



Stichting NIOC

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 (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 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 in 2027 en wordt dan georganiseerd door HAN University of Applied Sciences. Zodra daarover meer informatie beschikbaar is, is deze hier te vinden.

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.

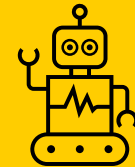
Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.



Universiteit Utrecht

Generative AI in Programming Education

NIOC 27-03-2025

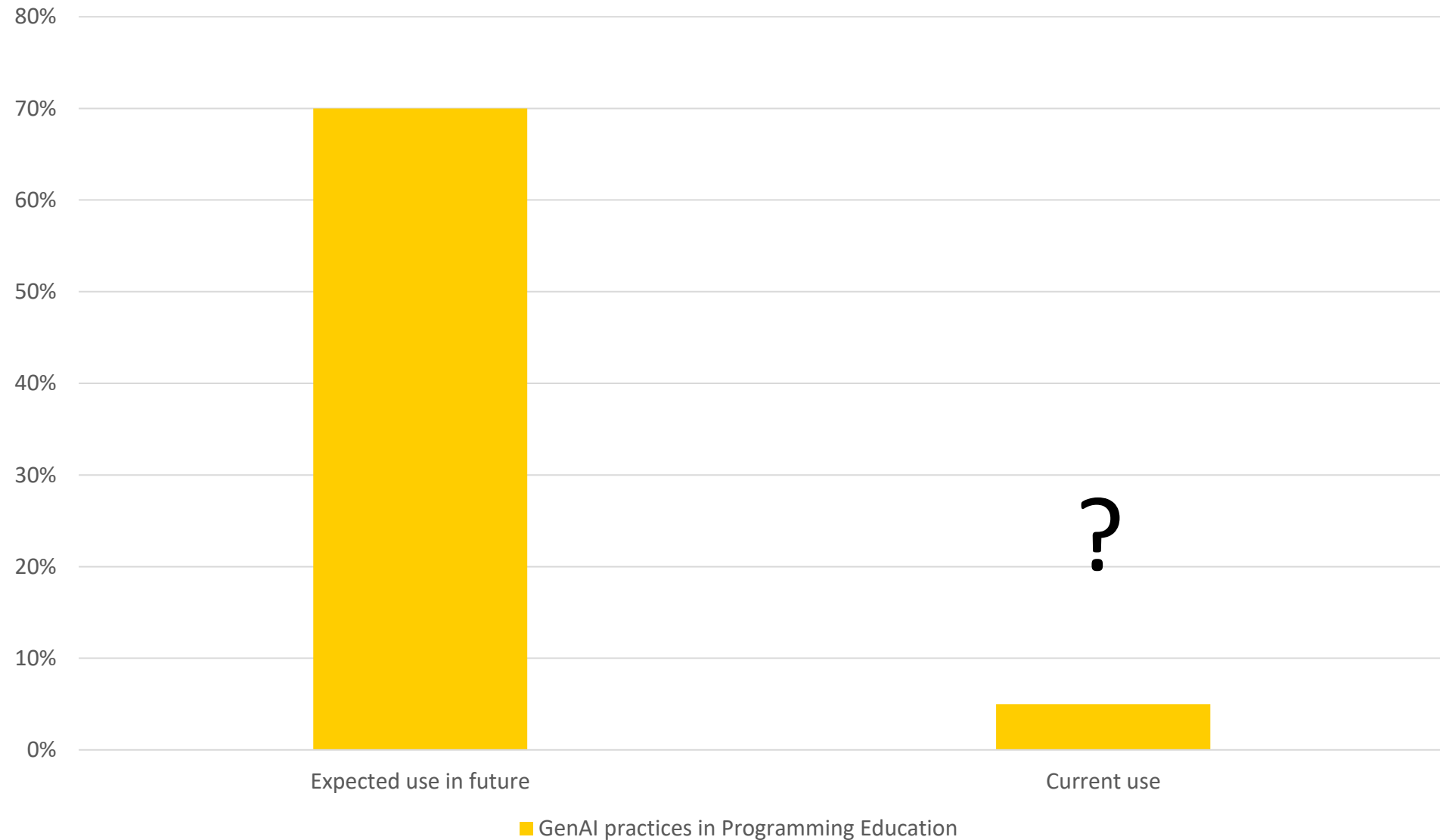


**Christian Köppe, Hieke Keuning, Isaac Alpizar-Chacon,
Ioanna Lykourantzou, Yfke Smit**

Utrecht University



Results from the ICS teacher survey and practices inventory 2023



Teacher practices in Higher Education 2024

75% of educators believe programming competencies have changed.

They believe code reading, and higher-level skills like code testing, problem decomposition, problem understanding, and debugging, have become more important.

30% of educators are integrating it into their classes.

22% of educators are explicitly disallowing GenAI use.

Generative AI for Computing Education

[People](#)[News & Events](#)[Toolkit for teachers](#)[Teacher perceptions on GenAI](#)[Student perceptions on GenAI](#)[Additional resources](#)[People](#)[News & Events](#)[Toolkit for teachers](#)[› Course policies](#)[› **Practices**](#)[› Teaching materials](#)[Teacher perceptions on GenAI](#)[Student perceptions on GenAI](#)[Additional resources](#)

Practices

In a recent survey, which was distributed as part of the current fUSO project Generative AI in Programming Education, 70% of the responding Information and Computing Sciences teachers indicated that they expect to use GenAI tools in teaching practices in the future. A short inventory of teachers' recent GenAI practices revealed only a limited application yet, which indicates the need for collecting and sharing knowledge on specific practices of using GenAI in Programming Education.

Part of the current fUSO project is the collection of such practices for integrating GenAI in Programming Education. These practices were collected from academic and non-academic resources. For the academic resources, a rapid systematic review was performed which resulted in six publications which contained descriptions of practices

Category: Students use AI for Learning Material/Activity Creation (1/2)

Concept
teaching/explanation



Explanation of how
code works

Exemplar solution
generation



GenAI review of
student-written
code

Category: Students use AI for Learning Material/Activity Creation (2/2)

Creation of extra
exercises for students
who want more
practice



Think-Pair-Share
(with AI)

Review of AI-generated
code



Category: Students use AI for solving errors

Error messages
explanation

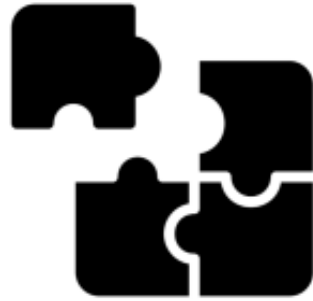


Debugging help



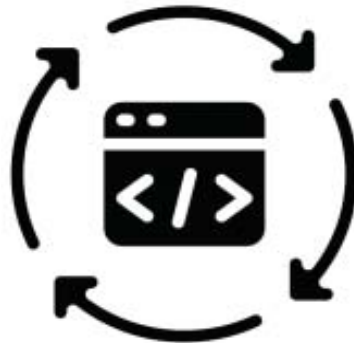
Category: Students use AI for co-creation

Co-code (or co-create)
a program



Specification-to-code
conversation

Code refactoring



Exercise quick start
(Alleviating
programmer's writer's
block)

Learning Outcomes

Two “levels” of learning outcomes

- o LOs for programming (where GenAI can support in achieving them)
- o "new" LOs directly related to GenAI (prompt engineering, deciding when and how GenAI can help, knowing GenAI strategies for better results/learning, e.g. Flipped Interaction Prompts or Persona Prompts, etc.)

Learning Outcomes from Computer Science Teachers Association (K-12)

Students are able to:

1. Evaluate algorithms in terms of their efficiency, correctness, and clarity.
2. Compare and contrast fundamental data structures and their uses.
3. Illustrate the flow of execution of a recursive algorithm.
4. Demonstrate code reuse by creating programming solutions using libraries and APIs.
5. Create clearly named variables that represent different data types and perform operations on their values.
6. Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
7. Create procedures with parameters to organize code and make it easier to reuse.
8. Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
9. Systematically test and refine programs using a range of test cases.
10. Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
11. Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
12. Using correct terminology, describe steps taken and choices made during the iterative process of program development.
13. Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
14. Use flowcharts and/or pseudocode to address complex problems as algorithms.

How to didacticise?

- We know quite a bit (or lot?) about teaching programming
 - > Didactical Approaches!
- Concrete GenAI applications can be combined with known (evidence-based) approaches/theories for learning to program

Approaches/techniques to improve learning programming

- Use-Modify-Create
- PRIMM
- Parsons Problems
- Think-Pair-Share
- (Reducing) Cognitive Load
- Pair programming
- Worked examples with annotations for both product and process (step-wise explanations)
- Live coding
- Semantic Waves
- Block Model
- Unplugged (?)
- ...

One example: Block Model

Relevant for Program
Comprehension

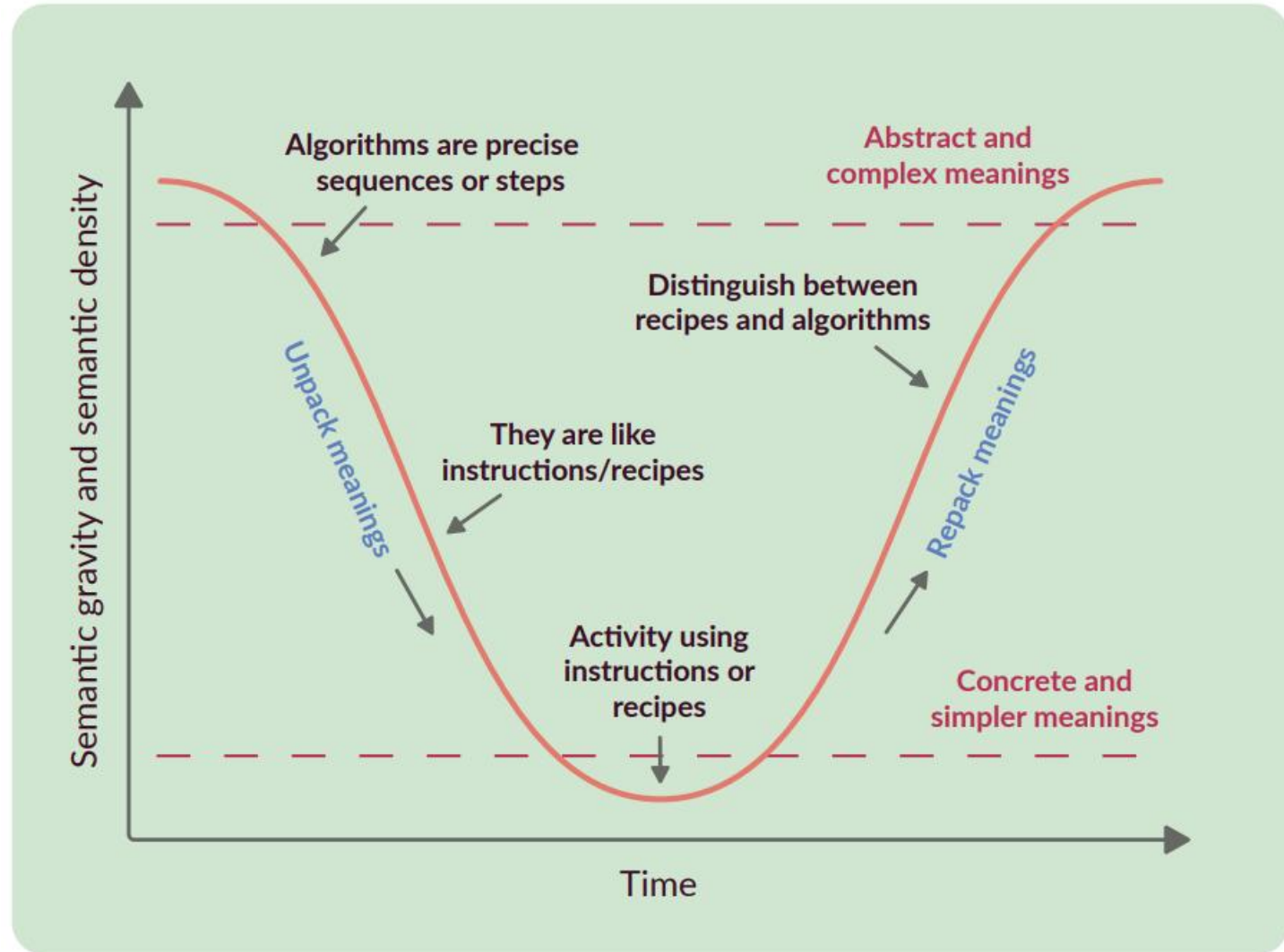
Impact on "Code Explanation"

Macro structure (M)	• Annotate code or draw a diagram to show the overall structure • Restructure an 'untidy' program	• Identify inputs needed to test all program branches • Will line X ever be executed?	• Choose a name for a given program • Select/write a sentence that describes a program's purpose
Relationships (R)	• Identify variable scope • Highlight function calls	• Draw the flow of control • Find redundant conditional branches	• Choose a name for a variable/function • Are two programs/segments functionally equivalent?
Blocks (B)	• Identify block types, such as finite loops, 'else' conditions, function definitions, etc	• Reordering lines of code • Parson's Problems	• Explain the purpose of a block of code
Atoms (A)	• Identify statement types, such as assignments and conditions	• Trace values through a program	• Explain the purpose of a single line
Text surface (T)		Program execution (P)	Function/purpose (F)

Another example: Semantic Wave

Relevant for generalization and
correct terminology

Impact on "Concept
Explanation"



Example LO 1: 10. Read and interpret code segments provided.

GenAI Practices

- Concept explanation
- Exemplar solution generation
- Code Explanation
- GenAI review of student code
- Exercise creation
- Review of AI-generated code
- Error message explanation
- Debugging help
- Co-create program
- Code refactoring
- Specification-to-code conversation
- Exercise Quick Start



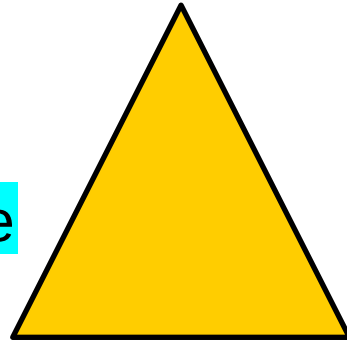
Didactic Approaches

- Use-Modify-Create
- PRIMM
- Parsons Problems
- Think-Pair-Share
- Reducing Cognitive Load
- Pair programming
- Worked examples with annotations for both product and process (step-wise explanations)
- Live coding
- Semantic Waves
- Block Model
- Unplugged (?)

Example LO 1: 10. Read and interpret code segments provided.

GenAI Practices

- Concept explanation
- Exemplar solution generation
- Code Explanation
- GenAI review of student code
- Exercise creation
- Review of AI-generated code
- Error message explanation
- Debugging help
- Co-create program
- Code refactoring
- Specification-to-code conversation
- Exercise Quick Start



Didactic Approaches

- **Use-Modify-Create**
- **PRIMM**
- **Parsons Problems**
- **Think-Pair-Share**
- **Reducing Cognitive Load**
- Pair programming
- **Worked examples with annotations for both product and process (step-wise explanations)**
- Live coding
- **Semantic Waves**
- **Block Model**
- **Unplugged (?)**

Possible solutions

LO 1: 10. *Read and interpret code segments provided.*



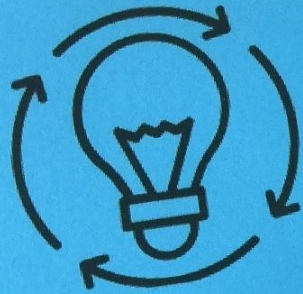
Explanation of how
code works



Think-Pair-Share

Possible solutions

LO 1: 10. *Read and interpret code segments provided.*



Creation of extra
exercises for
students who want
more practice



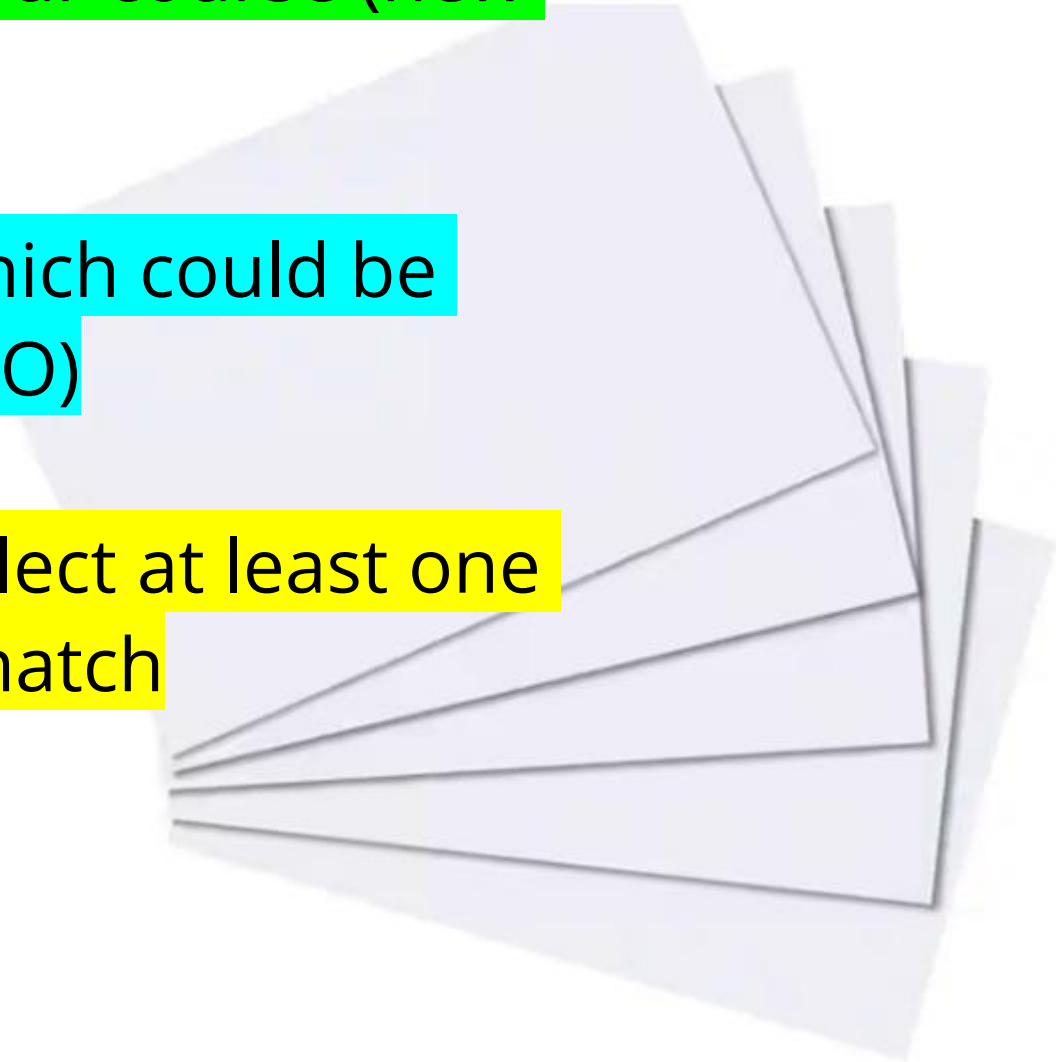
Worked examples

Ways of integrating GenAI in Programming Education

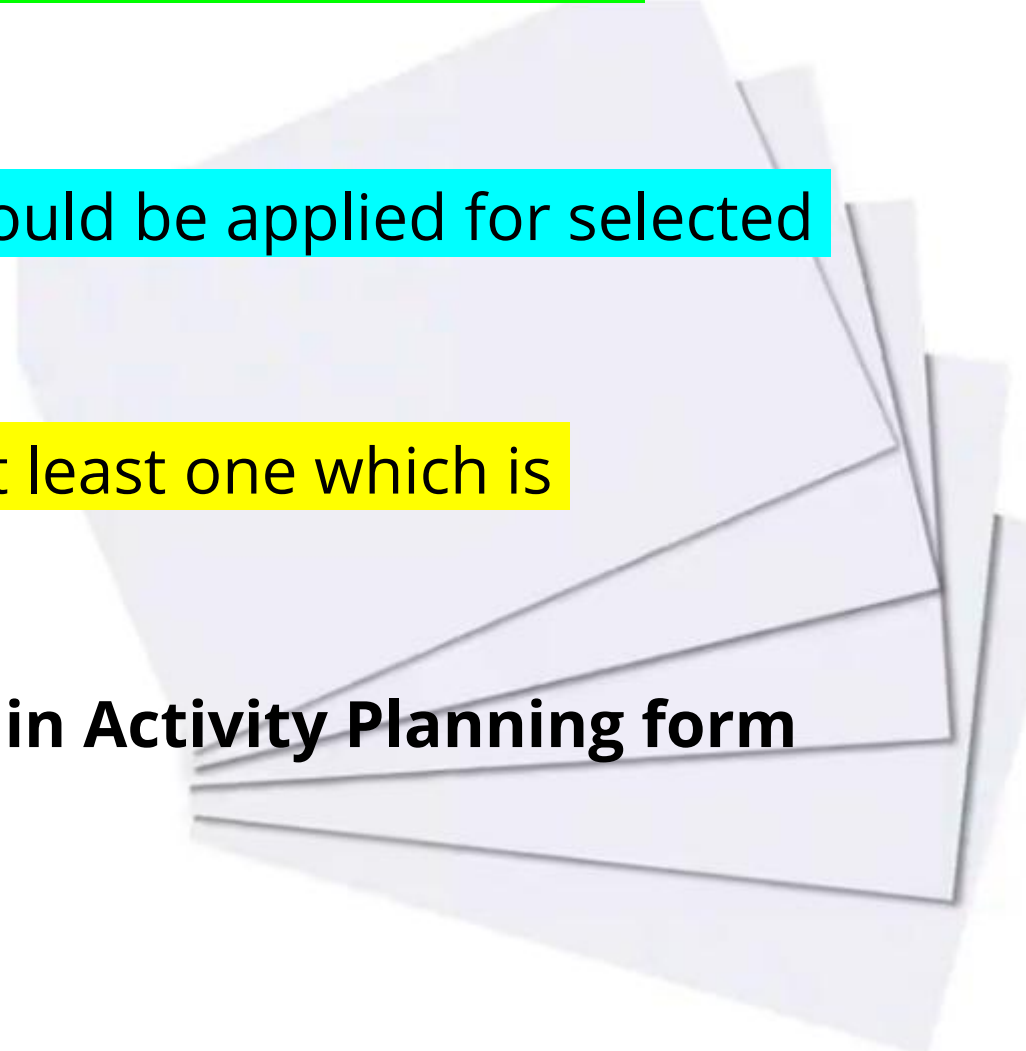
1. Unguided, but with training
2. Guided with specific exercises
3. Combination of guided and unguided

Applications (in pairs) – first round

1. Examine LOs, select 2-3 relevant for your course (new LOs can be added on empty cards)
2. Examine GenAI practices, select 2-3 which could be applied for selected LOs (or only one LO)
3. Examine didactical approaches and select at least one which is applicable for the GenAI-LO match



Applications (in pairs) – second round

1. Examine LOs, select 2-3 relevant for your course (new LOs can be added on empty cards)
 2. Examine GenAI practices, select 2-3 which could be applied for selected LOs (or only one LO)
 3. Examine didactical approaches and select at least one which is applicable for the GenAI-LO match
 4. **Think about concrete application and fill in Activity Planning form**
- 

Feedback (tops and tips)

Please fill in the short survey!

If consented, I'd like to make pictures of your activity plans.

Questions/contact:

Christian Köppe – c.koppe@uu.nl

THANK YOU!



De informatie in deze presentatie is met zorg samengesteld,
maar er kunnen geen rechten ontleend worden aan de inhoud.