

```
// Large language models  
// and the end of programming
```

```
// Matt Welsh <matt@fixie.ai>  
//  
// October 2023
```

*** COMPUTER SCIENCE IS DOOMED ***

Computer Science has always been about one thing:

Translating **ideas** into **programs**.

CS is the study of how to take a problem and map it onto instructions that can be executed by a Von Neumann machine.

*** COMPUTER SCIENCE IS DOOMED ***

Critically, the goal of CS has always been that programs are **implemented**, **maintained**, and **understood** by **humans**.

But -- spoiler alert! -- humans suck at all of these things.

Let's just make programming easier!

Fifty years of programming language research has done **nothing** to improve the state of affairs.

No amount of improvement to type systems, debugging, static analysis, linters, or documentation is going to magically solve this problem.

Let's just make programming easier!

FORTRAN (1957)

```
DO 200 NROWS = 1, 10000
  CALL OFETCH(CDA)
  IF (CDA(7) .EQ. 1403) GO TO 300
  IF (CDA(7) .NE. 0 .AND. CDA(7) .NE. 1406) THEN
    CALL ERRRPT(LDA, CDA)
    RV = -1
    RETURN
  ENDIF
  DO 100 COL = 1, NOV
    IF (INDP(COL) .LT. 0 .AND. DBTYPE(COL) .NE. 1) THEN
      WRITE (*, 903), '
      FORMAT ('+', A9, $)
    ELSE IF (INDP(COL) .LT. 0 .AND. DBTYPE(COL) .EQ. 1) THEN
      WRITE (*, 905), '
      FORMAT ('+', A11, $)
    ELSE
      IF (DBTYPE(COL) .EQ. 3) THEN
        WRITE (*, 904) DVARI(COL), '
        FORMAT ('+', I6, A3, $)
      ELSE IF (DBTYPE(COL) .EQ. 4) THEN
        WRITE (*, 906) DVARF(COL), '
        FORMAT ('+', F8.2, A1, $)
      ELSE
        WRITE (*, 908) DVARC(COL), '
        FORMAT ('+', A10, A1, $)
      ENDIF
    ENDIF
  CONTINUE
  WRITE (*, '(1X)')
CONTINUE
```

Let's just make programming easier!

BASIC (1964)

```
PROGRAM:CONWAY
:While 1
:For(X,2,9,1)
:For(Y,2,17,1)
:If [A](Y,X)
:Then
:Output(X-1,Y-1,"X")
:Else
:Output(X-1,Y-1," ")
:End
:[A](Y-1,X-1)+[A](Y,X-1)+[A](Y+1,X-1)+[A](Y-1,X)+[A](Y+1,X)+[A](Y-1,X+1)+[A](Y,X+1)+[A](Y+1,X+1)→N
:If ([A](Y,X) and (N=2 or N=3)) or (not([A](Y,X)) and N=3)
:Then
:1→[B](Y,X)
:Else
:0→[B](Y,X)
:End
:End
:End
:[B]→[A]
:End
```

Let's just make programming easier!

APL (1966)

```
life ← {>1 ω v.∧ 3 4 = +/ +≠ -1 0 1 ∘.⊖ -1 0 1 ϕ.. ⏟ω}
```

Let's just make programming easier!

Rust (2010)

```
fn neighbour_counts(col: &Colony) -> HashMap<Cell, i32> {
    let mut ncnts = HashMap::new();
    for cell in col.iter().flat_map(neighbours) {
        *ncnts.entry(cell).or_insert(0) += 1;
    }
    ncnts
}

fn generation(col: Colony) -> Colony {
    neighbour_counts(&col)
    .into_iter()
    .filter_map(|(cell, cnt)|
        match (cnt, col.contains(&cell)) {
            (2, true) |
            (3, ..) => Some(cell),
            _ => None
        })
    .collect()
}
```


This is how I program now...

48

```
49  const TRANSCRIPT_SUMMARY = `Provide a one or two sentence summary of the  
50  following segment of a podcast transcript. Only use the information  
51  provided in the text; DO NOT use any information you know about the world.  
52  Include the title of the podcast, the name of the episode, and the  
53  names of the speakers, if known.`;
```

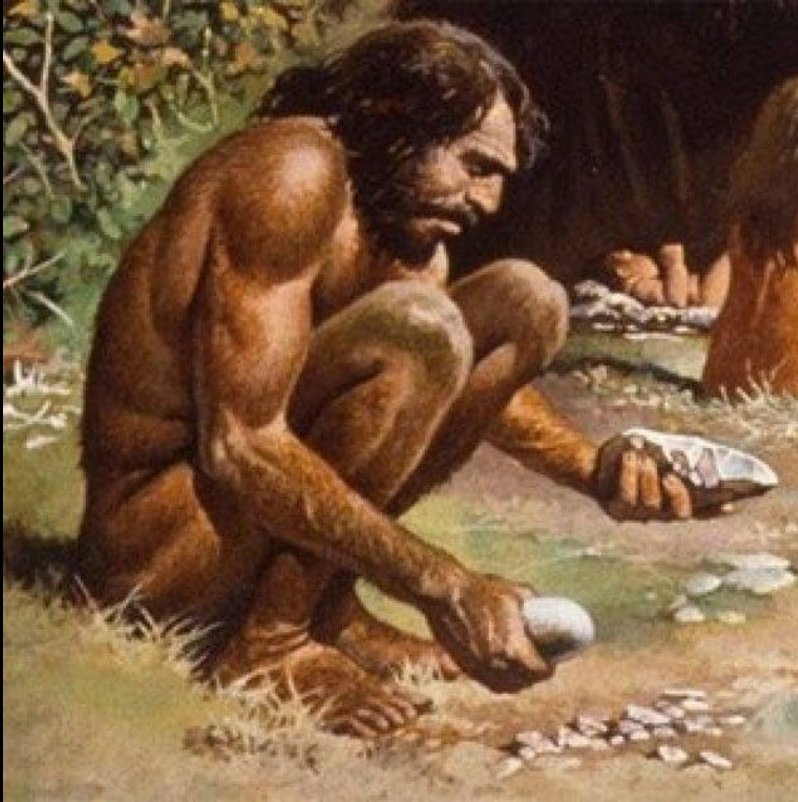
54

This is how I program now...

```
48
49  const TRANSCRIPT_SUMMARY = `Provide a one or two sentence summary of the
50     following segment of a podcast transcript. Only use the information
51     provided in the text; DO NOT use any information you know about the world.
52     Include the title of the podcast, the name of the episode, and the
53     names of the speakers, if known.`;
54
```

INTERCAL has many other features designed to make it even more aesthetically unpleasing to the programmer: it uses statements such as "READ OUT", "IGNORE", "FORGET", and modifiers such as "PLEASE". This last keyword provides two reasons for the program's rejection by the compiler: if "PLEASE" does not appear often enough, the program is considered insufficiently polite, and the error message says this; if it appears too often, the program could be rejected as excessively polite.

Bro, do you even CoPilot?



Programmer without CoPilot,
ca 2020 (colorized)

Bro, do you even CoPilot?



Programmer with CoPilot,
ca 2021

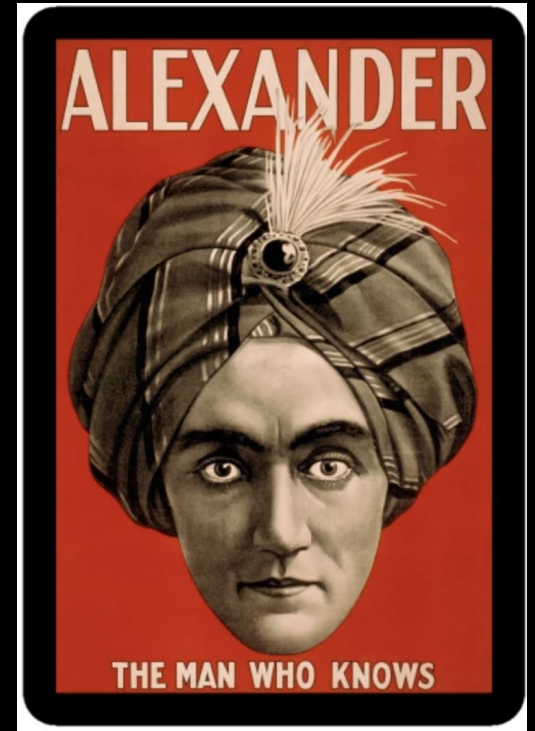
CoPilot is a tectonic shift

CoPilot will **READ YOUR MIND**

Just start typing, it finishes the job

CoPilot **KNOWS ALL**

Not sure how to do something? CoPilot it!



CoPilot is just the beginning

The only thing stopping CoPilot from getting really, *really* good is more data and more compute.

Both of which are in abundance.



I want to use the Deepgram Python SDK to transcribe an MP3 file. Can you show me how to do this?



As of my knowledge cutoff in September 2021, Deepgram provides a Python SDK that you can use to transcribe audio files. However, they don't provide explicit support for MP3 files, so you'll need to convert your MP3 file to a format that Deepgram supports, such as WAV or FLAC.



Here's a basic example of how you might use the Deepgram SDK to transcribe a WAV file:

python

 Copy code

```
from deepgram import Deepgram
import json

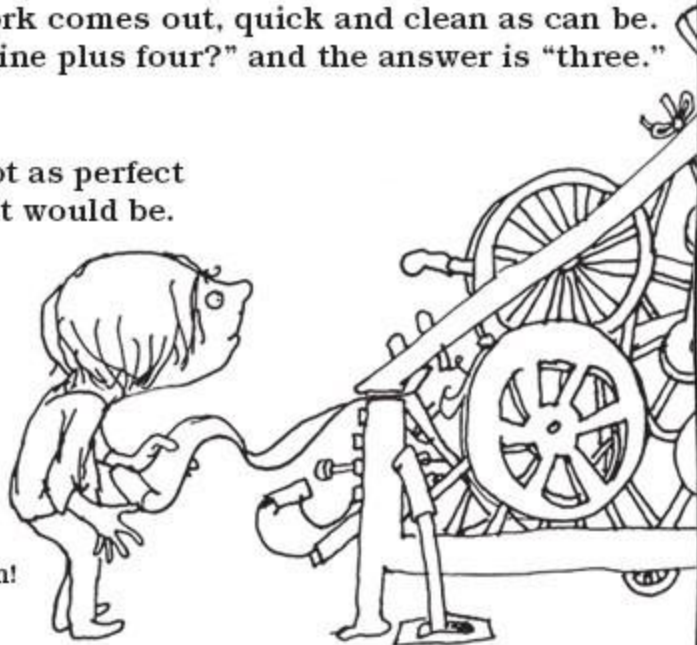
# Initialize the Deepgram client
client = Deepgram('YOUR_DEEPGRAM_API_KEY')

# Transcribe the audio file
response = client.transcription.speech(file='path_to_your_file.wav')
```

HOMWORK MACHINE

The Homework Machine, oh the Homework Machine,
Most perfect contraption that's ever been seen.
Just put in your homework, then drop in a dime,
Snap on the switch, and in ten seconds' time,
Your homework comes out, quick and clean as can be.
Here it is—"nine plus four?" and the answer is "three."
Three?
Oh me . . .
I guess it's not as perfect
As I thought it would be.

Read more
poems in
*A Light in
the Attic*
by Shel Silverstein!

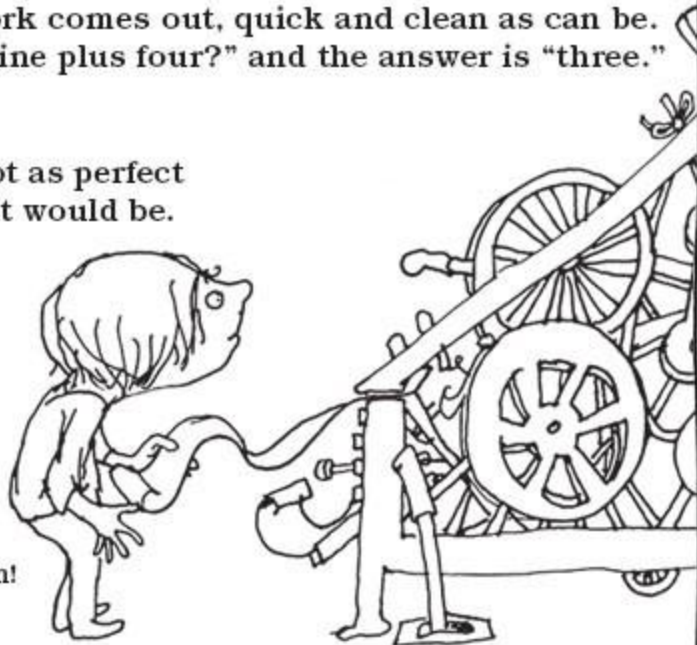


A Light in the Attic © 1981 Evil Eye Music, Inc.

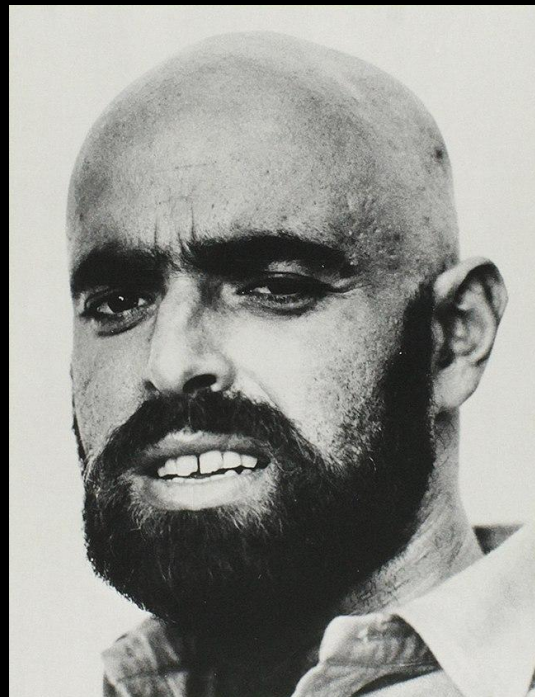
HOMWORK MACHINE

The Homework Machine, oh the Homework Machine,
Most perfect contraption that's ever been seen.
Just put in your homework, then drop in a dime,
Snap on the switch, and in ten seconds' time,
Your homework comes out, quick and clean as can be.
Here it is—"nine plus four?" and the answer is "three."
Three?
Oh me . . .
I guess it's not as perfect
As I thought it would be.

Read more
poems in
*A Light in
the Attic*
by Shel Silverstein!



A Light in the Attic © 1981 Evil Eye Music, Inc.



How much does it cost to replace one human with AI?

Typical SWE salary: \$220,000

Benefits, taxes, free breakfast, lunch, dinner, snacks,
masseur, shuttle bus, on-site doctor, bowling alley, ...

\$92,000

Total: \$312,000

Number of working days per year: 260

Total cost for one-human-SWE-day: \$1200

How much does it cost to replace one human with AI?

Let's do the math!

Average lines of code checked in per day ≈ 100

Average number of GPT-3 tokens per line ≈ 10

Price for GPT-3 = $\$0.02 / 1K$ tokens

How much does it cost to replace one human with AI?

Let's do the math!

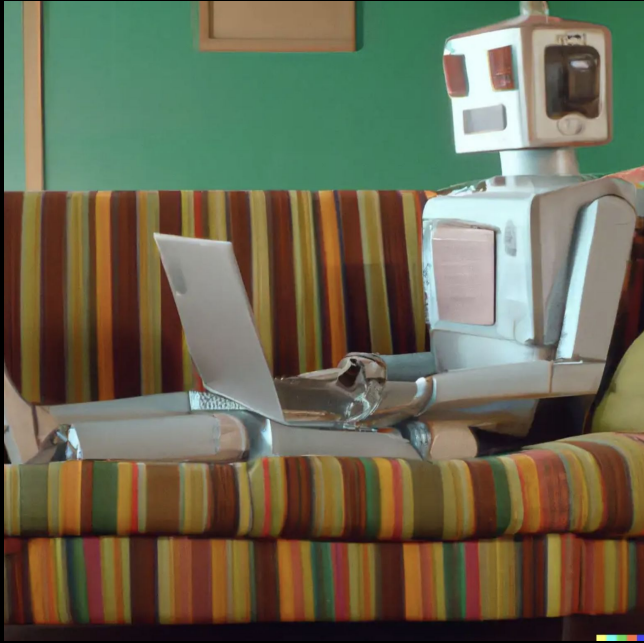
Average lines of code checked in per day ≈ 100

Average number of GPT-3 tokens per line ≈ 10

Price for GPT-3 = $\$0.02 / 1K$ tokens

Total cost for one-human-SWE-day equivalent work: $\$0.12$

How much does it cost to replace one human with AI?



\$0.12 / day

How much does it cost to replace one human with AI?

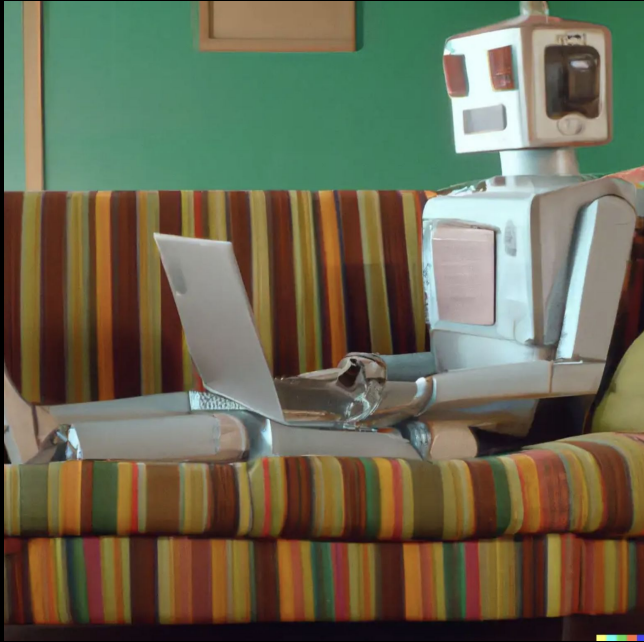


\$0.12 / day



\$1200 / day

How much does it cost to replace one human with AI?



The robot **does not** take breaks.

The robot **does not** require catered lunches or on-site massage.

The robot **takes the same length of time** whether it's a prototype or final production code.

The robot makes plenty of mistakes, but makes them **incredibly quickly**.

Cutting humans out of the loop



The age old question

Presenting ... The software team of the future

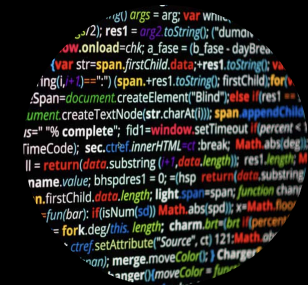
Human code reviewer



Red-blooded PM
(still human)

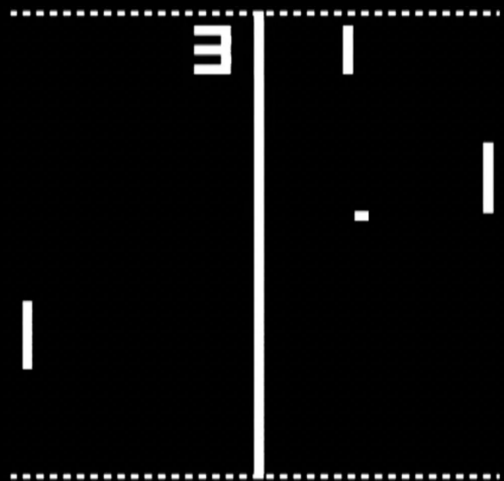


AI robot
code monkey



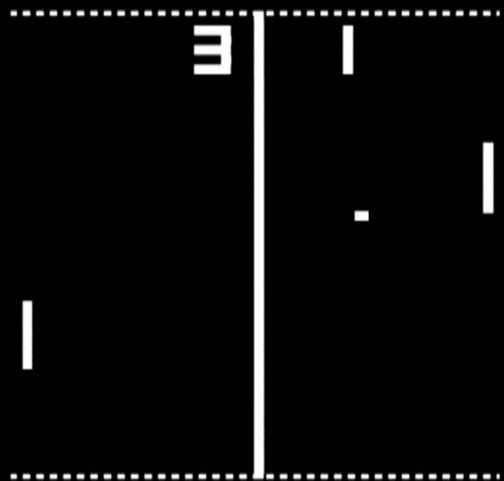
AI-generated code

Why is everyone freaking out about ChatGPT?



Pong
(1972)

Why is everyone freaking out about ChatGPT?

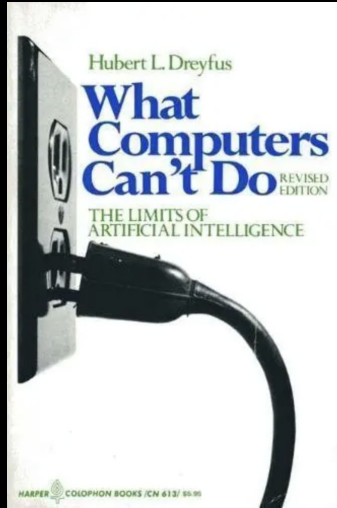


Pong
(1972)

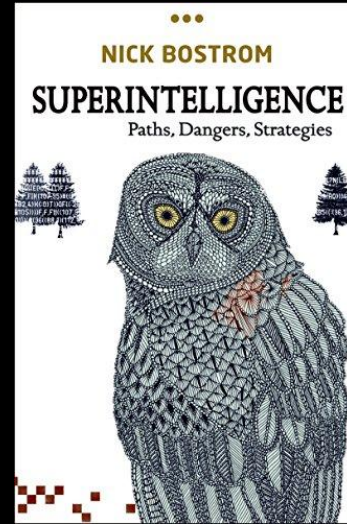


Red Dead Redemption 2
(2018)

Why is everyone freaking out about ChatGPT?



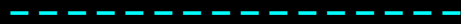
What Computers Can't Do
(1972)



Superintelligence
(2014)

The Evolution of Programming

The Dawn of Time



Humans directly write
machine instructions



The Evolution of Programming

Early Prehistory

Humans write in high level
languages, computers compile
to machine instructions



Bjarne Stroustrup, looking cool

The Evolution of Programming

Modern Times

Humans write in high-level languages, assisted by AI

```
def generate_rsa_key_pair() -> Tuple[str, str]:  
    """Generate a 2048-bit RSA public/private key pair."""  
    key = RSA.generate(2048)  
    private_key = key.export_key().decode("utf-8")  
    public_key = key.publickey().export_key().decode("utf-8")  
    return private_key, public_key
```

CoPilot doing my job for me

The Evolution of Programming

The Future

Humans teach AI models how to solve problems **directly**





I would like you to help me solve a puzzle. There are three stacks of cards on a table. In the first stack is a red card, a blue card, and a green card. In the second stack is two blue cards and a red card. The third stack has two green cards and a blue card. I would like you to tell me, step by step, how to order the cards so that there is one stack of red cards, one stack of blue cards, and one stack of green cards. Please give me each step one at a time.

Let's think step by step.



I would like you to help me solve a puzzle. There are three stacks of cards on a table. In the first stack is a red card, a blue card, and a green card. In the second stack is two blue cards and a red card. The third stack has two green cards and a blue card. I would like you to tell me, step by step, how to order the cards so that there is one stack of red cards, one stack of blue cards, and one stack of green cards. Please give me each step one at a time.

Let's think step by step.



1. Take the blue card from the first stack and place it in its own separate stack for blue cards.
2. Take the red card from the second stack and place it in the first stack with the red card.
3. Take the two blue cards from the second stack and place them in the stack for blue cards.
4. Take the blue card from the third stack and place it in the stack for blue cards.
5. Take the two green cards from the third stack and place them in their own separate stack for green cards.
6. Take the green card from the first stack and place it in the stack for green cards.



Now, you have one stack for red cards, one stack for blue cards, and one stack for green cards.

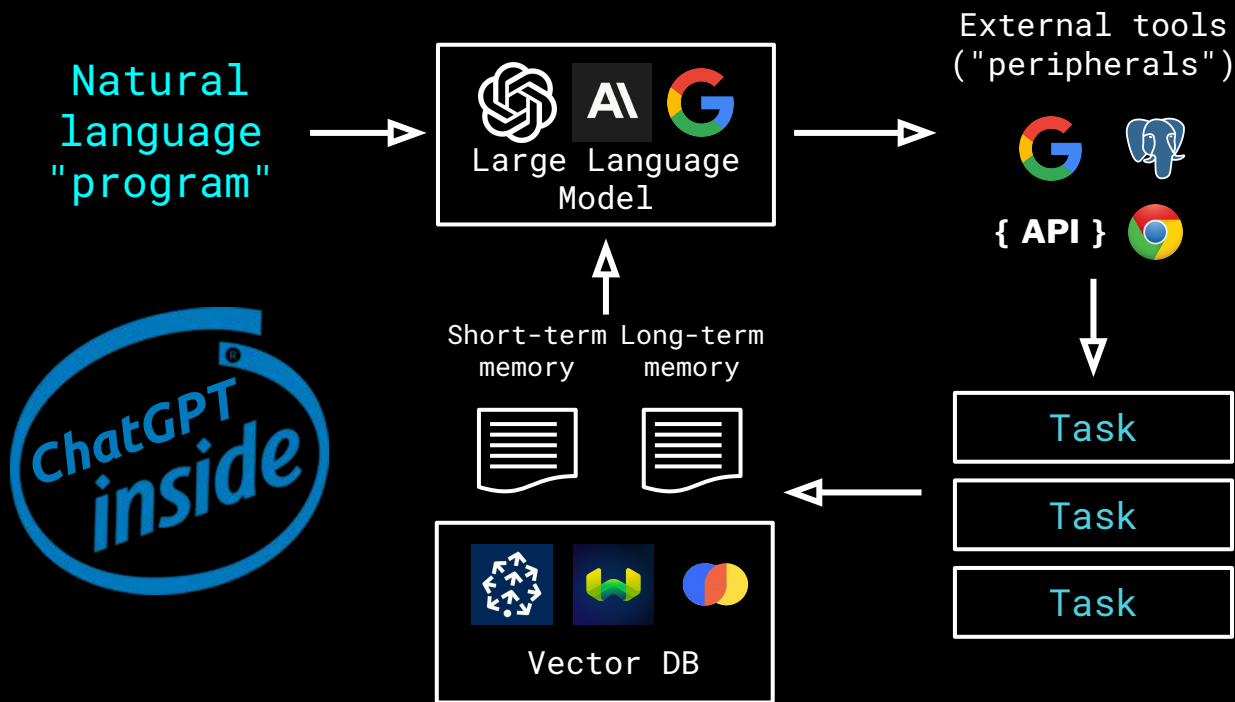
Teaching, not programming

Gradually, `programming` gets replaced by `teaching` models new skills:

- How to interface to an API
- How to pull data from a database
- How to transform data
- How to use software meant for humans

The Natural Language Computer

A new kind of computational substrate!



**AND NOW, A WORD
FROM OUR SPONSORS**

BUILDING LLM APPS WITH **FIXIE**

Fixie is a developer platform that makes it easy to build rich, conversational AI experiences

Built around **AI.JSX**, an open source framework for LLM apps

Ingest data, build an agent, deploy to a website or app in minutes - rather than months

The screenshot displays the Fixie web interface for configuring an agent named 'mdw/fox-agent'. The interface includes a navigation bar with 'Dashboard', 'Documents', and 'Agents' (the active tab), along with links to 'Documentation', 'GitHub', and 'Discord'. Below the navigation, there are tabs for 'Configuration', 'README', and 'Revisions'. The main content area shows a chat log where a user asks about foxes, and the agent responds with information about fox species and a function call to 'LookUpFromKnowLedgeBase'. The right sidebar contains settings for the agent, including a 'System Prompt' (describing the agent as a helpful fox expert), 'Inline UI components' (with toggles for 'Show citations' and 'Show suggested searches'), a 'Greeting Message' (with a toggle to 'Display Greeting message'), and the 'Model' set to 'GPT-4-32k'. A 'Save changes' button is at the bottom of the sidebar. At the bottom of the chat area, there is a 'Send a message.' input field and a 'Regenerate response' button.

Integrating code and natural language

AI.JSX Hello World

```
function MyChatApp() {  
  return (  
    <ChatCompletion>  
      <UserMessage>Write a Shakespearean sonnet about AI models.</UserMessage>  
    </ChatCompletion>  
  );  
}
```

Constraining output through composition

```
function KidSafe({ children }: { children: Node }) {
  return (
    <ChatCompletion>
      <SystemMessage>Rewrite the user's message so that it is safe for kids.</SystemMessage>
      <UserMessage>{children}</UserMessage>
    </ChatCompletion>
  );
}

const app = (
  <KidSafe>
    <MakeSetting>A fantasy world full of magical creatures and floating islands.</MakeSetting>
  </KidSafe>
);
```




RAG in 10 lines of code

```
● ● ●  
<ChatCompletion>  
  <SystemMessage>Use the following information to answer the user's query:  
    <DocsQA corpus={corpusId}>  
      {query}  
    </DocsQA>  
  </SystemMessage>  
  <UserMessage>  
    {query}  
  </UserMessage>  
</ChatCompletion>
```

Embed an AIJSX App using React

```
import { FloatingFixieEmbed } from 'fixie/web';

export default function MyReactApp() {
  return (
    <>
      Rest of page here...
      <FloatingFixieEmbed agentId='mdw/fox-agent' />
    </>
  );
}
```

mdw/fox-agent

 New

 Hey there! 🦊 I'm your go-to for all things fox-related. Ready to dive into some foxy facts? Let's make it lit! 🔥



What do foxes eat?



Foxes are omnivores, which means they eat a variety of foods. Their diet can include rabbits, mice, ground squirrels, insects, and lizards, but this can depend on the seasons and what food is readily available. During the summer, adult foxes will eat a vast amount of insects, beetles, and grasshoppers. They also feed their babies larger prey such as deer and carrion that have already been killed by another predator.

Foxes are opportunistic eaters, consuming anything they can easily seize. This can include fish, tiny birds, rotten meat, skunks

 Regenerate response

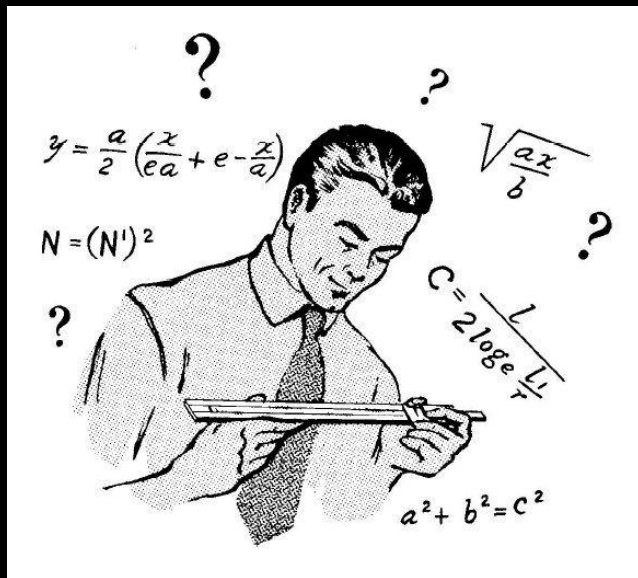
Send a message.



it comes to vegetation. They are also known to eat prickly

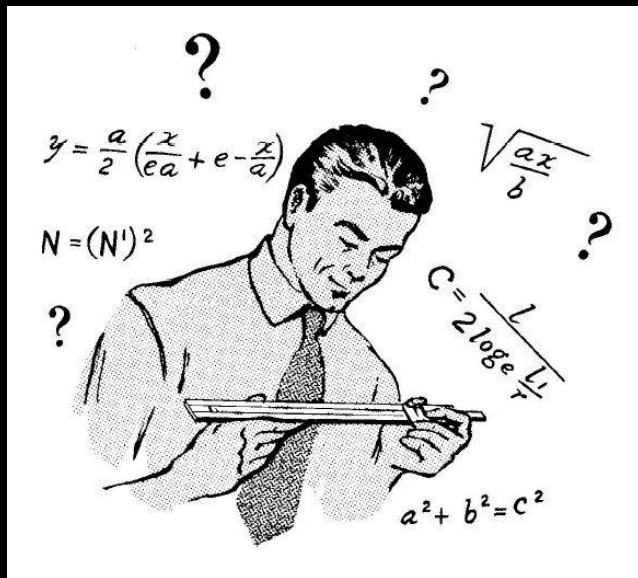
**WE NOW RETURN YOU TO YOUR
REGULARLY SCHEDULED TALK**

Evolving Computer Science



Slide rule
1859-1975

Evolving Computer Science



Slide rule
1859-1975



Computer science
1959-2030

Evolving Computer Science

Over time, CS looks more like EE: A more technical skill set necessary in some very specialized occupations.

The vast majority of people building “software” will not be programming: they will be interacting with an AI.

AI greatly expands access to computing to anyone who can express themselves in natural language.

Evolving Computer Science

The **network** is the **computer**.

-- John Gage, 1984

Evolving Computer Science

The `network` is the `computer`.

-- John Gage, 1984

The `model` is the `computer`.

-- Matt Welsh, 2023

Challenges ahead...

Dirty secret: **Nobody** understands how large AI models work.

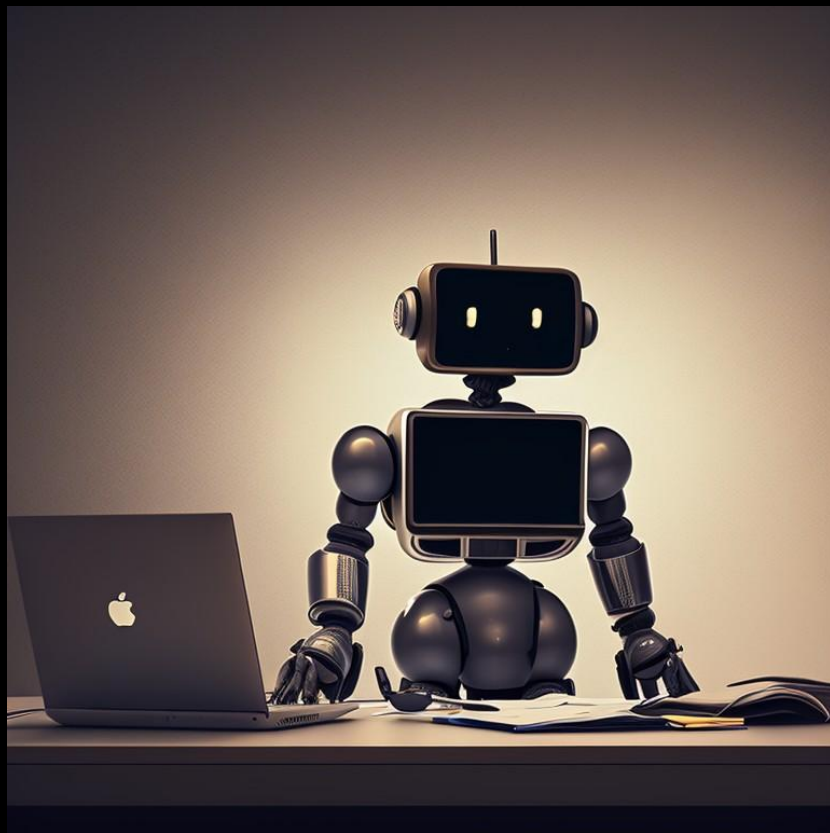
“Chain of thought” reasoning in LLMs was **discovered empirically**, not designed into the model architecture!

The silver lining

Writing code sucks anyway!

Let the robots do it.

Go and have a good life.





`matt@fixie.ai`