Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

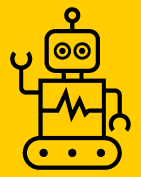
Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Utrecht University

MARCH 2023



ChatGPT and Copilot in programming education:

what do we know from a research perspective?

Hieke Keuning

NIOC 2023

CC BY NC SA

Universiteit Utrecht

dr. Hieke Keuning

Teaching: Assistant professor at Information & Computing Sciences UU, before Software Engineering at HBO.

Research group: Software Technology for Learning and Teaching

Research field & topics: Computer Science Education:

- Tutoring systems and automated feedback generation for learning programming
- Code quality in education
- AI in CS education

Informal observations in CS education

Suddenly top marks for a lab exam compared to the year before…

Students doing fine with ChatGPT, until they have to do a complex programming exercise…

Nothing changes vs. we have to completely change our teaching…

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

Pause Giant AI Experiments: An Open Letter

We call on all AI labs to immediately pause for at least 6 months the training of AI systems more powerful than GPT-4.

Signatures
1123

Add your signature

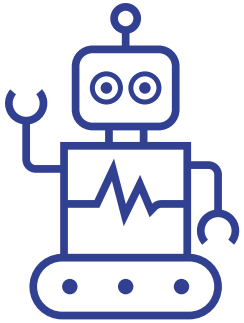
"...**out-of-control race** to develop and deploy ever more powerful digital minds that no one – not even their creators – can understand, predict, or reliably control."

"Powerful AI systems should be developed only once we are confident that their effects will be **positive** and their risks will be **manageable**."

"... we call on all AI labs to immediately **pause** for at least 6 months the training of AI systems more powerful than GPT-4."

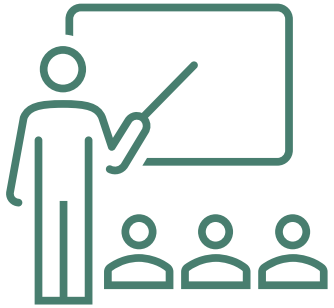
https://futureoflife.org/open-letter/pause-giant-ai-experiments/ [accessed 29 March 13:05]

Topics

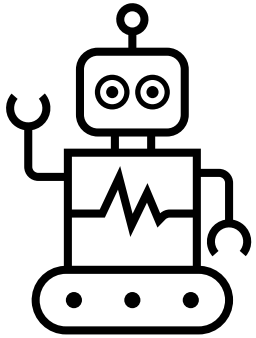


Some background:
The rise of Generative AI for programming

Studies on generative AI in CS education



Implications for CS teaching



Some background: The rise of Generative AI for programming

How familiar are you with these AI programming tools (ChatGPT/Codex, Copilot)?

- ○ I have never heard about them.
- ○ I have heard about them but haven't tried them.
- ○ I tried them a few times.
- ○ I use them regularly.

Generative AI

DALL-E image credits in notes

"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."

https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai

Generative Pre-trained Transformer: **GPT-4**

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%

- Large Language Model (LLM).
- Predicts the next token based on previous tokens.
- Trained on large datasets.
- Fine-tuned using *Reinforcement Learning from Human Feedback.*

Since GPT-3 few technical details.

Codex: GPT-3 finetuned on Github repo's

GPT-4 Technical Report, Open AI, *14 March 23*

ChatGPT

H

How do I write a Python program to calculate the avergae 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

```python
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 | code.cs | clock.java 4

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

```cs
class TicTacToe
{
    public static void Main()
    {
        var game = new Game();
        game.Play();
    }

    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 : player
            }
        }
    }
}
```

"..it has the potential to really revolutionize the way we learn. People are in classrooms of, say, 30 people. Everyone has different backgrounds, ways of learning, and everyone is getting basically the same curriculum. With tools like ChatGPT, you can endlessly converse with a model to understand a concept in a way that is catered to your level of understanding. It has immense potential to help us with personalized education."

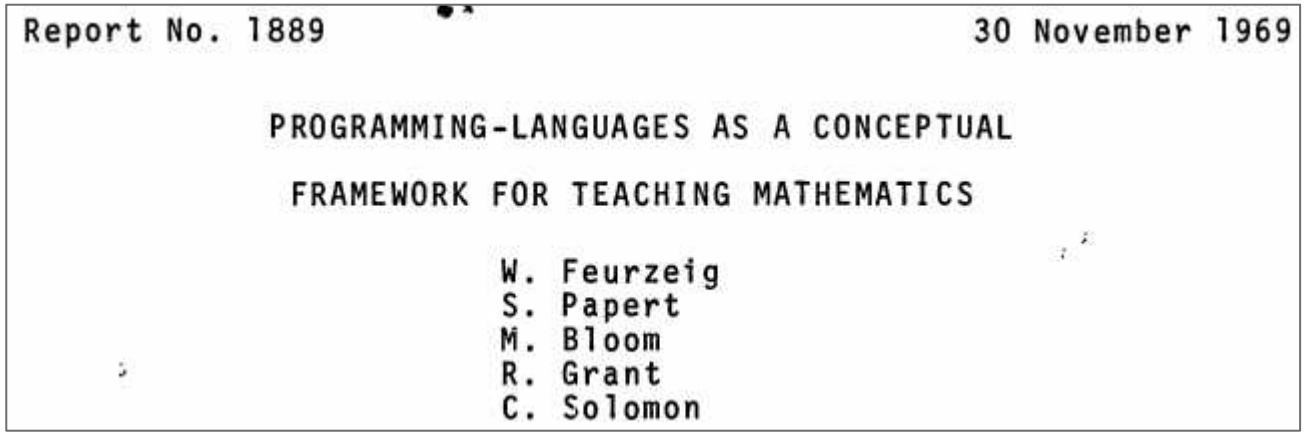
Mira Murati, CTO at OpenAI

https://time.com/6252404/mira-murati-chatgpt-openai-interview/

Studies in a CS education context

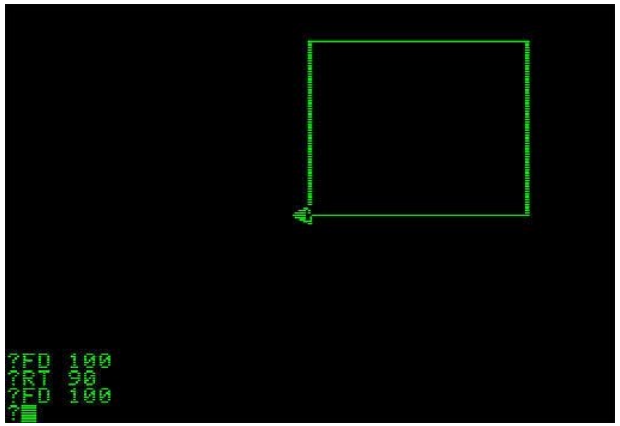
Computing Education Research (CEdR)

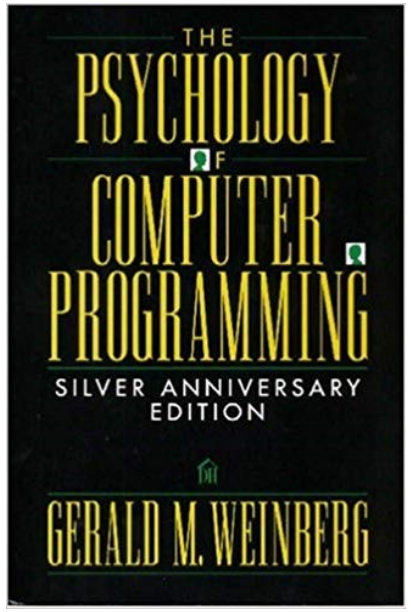
- Since the 1960s
- Computer Science Education ⇨ Computing Education
- **Novice programming**: a major topic[1]


Report No. 1889 30 November 1969

PROGRAMMING-LANGUAGES AS A CONCEPTUAL FRAMEWORK FOR TEACHING MATHEMATICS

W. Feurzeig
S. Papert
M. Bloom
R. Grant
C. Solomon





Luxton-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*.

First blog in August 2022

Emery Berger

Aug 9, 2022 · 7 min read · Member-only · Listen

Coping with Copilot

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

https://itnext.io/coping-with-copilot-b2b59671e516

First paper in February 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.

```python
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
```

```python
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 solving programming exercises

In the top ~20% on a CS1 exam[1]

More or less comparable to students on the Rainfall problem[1]

46,7% on first attempt on the 166 problems of the CodeCheck dataset[2]

79,5% after **prompt engineering** on the remaining problems[2]

[1]James Finnie-Ansley et al. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming, ACE'22
[2]Paul Denny et al. Conversing with Copilot: Exploring prompt engineering for solving cs1 problems using natural language. SIGSCE'23

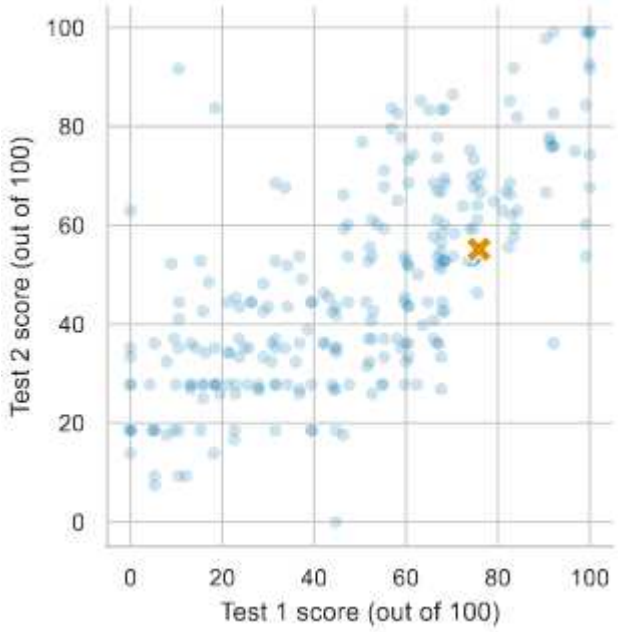
Codex in a datastructures & algorithms course

Codex is able to solve most questions

Performs in the top quartile of the class.

May perform better on questions that are:

- More precisely defined
- succinctly written
- have fewer edge cases
- do not require adapting existing code.



My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. Finnie-Ansley., et al. ACE '23. doi link

Position of Computing educators (Becker et al.)

*"AI-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 AI-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 AI Code 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

Opportunities[1]

Generation of code solutions for learning

Producing learning resources (exercises, code explanations, feedback)

New pedagogical approaches (e.g. algorithmic thinking)

[1]Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. Becker, B. et al. (2023). Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE '23). https://arxiv.org/pdf/2212.01020.pdf

[2]The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. Finnie-Ansley et al., ACE '22. doi link

[3]Conversing with Copilot: Exploring prompt engineering for solving cs1 problems using natural language. Denny et al. SIGSCE '23. link

Automatic program repair

Applied to introductory Python exercises
Fixes syntactic and semantic mistakes

Evaluated on 286 student programs, compared to a state-of-the-art baseline

Can fix more programs and produces smaller patches on average.

```
i = input()
S = lower(i)
l = len(S)
if(l%2!=0)
  B = S[:(l+1)/2]
  E = S[:(l+1)/2:-1]
  print(B,E)
  if(B==E)
    print(i,'is␣a␣
        palindrome.')
  else
    print(i,'is␣NOT␣a␣
        palindrome.')
else
  B = S[:l/2]
  E = S[:l/2:-1]
  print(B,E)
```

(a) Incorrect Program

```
i = input()
S = i.lower()
l = len(S)
if(l%2!=0):
  B = S[:(l+1)//2]
  E = S[:l//2-1:-1]
  #print(B,E)
  if(B==E):
    print(i,'is␣a␣
        palindrome.')
  else:
    print(i,'is␣NOT␣a␣
        palindrome.')
else:
  B = S[:l//2]
  E = S[:l//2-1:-1]
  #print(B,E)
  if(B==E):
    print(i,'is a
        palindrome.')
  else:
    print(i,'is NOT a
        palindrome.')
```

(b) MMAPR's Repair

Zhang et al. Repairing Bugs in Python Assignments Using Large Language Models. arXiv 2022.

Generate programming exercises

```
"""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.assertEquals(speeding_check(100), 'All
    ↪ good, race ahead')
    self.assertEquals(speeding_check(101), 'You
    ↪ are fined for $100')
    self.assertEquals(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.assertEquals(ice_hockey_check(5), 'All
    ↪ good, keep playing')
    self.assertEquals(ice_hockey_check(6), 'You
    ↪ are awarded a penalty for unsportsmanlike
    ↪ conduct')
```

Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. Sarsa et al. ICER '22. doi link

Results

Manual analysis of 120 exercises:

- 75.0% were sensible
- 81.8% were novel
- 76.7% had a matching sample solution

Programmatic analysis of 240 exercises:

- 70.8% have tests
- For 30.9% all tests pass

Codex could be a useful tool to facilitate exercise creation.
However, exercises could rarely be used directly.

Sarsa et al. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. ICER 2022.

Explaining code

```
print("Hello world!")

"""Step-by-step explanation of the above program:
1.
```

- From 20 code explanations, 90% explained all parts
- 174 line-by-line explanations, of which 67.2% correct

Codex very rarely provided high-level descriptions

Sarsa et al. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. ICER 2022.

Enhancing compiler error messages (PEMs)

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.
```

Analysis of explanations:

- Comprehensible 88%
- Has explanation 84%
- Correct 48% of all inputs
- Only 33% included a correct fix

Explanations and fixes are not yet ready for production use, pose a risk if students follow up on faulty messages.
At the same time, LLMs could be a useful tool for improving PEMs.

Using Large Language Models to Enhance Programming Error Messages. Leinonen et al. SIGCSE'23 link

Survey on usage by teachers (prim & sec. school) and students (12-17)

>2000 participants, US

	Teachers	Students
ChatGPT will be an essential tool for students' success in college and the workplace	71%	65%
ChatGPT can help learn more at a faster rate	73%	68%
Report using ChatGPT on a weekly basis or more	51%	22%

https://www.waltonfamilyfoundation.org/chatgpt-used-by-teachers-more-than-students-new-survey-from-walton-family-foundation-finds

Studies at UU

Utrecht University

Our research on A.I. in CS education

With Prof. Dr. Johan Jeuring, and BSc/MSc students:

- How can we use LLMs to generate next-step hints for programming exercises? (Lianne Roest) ⇨
- What is a good approach to prompt engineering for solving programming tasks? (Laurian Wannee) ⇨
- How do Computational Thinking skills and efficient use of ChatGPT relate? (Roel Groot) ⇨
- How good is Codex at program repair compared to state-of-the-art tools? (Sylvain Maissan)
- How is GitHub Copilot used by first-year students when writing object-oriented code? (Mieke Maarse)

Automatic Next-step Hint Generation Using Large Language Models, *Lianne Roest*

Feedback and hints are very effective for learning.

Generating next-step hint used to be complex.

Directly using LLM won't always give the best help.

Goal: Can we use LLMs to generate useful next-step hints?

L

Can you help me solve this Fibonnaci assignment:

def fib(n):

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

python 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:

ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Hint Generation

Generate one hint

- [x] Description
- [ ] Model Solution
- [ ] Example Hint

Exercise

Fibonnaci

Model

gpt-3.5-turbo

Instruction

Give a small next-step hint

Generate one hint

Output

Prompt:
--Description--
The fibonacci sequence is a famous bit of mathematics, and it happens to have a recursive definition. The first two values in the sequence are 0 and 1 (essentially 2 base cases). Each subsequent value is the sum of the previous two values, so the whole sequence is: 0, 1, 1, 2, 3, 5, 8, 13, 21 and so on. Define a recursive fibonacci(n) method that returns the nth fibonacci number, with n=0 representing the start of the sequence.
--Student Input--
def fib(n):
--Instruction--
Give a small next-step hint

Hint:
You can start by adding the base cases for n=0 and n=1.

Generate all hints

- [ ] All Exercises (otherwise from options left)
- [ ] All Models (otherwise from options left)
- [ ] All Instructions (otherwise from options left)
- [ ] Save Results

Filename Results

results.csv

Generate all hints

Enhancing natural-language prompts for code completion tools using subgoals, *Laurian Wannee*

LLM model performance improves when problems are approached **step-by-step**.

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

Goal: develop a subgoal-based prompting approach that programming students can use to interact with LLMs more easily.

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

Composition of a sub-goal label

A programming problem consists of multiple subgoals, each of which consists of three parts:

1. Imperative verb
2. Object
3. Optional: Condition

Example:

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

Without subgoals

```python
# Given a list of integers, return the position
# of the last occurrence of the largest element.
# You may assume the given list has at least one element.
# Do not use the index, find, or max methods.

def lastOccur(arr):

    # Assume the first element is the largest
    largest = arr[0]
    pos = 0

    # Loop through the list
    for i in range(len(arr)):

        # If the current element is larger than the largest
        # element, update the largest element and the posit
        if arr[i] > largest:
            largest = arr[i]
            pos = i

    # Return the position of the largest element
    return pos
```

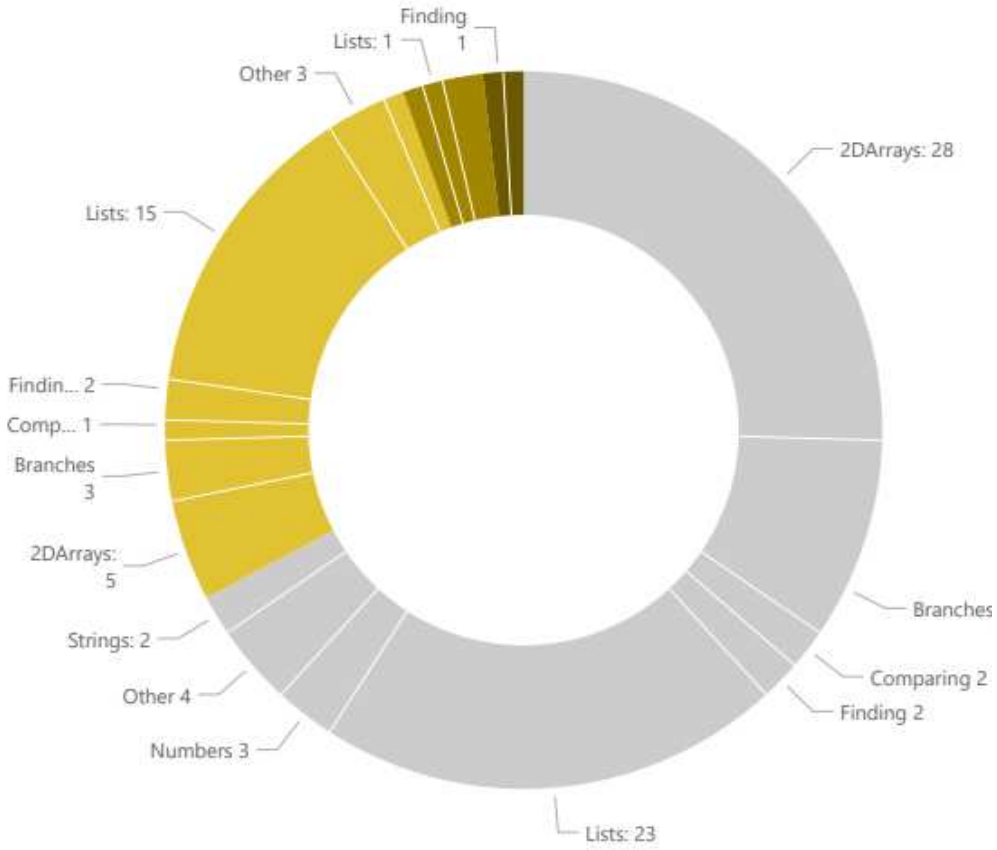
With subgoals

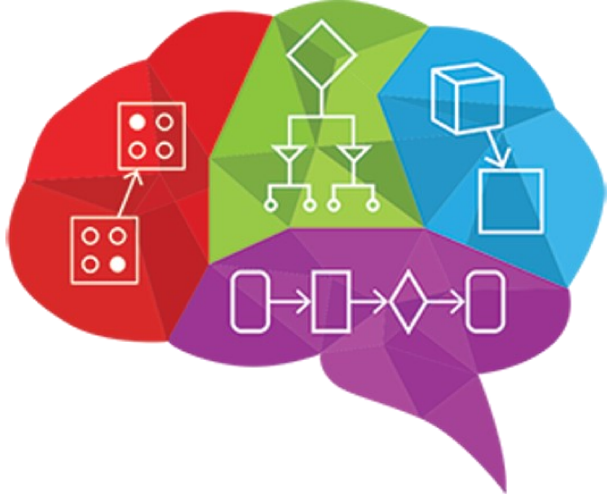
```python
# Given a list of integers, return the position
# of the last occurrence of the largest element.
# You may assume the given list has at least one element.
# Do not use the index, find, or max methods.
# Step 1: loop through the list
# Step 2: remember the element if the current element is larger than or equal to the largest element

def LastOccur(arr):

    largest = arr[0]
    index = 0
    for i in range(len(arr)):
        if arr[i] >= largest:
            largest = arr[i]
            index = i
    return index
```

Preliminary tests

Copilot answers 66% of problems correctly on first sight

Copilot has much more trouble with **lists** than other Python programming concepts.



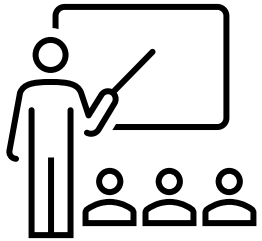


The relation between Computational Thinking skills and the use of ChatGPT, *Roel Groot*

Hypothesis: people with a high score on CT skills will produce a better program when they use ChatGPT.

16 participants first did a standardized CT test, then created a program using ChatGPT.

Preliminary observation: visible differences in how high/low CT performers ask ChatGPT questions.



Implications for CS teaching

Challenges[1]

Ethical Issues: academic misconduct, licensing, energy consumption

Bias and bad habits

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

[1]Programming 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/

More challenges[1]

Over-reliance:

- incorrect solutions
- reduced learning

How can we ensure that our students still learn how to program?

[1]Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. Becker, B. et al. (2023). Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE '23). https://arxiv.org/pdf/2212.01020.pdf

What do students actually still need to learn?

More focus on higher-level program design, code quality, evaluating generated code?

Blogs by Amy Ko:
[Large language models will change programming... a little]
[Large language models will change programming... a lot]

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

To what extent do you think it’s impacting learning CS?

- Not at all
- A little
- A lot

Discussion topics

What topics do we need to research?

How do we need to change our teaching in the context of these tools?

Project at Beta Faculty:
Generative Artificial Intelligence in Programming Education

Hieke Keuning, Ioanna Lykourentzou, Sergey Sosnovsky, Matthieu Brinkhuis, Albert Gatt, Dong Nguyen, Georg Krempl, Christian Köppe

Goal: *explore the implications of generative AI tools, with emphasis on Copilot and ChatGPT, on programming education and to develop a web toolkit with resources and best practices to help teachers understand and successfully embed these tools into their teaching and assessment practices.*

May 2023 - April 2024

Utrecht University

Thank you!

h.w.keuning@uu.nl

www.hkeuning.nl