



Utrecht University

OCTOBER 2023



LLMs in het programmeeronderwijs

Johan Jeuring

gebaseerd op een presentatie van Hieke Keuning
 A small icon of a presentation slide with a grid pattern.



Wie ben ik?

Hoogleraar Softwaretechnologie voor Leren en Onderwijs bij Informatica en het Freudental Instituut van de Universiteit Utrecht

Een dag per week betrokken bij het Nationaal Onderwijslab A@ (groeifondsproject)

Onderzoeksthema's: feedback en hints voor leren programmeren; intelligente tutoring systemen voor programmeren, wiskunde ...



Informele observaties in het programmeeronderwijs

Plotseling excellente cijfers voor een toets waarbij leerlingen toegang tot het internet hebben

Leerlingen programmeren zonder problemen met ChatGPT, totdat ze een complexe programmeeropgave moeten oplossen

Niets nieuws onder de zon¹, versus we moeten alles anders doen²



"Five students asking the computer questions in colorful, digital art" (DALL-E)

¹<https://medium.com/bits-and-behavior/large-language-models-will-change-programming-a-little-81445778d957>

²<https://medium.com/bits-and-behavior/large-language-models-will-change-programming-a-lot-5cfe13afa46c>

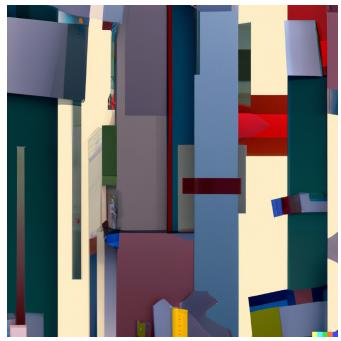
Onderwerpen



Achtergrond:
Generatieve AI(LLMs) voor
programmeren



Toepassingen van LLMs in het
programmeeronderwijs



Achtergrond:
Generatieve AI(LLMs) voor
programmeren en het onderwijs

Generative AI

*“Generative artificial intelligence (AI) describes algorithms (such as ChatGPT) that can be used to create new content, including audio, **code**, images, text, simulations, and videos.*

Recent new breakthroughs in the field have the potential to drastically change the way we approach content creation.”



DALL-E image credits in notes

Generative Pre-trained Transformer: GPT-4

- Large Language Model (LLM).
- Voorspelt het volgende token gebaseerd op vorige tokens.
- Getraind op grote datasets.
- Gefinetuned mbv *Reinforcement Learning from Human Feedback*.

Sinds GPT-3 weinig technische details.

Codex: GPT-3 gefinetuned op Github repo's

GPT-3 training data		
Dataset	# tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%



ChatGPT

H

How do I write a Python program to calculate the average of a list of numbers?



Here is one way to write a Python program to calculate the average of a list of numbers:

Copy code

```
numbers = [1, 2, 3, 4, 5]

sum = 0
for number in numbers:
    sum += number

average = sum / len(numbers)
print(average)
```

This code creates a list of numbers, then uses a for loop to add up the numbers in the list. The `len()` function is used to get the number of items in the list, and the sum is divided by that number to get the average. The result is then printed using the `print()` function.



GitHub
Copilot

Untitled-1.py 1 ● C# code.cs 0 ● J clock.java 4 ● C# code.cs

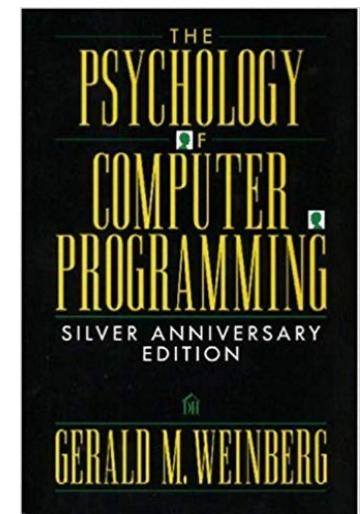
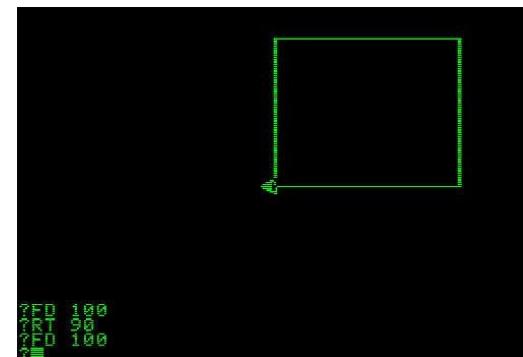
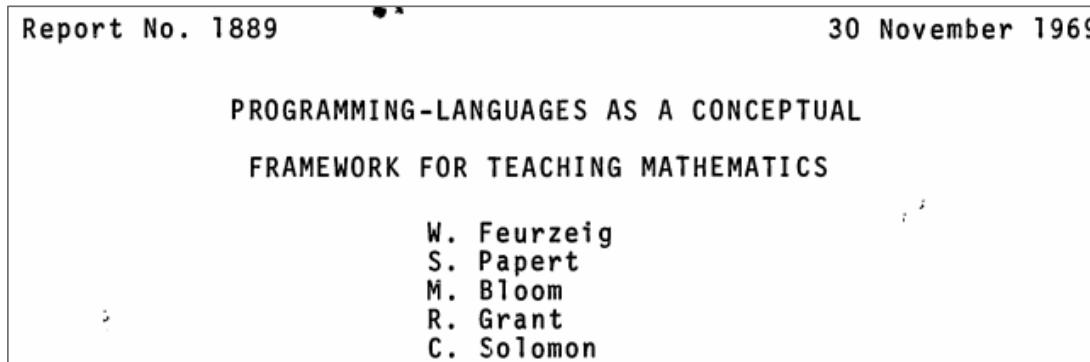
C: > Users > Keuni001 > surfdrive > Shared > MSO > Tentamens > oefen > C# code.cs

```
1 class TicTacToe
2 {
3     public static void Main()
4     {
5         var game = new Game();
6         game.Play();
7     }
8 }
9
public class Game
{
    public void Play()
    {
        var board = new Board();
        var player1 = new Player("X");
        var player2 = new Player("O");
        var currentPlayer = player1;

        while (true)
        {
            board.Print();
            Console.WriteLine("Player {0} turn", currentPlayer.Symbol);
            var move = Console.ReadLine();
            board.MakeMove(move, currentPlayer.Symbol);
            currentPlayer = currentPlayer == player1 ? player2 : player1;
        }
    }
}
10
11
12
```

Computing Education Research (CEdR)

- Sinds de zestiger jaren
- **Programmeren voor beginners**: één van de hoofthema's¹



¹Luxton-Reilly, A., et al. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*.

Eerste artikel in Februari 2022

The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming

James Finnie-Ansley

The University of Auckland

Auckland, New Zealand

james.finnie-ansley@auckland.ac.nz

Paul Denny

The University of Auckland

Auckland, New Zealand

paul@cs.auckland.ac.nz

Brett A. Becker

University College Dublin

Dublin, Ireland

brett.becker@ucd.ie

Andrew Luxton-Reilly

The University of Auckland

Auckland, New Zealand

a.luxton-reilly@auckland.ac.nz

James Prather

Abilene Christian University

Abilene, Texas, USA

james.prather@acu.edu

Mary has been collecting daily rainfall data and has stored the data in a list. She wants to calculate the average of the rainfall data (excluding any negative values) and the maximum. Write a function that returns these two values as a tuple, or -1 if there is no valid data.

```
def rainfall(data):
    total = 0
    count = 0
    max_rain = 0
    for i in range(len(data)):
        if data[i] >= 0:
            total += data[i]
            count += 1
            if data[i] > max_rain:
                max_rain = data[i]
    if count == 0:
        return -1
    else:
        return total / count, max_rain
```

```
def rainfall_data(rainfall):
    valid_data = []
    for i in rainfall:
        if i > 0:
            valid_data.append(i)
    if len(valid_data) == 0:
        return -1
    else:
        return (sum(valid_data)/len(valid_data),
               max(valid_data))
```

Codex lost programmeeropgaven op

In de top 20% op een CS1 tentamen¹

Ongeveer vergelijkbaar met studenten op het Rainfall probleem¹

46,7% op de eerste poging op de 166 problemen van de CodeCheck dataset²

79,5% na **prompt engineering** voor de overige problemen²

¹James Finnie-Ansley et al. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming, ACE'22

²Paul Denny et al. Conversing with Copilot: Exploring prompt engineering for solving cs1 problems using natural language. SIGSCE'23

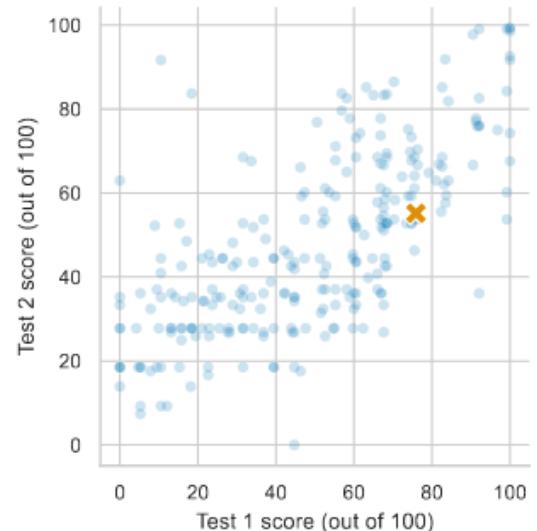
Codex in een datastructuren & algoritmen vak

Codex kan de meeste problemen oplossen

Scoort in het top kwartiel van de groep.

Lijkt beter te scoren op vragen die:

- Meer precies gedefinieerd zijn
- Kort zijn
- Weinig rand gevallen hebben
- Niet vragen om bestaande code aan te passen



COMMUNICATIONS OF THE ACM

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESE.

[Home](#) / [Magazine Archive](#) / [January 2023 \(Vol. 66, No. 1\)](#) / [The End of Programming](#) / [Full Text](#)

VIEWPOINT

The End of Programming

By Matt Welsh

Communications of the ACM, January 2023, Vol. 66 No. 1, Pages 34-35

10.1145/3570220

[Comments \(13\)](#)

VIEW AS:



SHARE:



<https://cacm.acm.org/magazines/2023/1/267976-the-end-of-programming/fulltext>

Moet je nog wel leren programmeren?



“Gezien de ontwikkelingen van machine learning zou het mij niet verbazen als machines straks zichzelf programmeren”



Paul A. Kirschner
@PA_Kirschner

...

A few years ago I argued that the rage in Ed that kids needed to learn to program/code was absurd and a waste of time, money, and effort. I didn't know that ChatGPT and other LLMs would prove this so quickly. Stop with sexy (sounding) fads in Ed and get back to basics.

8:58 AM · Oct 8, 2023 · 36.3K Views

28

36

137

9



Ja!

Veelgehoord misverstand

Analogie:

rekenen versus rekenmachines
feitenkennis versus het Internet
talen versus automatische vertalers

Je moet kunnen beschrijven wat je wilt programmeren

LLMs zullen altijd onbetrouwbaar blijven



Johan Jeuring

@johanjeuring

...

(1/10) All too often I see statements of the form: "since LLMs like ChatGPT can generate code, kids don't need to learn programming anymore". Only a couple of days ago @P_A_Kirschner claimed this again, but you see it more. I think such a statement is misleading.

3:58 PM · Oct 11, 2023 · 4,751 Views

View post engagements

3

9

26

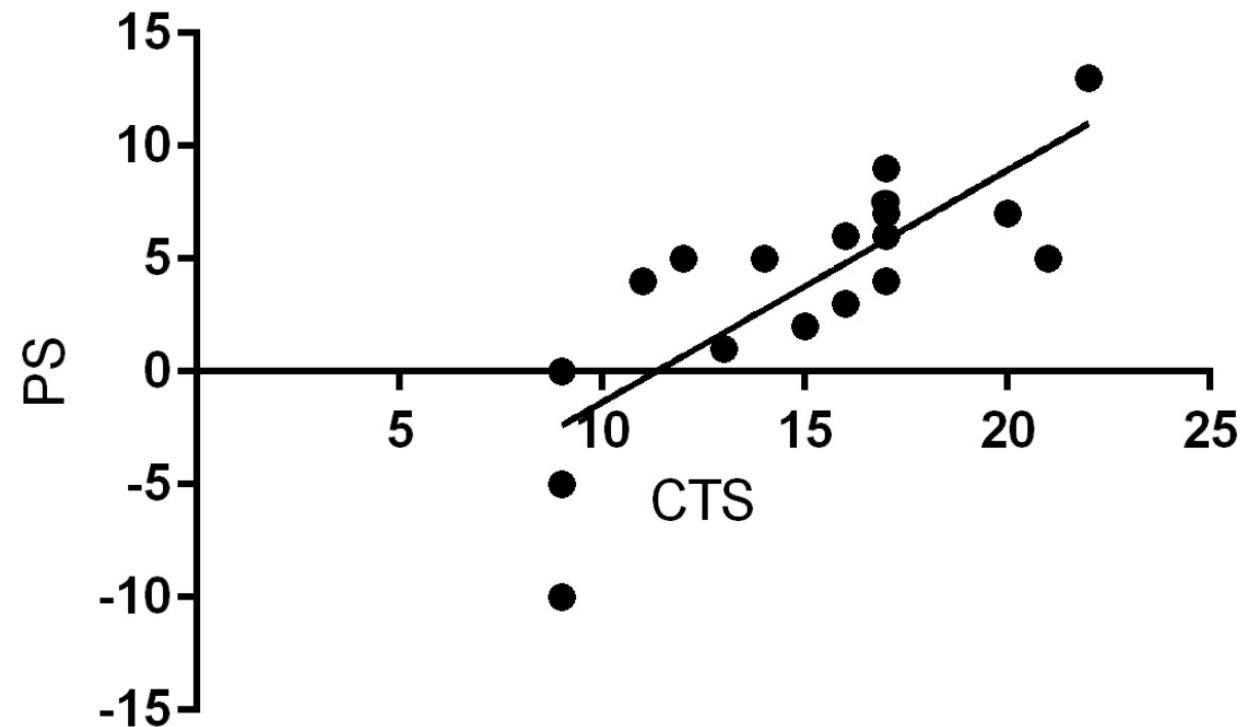
2

Een experiment

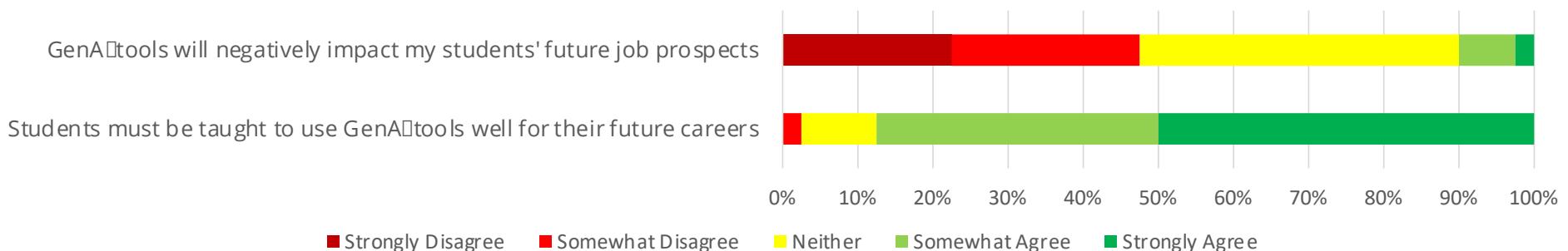
19 mensen met een
zonder voorkennis van
programmeren:

Doe een Computational
Thinking test

Ontwikkel een
programma met
ChatGPT



UU ICS docenten: curriculum



- Betrouwbaarheid en beperkingen van GenAI
- Prompt engineering
- University Guidelines seminar
- Benadruk onderdelen die niet ondersteund worden door GenAI
- Controleren en repareren van gegenereerde code
- Ethische discussies rondom: copyright, energiegebruik, academische integriteit, ...

Wat vinden Computing educators (Becker et al.)?

*"A-generated code presents both **opportunities** and **challenges** for students and educators in introductory programming and related courses. The sudden viability and ease of access to these tools suggest **educators may be caught unaware or unprepared** for the significant **impact** on education practice resulting from A-generated code. We therefore urgently need to **review our educational practices** in the light of these new technologies."*

Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of ACode Generation. Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE '23). <https://arxiv.org/pdf/2212.01020.pdf>



Toepassingen van LLMs in het
programmeeronderwijs

Tekstboek

Learn AI-Assisted Python Programming

Leo Porter and Daniel ...

 add to cart

search in this book

1 INTRODUCING AI-ASSISTED PROGRAMMING WITH COPILOT ▾

2 GETTING STARTED WITH COPILOT ▾

3 DESIGNING FUNCTIONS ▾

4 READING PYTHON CODE - PART 1 ▾

5 READING PYTHON CODE - PART 2 ▾

6 TESTING AND PROMPT ENGINEERING ▾

7 PROBLEM DECOMPOSITION ▾

8 DEBUGGING AND BETTER UNDERSTANDING YOUR CODE ▾

9 AUTOMATING TEDIOUS TASKS ▾

10 MAKING SOME GAMES ▾

11 FUTURE DIRECTIONS ▾

1 Introducing AI-Assisted Programming with Copilot

This chapter covers

- How AI assistants change how new programmers learn
- Why programming is never going to be the same
- How AI assistants like Copilot work
- How Copilot solves introductory programming problems
- Possible perils of AI-assisted programming

In this chapter, we'll talk about how humans communicate with computers. We'll introduce you to your *AI Assistant*, Copilot, an amazing tool that uses Artificial Intelligence (AI) to help people write software. More importantly, we'll show you how Copilot can help you learn how to program. We're not expecting that you've written any programs before. If you have, please don't skip this chapter, even if you already know a little bit about programming. Everyone needs to know why writing programs is different now that we have AI assistants like ChatGPT and Copilot, and that the skills we need to be effective programmers change. As we'll see, we also need to be vigilant, because sometimes tools like ChatGPT and Copilot lie.

1.1 How we talk to computers

Would you be happy if we started by asking you to read and understand this:^[1]

```
1 section .text
2 global _start
3 _start:
4     mov ecx, 10
5     mov eax, '0'
6     l1:
7     mov [num], eax
8     mov eax, 4
9     mov [num], eax
```

[read in livebook](#)

Wat gebruiken jullie al?

Mogelijkheden

Genereren van oplossingen voor programmeerproblemen

Genereren van leermaterialen (opgaven, uitleg van code, feedback)

Nieuwe didaktische aanpakken (e.g. algoritmisch denken)

Genereren van oplossingen voor programmeerproblemen

Copy-paste de opgave

Controleer het resultaat

Experiment met CodeCheck: <https://codecheck.io/>

<https://codecheck.io/files/wiley/codecheck-python-SumsAveragesProducts-1>

<https://codecheck.io/files/wiley/codecheck-python-SumsAveragesProducts-14>

Subgoals in het programmeeronderwijs

"Aiding students to learn the **subgoals** for solving problems has helped them focus on the structure of a procedure and reduced cognitive load..."¹

LLMs resultaten worden beter wanneer je vraagt een probleem **stap-voor-stap** op te lossen.

Vraag: zou een subgoal-gebaseerde prompt engineering aanpak voor het genereren van oplossingen voor programmeerproblemen leiden tot meer correcte oplossingen?

Catrambone, R. (1996). Generalizing solution procedures learned from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22, 1020–1031. doi:10.1037/0278-7393.22.4.1020

Wat is een sub-goal label?

Veel programmeerproblemen bestaan uit onderdelen met ieder een eigen doel. Een subgoal bestaat uit twee of drie delen:

1. Werkwoord
2. Object
3. Optioneel: Conditie

Voorbeeld:

“remember | the element | if the current element is
larger than or equal to the largest element”

Zonder subgoals

```
2 # Given a list of integers, return the position
3 # of the last occurrence of the largest element.
4 # You may assume the given list has at least one element.
5 # Do not use the index, find, or max methods.
6
7 def lastOccur(arr):
8
9     # Assume the first element is the largest
10    largest = arr[0]
11    pos = 0
12
13    # Loop through the list
14    for i in range(len(arr)):
15
16        # If the current element is larger than the largest
17        # element, update the largest element and the position
18        if arr[i] > largest:
19            largest = arr[i]
20            pos = i
21
22    # Return the position of the largest element
23    return pos
```



Met subgoals

```
1
2     # Given a list of integers, return the position
3     # of the last occurrence of the largest element.
4     # You may assume the given list has at least one element.
5     # Do not use the index, find, or max methods.
6     # Step 1: loop through the list
7     # Step 2: remember the element if the current element is larger than or equal to the largest element
8
9 def LastOccur(arr):
10
11     largest = arr[0]
12     index = 0
13     for i in range(len(arr)):
14         if arr[i] >= largest:
15             largest = arr[i]
16             index = i
17     return index
```



Genereren van oplossingen voor programmeerproblemen

<https://codecheck.io/files/wiley/codecheck-python-TwoAnswers-1>

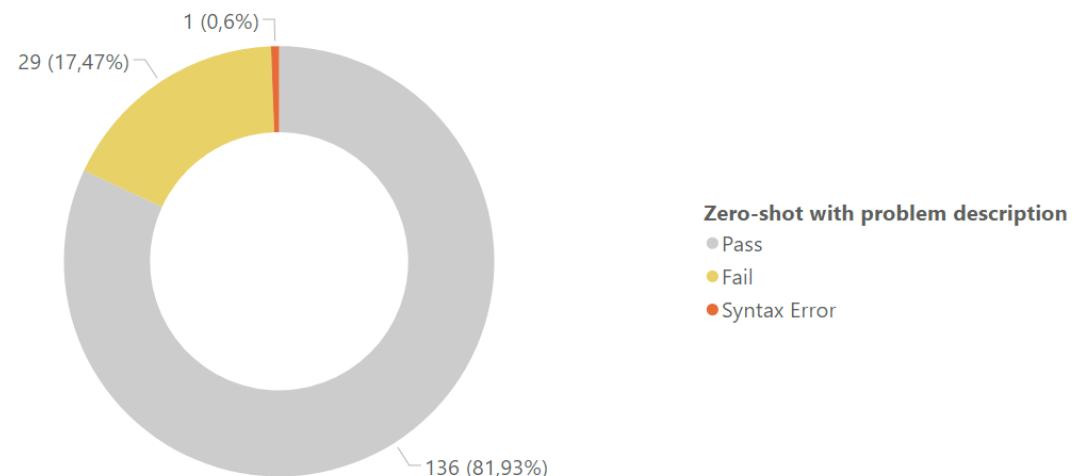
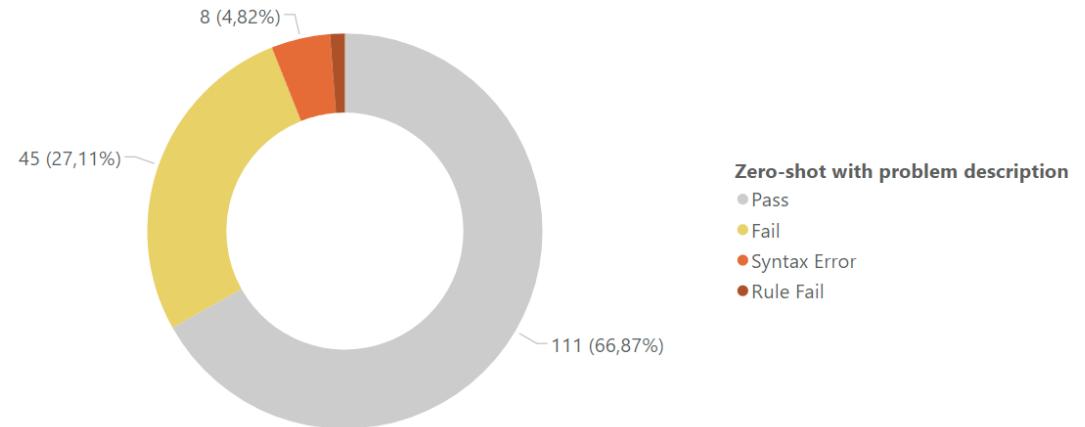
<https://codecheck.io/files/wiley/codecheck-python-ComplexLoops-6>

Subgoals voor Codex

Met subgoals:

- ChatGPT: 1 niet
- Copilot: 2 niet

van de 166



Automatisch programma's repareren

Toegepast op beginnersopgaven voor Python.

Repareert syntax en semantische fouten

Geevalueerd op 286 studentprogramma's,
vergeleken met een state-of-the-art baseline

*Repareert meer programma's en produceert
kleinere patches dan gemiddeld*

```
1 i = input()
2 S = i.lower()
3 l = len(S)
4 if(l%2!=0):
5     B = S[:((l+1)//2)]
6     E = S[:(l+1)//2-1:-1]
7     #print(B,E)
8     if(B==E):
9         print(i,'is_a_'
10            palindrome.')
11    else:
12        print(i,'is_NOT_a_'
13            palindrome.')
14    else:
15        B = S[:l//2]
16        E = S[l//2-1:-1]
17        print(B,E)
18
19
```

(a) Incorrect Program

(b) MMAPR's Repair

Programmeerproblemen genereren

```
""" Exercise 1
--Keywords--
cars
function
parameters
conditional
--Problem statement--
Write a function called speeding_check that takes
    ↪ a single parameter speed and prints out "
    ↪ You are
fined for $200" if the speed is above 120, "You
    ↪ are fined for $100" if the speed is above
    ↪ 100 but
below 120 and otherwise prints "All good, race
    ↪ ahead".
--Sample solution--
def speeding_check(speed):
    if speed > 120:
        return "You are fined for $200"
    elif speed > 100:
        return "You are fined for $100"
    else:
        return "All good, race ahead"
--Tests--
class Test(unittest.TestCase):
    def test_speeding_check(self):
        self.assertEqual(speeding_check(100), 'All
            ↪ good, race ahead')
        self.assertEqual(speeding_check(101), 'You
            ↪ are fined for $100')
        self.assertEqual(speeding_check(121), 'You
            ↪ are fined for $200')
```

```
""" Exercise 2
--Keywords--
ice hockey
--Problem statement--
```



```
Write a function called ice_hockey_check that
    ↪ takes a single parameter
called score and prints out "You are awarded a
    ↪ penalty for unsportsmanlike conduct"
if the score is greater than 5 and otherwise
    ↪ prints "All good, keep playing".
--Sample solution--
def ice_hockey_check(score):
    if score > 5:
        return "You are awarded a penalty for
            ↪ unsportsmanlike conduct"
    else:
        return "All good, keep playing"
--Tests--
class Test(unittest.TestCase):
    def test_ice_hockey_check(self):
        self.assertEqual(ice_hockey_check(5), 'All
            ↪ good, keep playing')
        self.assertEqual(ice_hockey_check(6), 'You
            ↪ are awarded a penalty for unsportsmanlike
            ↪ conduct')
```

Resultaten

Analyse (met de hand) van 120 gegenereerde opgaven:

- 75.0% waren passend
- 81.8% waren nieuw
- 76.7% hadden een voorbeeldoplossing

Analyse (automatisch) van 240 gegenereerde opgaven:

- 70.8% hebben testcases
- Voor 30.9% slagen alle testen

*Codex zou nuttig kunnen zijn voor het genereren van opgaven,
maar goede kwaliteitscontrole is nodig.*

Code uitleggen

```
print("Hello world!")  
  
""" Step-by-step explanation of the above program:  
1.
```

Van 20 uitgelegde programma's worden in 18 gevallen alle onderdelen uitgelegd

174 regel-voor-regel uitleg, waarvan 67.2% correct

Codex produceert zelden uitleg op abstract niveau.

Compiler error messages

Codex Example 1 (input in black, output in red)

```
""" Code
def check_password(password, input):
    return password == input
input = "hunter2"
if check_password("s3cr37", input):
    """ Output
    File "main.py", line 5

    ^
SyntaxError: unexpected EOF while parsing
""" Plain English explanation of why does running
→ the above code cause an error and how to fix the
→ problem
The error is caused because the block of code is
→ expecting another line of code after the colon.
To fix the issue, I would add another line of code
→ after the colon.
```

Analyse van de extra uitleg:

- Begrijpelijk 88%
- Heeft uitleg 84%
- Correct op 48% van alle inputs
- 33% met een correcte fix

Uitleg en reparaties zijn nog niet goed genoeg voor gebruik, en introduceren risicos wanneer studenten ze blind opvolgen. Mogelijk wordt het nog beter...

Automatische next-step hints mbv LLMs

Feedback en hints zijn belangrijk voor leren

Automatisch next-step hints genereren is vaak moeilijk

LLMs direct gebruiken leidt vaak niet tot de beste hulp

L

Can you help me solve this Fibonacci assignment:



```
def fib(n):
```



Sure, I can help you with your Fibonacci assignment. Here's an implementation of the Fibonacci sequence using recursion:



Copy code

```
python
def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n-1) + fib(n-2)
```

The `fib` function takes an integer `n` as input and returns the `n`th number in the Fibonacci sequence. The Fibonacci sequence is a series of numbers in which each number is the sum of the two preceding ones, usually starting with 0 and 1. So the first few numbers in the sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on.

The `fib` function uses recursion to calculate the `n`th number in the sequence. If `n` is less than or equal to 1, the function simply returns `n`. Otherwise, it recursively calls itself with `n-1` and `n-2` as inputs, and adds the results together to get the `n`th number in the sequence.

Regenerate response

Here's an example usage of the function:





StAP Tutor

Choose exercise:

Pies

Restart exercise

Exercise: Pies

A single pie costs A dollars and B cents in the cafe. Calculate how many dollars and cents one needs to pay for N pies.

Input: The program receives three numbers as an input:

A - how many dollars a pie costs;
B - how many cents a pie costs;
N - how many pies do you need to buy.

Output: Print out two numbers: the cost of N pies in dollars and cents.

Hint

Show solution

Check progress

Type code here:

```
1 a = int(input())
```

```
2
```

Feedback: The next step should involve requesting input for the second variable, B - the cost of a pie in cents.

Help! Wat voor feedback geven docenten?

We halen input op

- van een grote groep docenten/experts
- uit verschillende landen
- over wanneer, waarom, en hoe feedback te geven,
- op verschillende reeksen van stappen die studenten zetten in een online omgeving



Andere studies?

Mogelijkheden en beperkingen van LLMs bepalen (35)

Positiepapers en surveys/interviews (17)

Interacties tussen programmeurs en LLMs (9)

LLMs gebruiken voor analyse van studentwerk (5)

LLMs gebruiken voor genereren van onderwijsmateriaal (5)

Hellas et al.	Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests	arXiv	2023
Idialu et al.	Whodunnit: Human or AI?	-	2023
Jaipersaud et al.	Decomposed Prompting to Answer Questions on a Course Discussion Board	AI in Education	2023
Jalil et al.	ChatGPT and Software Testing Education: Promises & Perils	IEEE ICSTW	2023
Kazemitaabari et al.	Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming	CHI	2023
Kendon et al.	AI-Generated Code Not Considered Harmful	WCCCE	2023
Kiesler and Schiffner	Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments	arXiv	2023
Lau and Guo	From 'Ban It Till We Understand It' to 'Resistance is Futile': How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation	ICER	2023
Leinonen et al.	Comparing Code Explanations Created by Students and Large Language Models	arXiv	2023
Leinonen et al.	Using Large Language Models to Enhance Programming Error Messages	SIGCSE TS	2023
Liffiton et al.	CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes	arXiv	2023
Ma et al.	Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI pair Programming	arXiv	2023
MacNeil et al.	Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book	SIGCSE TS	2023
Matelsky et al.	A large language model-assisted education tool to provide feedback on open-ended responses	arXiv	2023
Nam et al.	In-IDE Generation-based Information Support with a Large Language Model	arXiv	2023
Orenstrakh et al.	Detecting LLM-Generated Text in Computing Education: A Comparative Study for ChatGPT Cases	arXiv	2023
Pădurean et al.	Neural Task Synthesis for Visual Programming	arXiv	2023
Pankiewicz and Baker	Large Language Models (GPT) for automating feedback on programming assignments	arXiv	2023
Philbin	Exploring the Potential of Artificial Intelligence Program Generators in Computer Programming Education for Students	Inroads	2023
Phung et al.	Generating High-Precision Feedback for Programming Syntax Errors using Large Language Models	arXiv	2023
Phung et al.	Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors	International J. of Management	2023
Piccolo et al.	Many bioinformatics programming tasks can be automated with ChatGPT	arXiv	2023
Poldrack et al.	AI-assisted coding: Experiments with GPT-4	arXiv	2023
Prather et al.	"It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers	TOCHI	2023
Rajabi et al.	Exploring ChatGPT's impact on post-secondary education: A qualitative study	WCCCE	2023
Reeves et al.	Evaluating the Performance of Code Generation Models for Solving Parsons Problems With Small Prompt Variations	ITiCSE	2023
Ross et al.	A Case Study in Engineering a Conversational Programming Assistant's Persona	ACM IUI	2023
Sandoval et al.	Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants	USENIX	2023
Savelka et al.	Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?	arXiv	2023
Savelka et al.	Large Language Models (GPT) Struggle to Answer Multiple-Choice Questions about Code	arXiv	2023
Savelka et al.	Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses	arXiv	2023
Singla	Evaluating ChatGPT and GPT-4 for Visual Programming	arXiv	2023
Sridhar et al.	Harnessing LLMs in curricular design: Using gpt-4 to support authoring of learning objectives	arXiv	2023
Wang et al.	Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives	VL/HCC	2023
Wermelinger	Using GitHub Copilot to Solve Simple Programming Problems	SIGCSE TS	2023
Widjojo and Treude	Addressing Compiler Errors: Stack Overflow or Large Language Models?	arXiv	2023
Yan et al.	Practical and Ethical Challenges of Large Language Models in Education: A Systematic Literature Review	arXiv	2023
Zan et al.	Large language models meet NL2Code: A survey	Annual Meeting of the ACL	2023
Zastudil et al.	Generative AI in Computing Education: Perspectives of Students and Instructors	arXiv	2023

Uitdagingen¹

Ethische aspecten: academische integriteit, licensing, energiebehoefte

Bias en slechte responses

Over-reliance:

- incorrecte oplossingen
- bemoeilijkt leren

Wat is er open aan slimme chatbot-maker OpenAI?

Artificiële intelligentie OpenAI, het meest succesvolle AI-laboratorium van dit moment, werd opgericht om ervoor te zorgen dat AI „ten goede komt aan de gehele mensheid“. Volgens critici is die missie een rookgordijn voor commerciële belangen.

BUSINESS • TECHNOLOGY

Exclusive: OpenAI Used Kenyan Workers on Less Than \$2 Per Hour to Make ChatGPT Less Toxic

Lawsuit Takes Aim at the Way A.I. Is Built

A programmer is suing Microsoft, GitHub and OpenAI over artificial intelligence technology that generates its own computer code.

AI me to the Moon... Carbon footprint for 'training GPT-3' same as driving to our natural satellite and back

¹Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. Becker, B. et al. (2023). Proceedings of SIGCSE '23.

https://www.nrc.nl/nieuws/2023/01/01/wat-is-er-open-aan-openai-a4153050?utm_campaign=share&utm_medium=social&utm_source=twitter&utm_term=in-byline

<https://time.com/6247678/openai-chatgpt-kenya-workers/>

Discussie



Hoe zouden jullie je onderwijs nog verder willen aanpassen?



Wat zouden we (CEdR) nog verder moeten onderzoeken?



Utrecht University

*Neem graag contact op voor
vervolggesprekken*

Survey:



j.t.jeuring@uu.nl



www.jeuring.nl