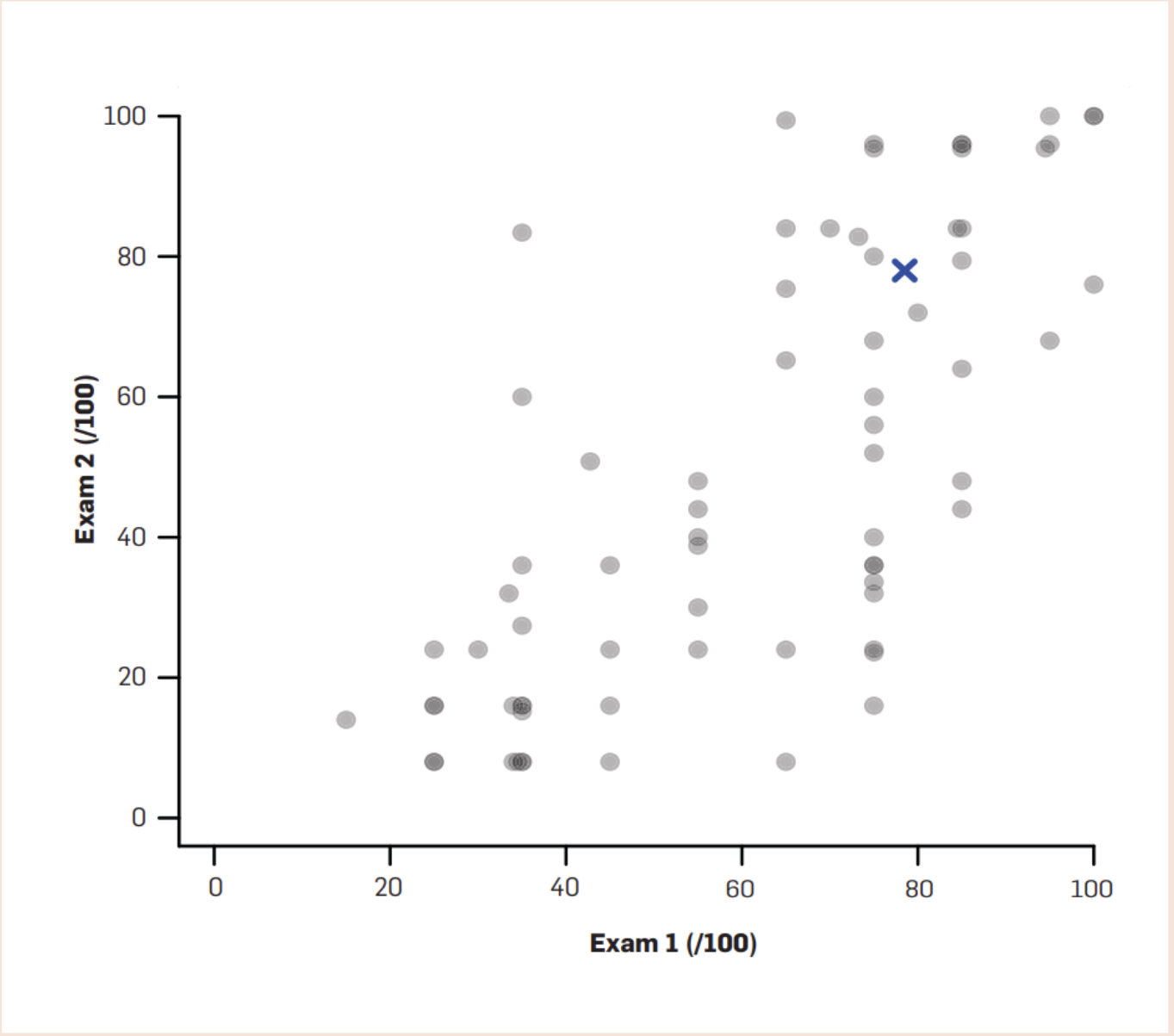
Codex's responses are distinct, depending on the prompt.



Graph:
Showcasing Codex's Exam Scores Against 71 Other Students

GPT-4 On the Other Hand….

Figure 1. Student scores on Exam 1 and Exam 2, represented by circles. Codex's 2021 score is represented by the blue 'X'.



GPT-4's Impressive Performance

- GPT-4 had scores of..
  - 99.5% on the first exam, 94.4% on the second exam
  - GPT-4, successfully solved all rainfall variants, with only minor formatting errors in some evaluations.
  - A follow up study conducted by James Finnie-Ansley showcased GPT-4 performing quite well in CS2 exams
  - GPT-4 has also been shown to perform well in Parson's problems

Academic Honesty With LLM's

A Nuanced Issue

The advent of AI has brought about new issues

AI generated code is troublesome for plagiarism detectors

Should they be allowed? Misconduct?

University policies dictate that it's plagiarism however…

Comes down to the instructor.

Can be seen as productivity tools used professionally.

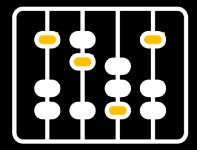
Learner Over Reliance

Over reliance on AI for novices can be detrimental

- Limited metacognition development(thinking about thinking)
- LLM's aren't perfect, e.g AlphaCode.
- Biases

Opportunities Ahead, Taking Advantage

Teaching



Per Brett A Becker and others, generative tools are revolutionary

Learning Oppurtunities

Mark Chen and others described Codex's as a tool to "aid in education and exploration"

Learning Resources

The Potential of LLM's for Computer Science Educators

Figure 2. A 'priming' exercise consisting of one complete example followed by the prompt for a new one.

```
"""Exercise 1
```

Table 1. Analysis of 240 generated programming exercises, showing the proportion that met criteria regarding sample solutions and test cases.

Has sample solution?	203/240	84.6%
Sample solution executable?	182/203	89.7%
Has test cases?	170/240	70.8%
All tests pass?	51/165	30.9%
Full (100%) statement coverage?	48/51	94.1%

```
--Keywords--
basketball
function
list
for loop
--Problem statement--
```

Figure 3. Example output generated by Codex using the priming exercise from Figure 2.

```
Write a function called count_rebounds that
↪  takes a list of basketball players as an
↪  input.  The function should return the
↪  total number of rebounds for the entire
↪  team.  Each element in the list is itself a
↪  list containing the player's name, their
↪  points, and their rebounds.
--Sample solution--
def count_rebounds(players):
  total = 0
  for player in players:
    total = total + player[2]
  return total
```

Figure 4. Explanation generated by the ChatGPT model of the code shown in the 'Sample solution' area in Figure 3.

Certainly! Here's a step-by-step explanation of the program:

1. The function **'count_rebounds'** takes in a list of players as an argument. Each player is represented as a tuple containing three elements: a player's name (a string), their number of points scored (an integer), and their number of rebounds (also an integer).
2. The function initializes a variable **'total'** to 0. This variable will be used to store the total number of rebounds among all the players.
3. The function enters a **'for'** loop that iterates over each player in the list of players.
4. For each player, the function adds the player's number of rebound to the **'total'** variable.
5. After the **'for'** loop completes, the function returns the value of the **'total'** variable. This is the total number of rebounds among all the players.

Code Explanations

- Codex's explanations were found to be 90% correct
- GPT-3 Model is really consistent
- Students find high level summaries of code helpful

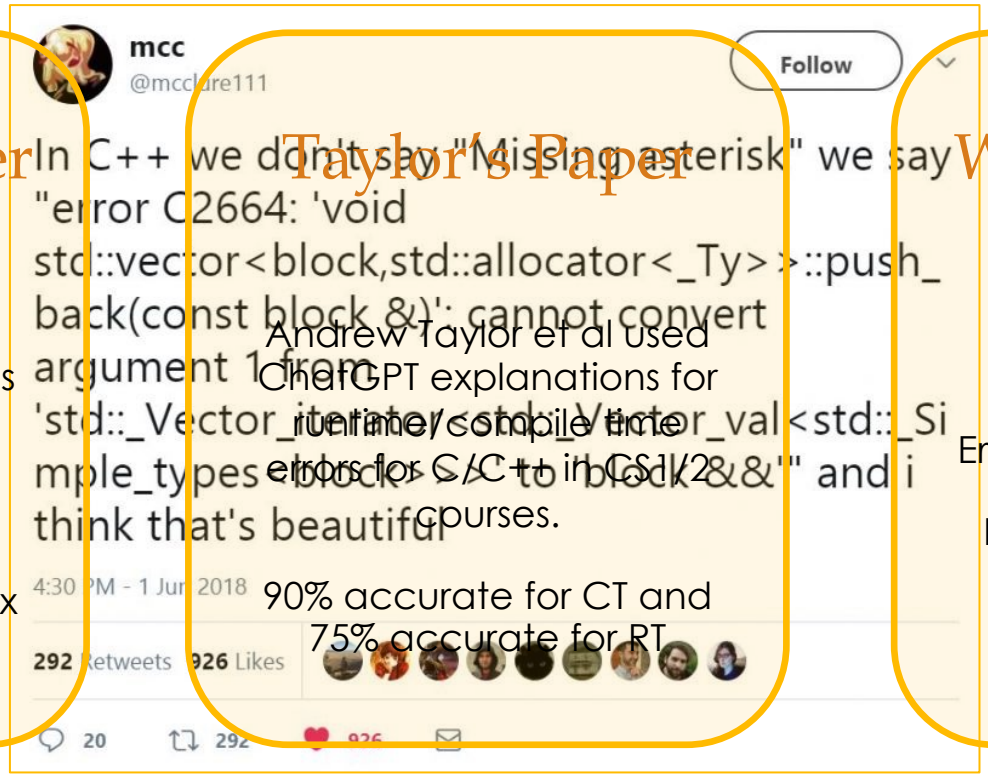
Figure 3. Example output generated by Codex using the priming exercise from Figure 2.

```
Write a function called count_rebounds that
↪  takes a list of basketball players as an
↪  input.  The function should return the
↪  total number of rebounds for the entire
↪  team.  Each element in the list is itself a
↪  list containing the player's name, their
↪  points, and their rebounds.
--Sample solution--
def count_rebounds(players):
  total = 0
  for player in players:
    total = total + player[2]
  return total
```

LLM's to improve Programming Error Messages

Leinonen's Paper

Juho Leinonen et al explored the use of LLM's in PEM's in their paper "Using Large Language Models to Enhance Programming Error Messages" utilizing codex



Taylor's Paper

Andrew Taylor et al used ChatGPT explanations for runtime/compile time errors for C/C++ in CS1/2 courses.

90% accurate for CT and 75% accurate for RT

Wang's Findings

Wang et al found LLM boosted PEM's led to

Errors repeated 23.5% less

Errors resolved in 36.1% less attempts

New Pedagogical Approaches

Weighing Different Options

Restructuring CS1's Syllabus

- A new method introduced by Zingaro and Porter in their paper, "Learn AI-Assisted Python Programming: With GitHub Copilot and ChatGPT".
- Introducing LLM's early for CS1, rather than focusing on creating low level code,
- A top-down approach is conducted, CoPilot is immediately introduced
- Advantages: still requires students to know how to code, develops problem solving/divide and conquer skills

Prompt Engineering/ Assesments

- A new skill that will be developed by students is prompt curating/engineering
- Including specific hints about the problem helps the LLM
- Paul Denny and others found PE/breaking down the problem into smaller sub problems to be particularly helpful
- A new assessment, assigning prompt problems
  - E.g the rainfall problem
  - Useful for debugging

Integrating LLM Tools

Curating Guardrails for These Tools

GPT-4 as a ‘Socratic Tutor’

Responds to students with probing questions rather than the answer

For example CodeHelp.

Code generators decrease errors

May seem good but slows the error-feedback loop

Large blocks of code can be counterproductive

LLM’s may generate code that is against assignment requirements/too advanced for new programmers

Proper LLM’s should be controlled with how much code is inserted at once e.g multiple steps taken/what tools are used.

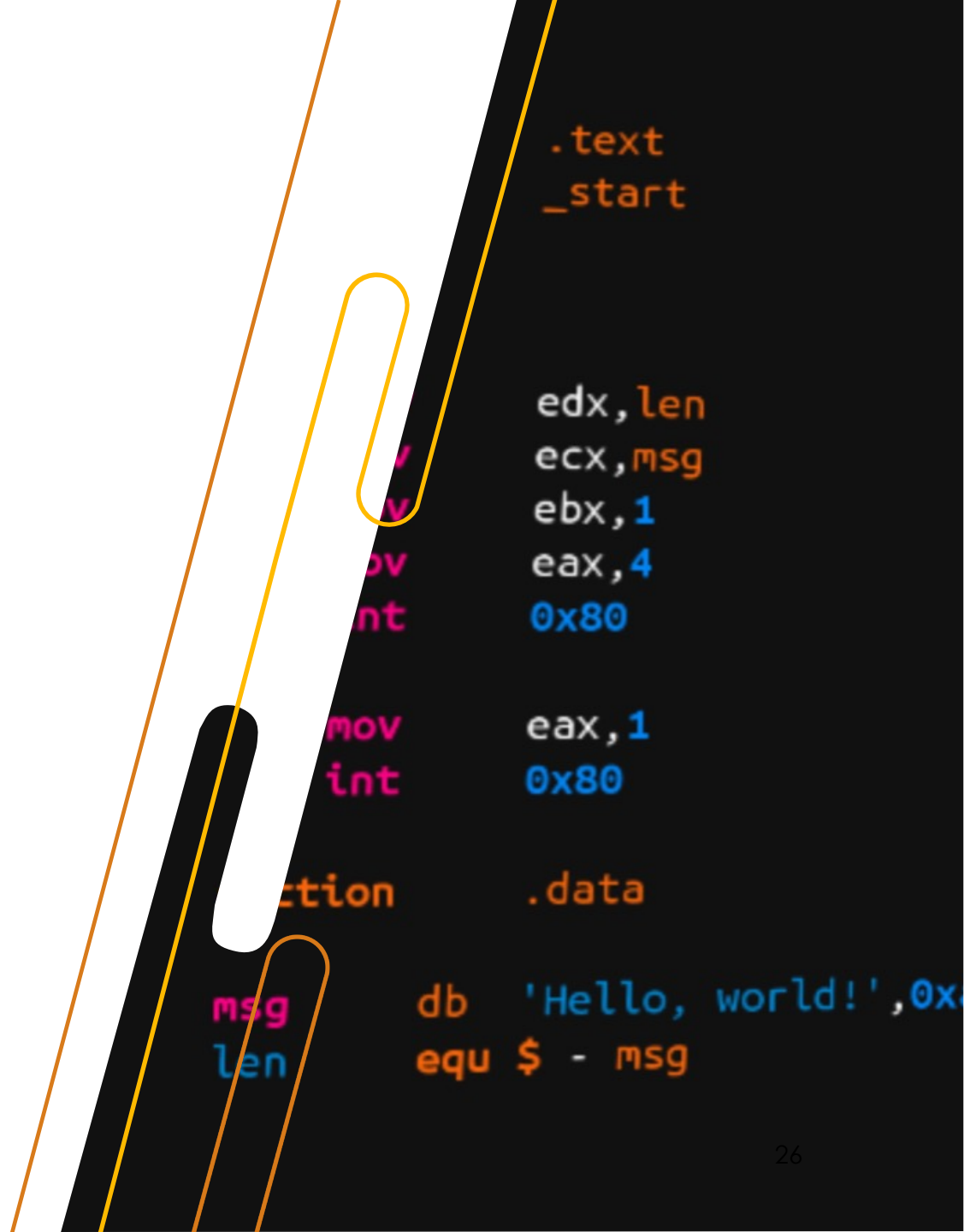
Where Do We Go Now? The Emergence of Powerful Tools = The End of Programming?

Views From Experts

- Matt Welsh forsees “the end of programming” due to AI
- Bertrand Meyer explains the emergence of AI as a new pardigm.
- Meyer resurgence of classic software-engineering skills

Productivity Benefits

The advent of AI is much like the transition from Assembly to higher level languages.



Role of Code Literacy

- Essential for ensuring alignment between intentions and generated code.
- Avoiding reliance on blind faith in generated content.
- Writing code remains valuable for learning fundamental concepts.

Professional Development and Coding

- Future developers may write less 'low-level' code.
- Generated code will still need modification and integration.
- Shift in emphasis towards modifying AI-generated code in introductory courses.
- Editing code directly is more effective than using natural language for complex programs.

Wrapping Up

- Tools like Copilot and ChatGPT as valuable learning assistants.
- Explaining concepts to diverse learners.
- Generating exemplar code and contextualized learning resources.
- Emergence of new pedagogies leveraging code-generation tools.
- Teaching effective communication with tools and focusing on problem specification.

Thank You!