Generative AI in programming education:
Bridging the gap from school to what lies ahead

Raspberry Pi Computing Education Research Seminar

Brett A. Becker
brett.becker@ucd.ie
www.brettbecker.com*

*all papers mentioned that aren’t explicitly linked are available here open-access. Slides available
About Me

• K-12 & Undergrad (Physics & CS Dual Degree)
  • in USA

• MSc (Computational Sci), PhD (Parallel Comp), MA (Higher Ed)
  • In Ireland

• Teach
  • In China

• Research
  • In Ireland (or wherever else I am)
  • 6x PhD students
    • all in Computing Education
    • four are educators themselves (primary – university)
    • one focussing on pre-university AI education
About Me

A few recent projects (not about GenAI 😊)

• *Irish National Forum for the Advancement of Teaching and Learning in Higher Education*: Teaching and Learning Research Fellowship (5 given nationally)
  • “Teaching and Learning for the Next Era of Digital Innovation”

• “Computing Crossroads” *Highlighting Career Diversity in Computing* (now a regular column in ACM Inroads magazine) [computingcrossroads.org](http://computingcrossroads.org)

• Just finished term on Steering Committee of the *ACM/IEEE/AAAI CS2023 CS Curriculum* ([csed.acm.org](http://csed.acm.org)) – Chair of the “Society, Ethics, and The Profession” Knowledge Area
What are we talking about?

- AI
- Programming
- Education
  - Moving from school to wherever life takes one after school

But what is coming down the tracks (at full-speed)?
Do developers still need to learn programming languages in the age of AI?

The impact of generative AI and low-code/no-code tools

Top computer scientists say the future of artificial intelligence is similar to that of Star Trek

by Meg Cox, Loughborough University

Research fields contributing to SHELL. Credit: Nature Machine Intelligence (2024). DOI: 10.1038/s...

April 9, 2024 - 6:36 am
A few words of warning

• Whatever we discuss today might be ancient fossils in just a few years
• Don’t be expecting much technical “wisdom”
• Do be expecting a lot of complicated questions (from me)!
• A lot of this is opinion and conjecture
• My goal is to stimulate thinking about how GenAI and programming could affect all school students and how we can use it best to help them wherever they go after school.
(Approximate) Overview

- “Definitions”
- What we know today (that will change by tomorrow)
- What do we need?
  - Primary-level
  - Second-level (and the transition to) the Real World (including Third-level education)
- How not to do it (fun curricular example)
- How we might do it right (tools and resources examples)
- Beyond programming
- Questions (from me, for you, interspersed throughout 😊)
  - Because I don’t have the answers!
“Definitions”

Always a dangerous idea to put in a presentation but as I believe we have a pretty generalist audience today, here goes...

• Artificial Intelligence (AI): an attempt to model aspects of **human thought on computers**

• Generative AI (GenAI) is an application of specific types of AI, which feature models that are trained to **Generate content typical of a human** (e.g. natural language prose, music, images, or computer programs)

* Free On-Line Dictionary of Computer Science (foldoc.org) so don’t blame me for this one.
“Definitions”

I am going to be a little lazy...

• First- or primary-level: up to ~10 years of age
• Second-level: ~11 to ~17 years of age (think U.S. “high school”)
• Third-level: non-compulsory further education / university / U.S. “college”, etc. Typically, students are adults.
• The real-world: What happens after compulsory school is over and one is an adult, regardless of whatever it is they do, *including third-level education*
“Definitions”

Yes, I’m including “university” as part of the real world.

• Adults
• There by choice
• Serious responsibility

• ...and because today I want to talk about the transition from second-level school to... *wherever one goes after that*
What we know
What we know

Generative AI can program.
  Well.

But it’s not all about programming. GenAI is coming... to everything.
  Fast.

By the end of this talk we won’t even be talking about programming.
  It’s unavoidable!
What we know

Here we go...
Back in the summer of 2021! we ran an experiment seeing how well Codex (GPT-3 with additional layer trained on “all” of the Python code in GitHub) could perform against real students on real student assessments in introductory programming (CS1).
Figure 3: Raw score achieved by Codex on CS1 test problems (accumulating penalties applied for incorrect submissions; problems abandoned after 10 failing submissions). Empty caps on some bars indicate potential scores in the absence of trivial errors.

Figure 4: Student scores on invigilated tests (Test 1 and Test 2), with performance of Codex (plotted as red asterisk).
December 2023: 52 pages of literature review (199 references total) student and instructor interviews, surveys, a thorough treatment of ethical considerations, and a modern benchmarking of GenAI.

...and more I’ll talk about later
The robots really were coming ...and they really are here.
Inherent Limitations of AI Fairness

Computing Education in the Era of Generative AI

Service Robot Anthropomorphism

Gaining Benefits from AI and Data Science

Computing Education in the Era of Generative AI

Talking about Large Language Models

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 2023 score on the same questions is represented by the red ‘X’.
ChatGPT scored up to H1 on Leaving Cert computer science exam

A UCD PhD researcher found that ChatGPT breezed through the exam paper.

No More Pencils No More Books: Capabilities of Generative AI on Irish and UK Computer Science School Leaving Examinations

Joyce Mahon
University College Dublin
Joyce.mahon@ucd.ie

Brian Mac Namee
University College Dublin
Brian.macnamee@ucd.ie

Brett A. Becker
University College Dublin
Brett.Becker@ucd.ie

ABSTRACT

We investigate the capabilities of ChatGPT (GPT-4) on second-level (high-school) computer science exam questions. We examine the output of GPT-4, a large language model, and analyze whether the model is capable of answering questions accurately and efficiently. The exam questions are designed to test students' understanding of computer science concepts, including algorithms, data structures, and programming fundamentals. We evaluate the model's performance by comparing its responses to human-generated solutions. The results indicate that GPT-4 is capable of generating high-quality responses, but it also shows limitations, such as the need for human oversight in certain cases. Overall, the findings suggest that the capabilities of generative AI models like GPT-4 continue to evolve and improve, challenging traditional methods of assessment in education.

1 INTRODUCTION

In recent years, artificial intelligence (AI) and natural language processing (NLP) have seen impressive advances, perhaps only surpassed by societal interest in products that advance large language models (LLMs) and what has become known as Generative AI. Only eight months since its public release, ChatGPT has captured the world’s imagination, including predictable speculations about seismic economic changes, the replacement of millions of jobs, and the dubious predictions of “the robots taking over” and humankind’s extinction [4]. Most concern on the numerous fields where Generative AI has been used with some impressive results including computer programming where LLMs have delivered on (not always perfect) AI code generation. In this, natural language prompts serve as inputs to a model, and code is returned. This is potentially revolutionary for computing education, particularly as programming is essential to the study of computing, yet also presents many challenges [1, 19].

GPT-4 is the latest model from OpenAI which like earlier models greatly improves upon its predecessor [6]—although the age of greater performance being achieved by training an existing model has been flagged as ending by AI leaders including the CEO of OpenAI [6]. In this study, we assess the capabilities of ChatGPT (GPT-4) on two second-level school leaving exams: The National Assessment International Examination (IAE) A-Level Computer Science (A-Level CS) exam used in most of the U.K. and several other countries, and the Irish Leaving Certificate Computer Science (LCS) exam.

In this work, our interest goes beyond gauging the performance of ChatGPT particularly as related work indicates that ChatGPT should perform quite well [4]—although how well has not yet been assessed. There are differences between these exams that go beyond content. For instance, the LCS features student choice in the form of optional questions, unlike the A-Level CS. Additionally, “coding” questions require sub-questions (i.e., upon which sub-operation) feature more heavily in the A-Level CS. We are also interested in how the OpenAI knowledge cutoff (as of September 2022) affects the performance of ChatGPT. This may lead to insight on the true “capability transfer” of ChatGPT beyond simple instantiation. We aim to answer the following research questions:

RQ1: How does ChatGPT perform on the A-Level Computer Science and the Leaving Certificate Computer Science exams?

RQ2: What impact do optional and recurring questions have on ChatGPT’s performance?
What are we really dealing with?

Step back from programming for just one second...

Let’s not forget the big picture...
Generative AI can be used to generate original content.

Generative AI can be a great creativity tool.

AI assistants can offer compelling conversational interfaces to information.

Generate some alternative titles for a talk on the impacts of AI on education.

Generative AI can be used to build life changing accessibility tools.
Generative AI is not yet reliable - hallucinations

ChatGPT may produce inaccurate information about people, places, or facts.

Educating the AI workforce is a huge job

AI brings huge legal and ethical questions

Challenges

Generative AI has some industries in turmoil

There are some questionable uses of generative AI

The tally of how many jobs will be “affected by” world-changing technology is different depending on who you ask.
So what do we need?
What do primary students need?

• To be aware enough to:
  • Be safe
  • Learn their way into second-level (which includes a lot of subjects – sometimes including computer science)
  • But it’s not all about CS. Programming is going to faster-than-creep into all? other subjects.
    • It already is.
  • What else?
What can primary students get?

• Safety
• A more *enjoyable* *more engaging* introduction to programming.
• Think Scratch, but easier, more adaptable, more forgiving.
• Perhaps soon a Scratch-like “game” that teaches programming but *knows the student* and can push them just enough at just the right time.
• Perhaps a world where programming really is just like reading and maths. Not just applicable but needed for almost any subject at second-level.
• **What else?**
What do secondary students need?

• This is complicated!
• AI and society and ethics!
• Many different subjects!
• Choice and Choices to make!
• Technical aspects to keep safe in the more complex world that approaches with coming adulthood.
• More (and more complex) problems and solutions.
• Ways to deal with increasing responsibilities and pressures.
What do secondary students need?

• To be aware enough to:
  • Be safe in more complex situations with more powerful tools
  • Learn effectively in all of their subjects
  • Be equipped with the (knowledge, skills, competencies, and dispositions) to be safe and productive wherever they end up:
    • Maybe third-level
      • Any discipline
        • Maybe computer science but statistically unlikely for any randomly chosen student
    • Maybe straight to work
      • Any area
    • Maybe into some other area of society
      • Some might immediately become carers for a relative – what do they need to know?

• What else?
So what do we need to do?
How not to do it

• An historical-ish – yet currently in use – curricular example
• Irish Computer Science Specification for “senior cycle” (final two years of second-level)

• New in 2018 (developed 2016-17)

• An excellent curriculum overall (IMO)

• 59 Learning outcomes

• Caveat: It is a CS curriculum and only for those who choose to study CS
BUT...!!!
ONE! Learning Outcome on AI.

Explain when and what machine learning and AI algorithms might be used in certain contexts
ONE! Learning Outcome on ethics.

Discuss the complex relationship between computing technologies and society including issues of ethics.
In fairness, this was written 2016-17 (eight years ago)! *GenAI wasn’t even on the horizon yet.*

Things are only going to continue to accelerate though...
How not to do it

• a biased resource example
Computer Science
for Leaving Certificate

Brett A. Becker and Keith Quille

Contents

Foreword ................................................................................. 1
About the authors ................................................................. 3
About the contributors ......................................................... 3
How to use this book ............................................................. 3

Chapter 1 Getting started with Python ........................................ 9
Chapter 2 micro:bit .................................................................. 47
Chapter 3 Analytics ................................................................. 73
Chapter 4 HTML and CSS ..................................................... 95
Chapter 5 A brief history of computing ..................................... 118
Chapter 6 Computer systems ................................................. 129
Chapter 7 Computational thinking, algorithms and data representation ......................................................... 144
Chapter 8 Software development ........................................... 169
Chapter 9 More about Python ................................................ 187
Chapter 10 JavaScript ............................................................. 220
Chapter 11 Databases ............................................................. 245
Chapter 12 Modelling ............................................................. 263
Chapter 13 Transforming society: Improving lives, AI and machine learning ......................................................... 277
Chapter 14 Ethics and computing ......................................... 300

Appendix

Afterword ........................................................................... 312
Picture credits ................................................................. 314

https://goldenkey.ie/computer-science-for-leaving-cert/
CHAPTER 13

Artificial intelligence and machine learning

How does machine learning work?
In Chapter 13, we looked at how a computer can be manually programmed to create a model. This works well for small datasets or modeling relatively simple systems.

However, the real value of machine learning becomes apparent in situations when we need to build models from very large data sets with many features. For instance, to model a system that is too complex for an expert to understand and manually program.

Machine learning algorithms
Machine learning uses computers to identify patterns in data and make decisions based on these patterns. In doing so, machine learning algorithms can speed up many of the processes that can be lengthy for humans to undertake.

Although machine learning systems are not programmed in the traditional sense, they still require algorithms in order to "learn," i.e., to solve problems or carry out tasks. There are four broad categories of machine learning algorithms:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

We'll briefly explain each category.

Supervised learning
Supervised learning is the task of learning a function that maps an input to an output based on example input-output combinations, while the data is labeled.

CHAPTER 14

Ethics and computing

Introduction
As we have seen in Chapter 13, many aspects of life have changed radically since the development of computing. The speed and simplicity of digital communications now serve to many industries and services better, cheaper, and more conveniently. It's easy to think that this has made life better for everybody and that computing mostly solves problems.

However, the design and development of computer software applications and systems involves many decisions. Whether instantiating new processes, planning new services, or even making system in a new context, there are many choices to be made in design, development, and deployment.

Decisions often involve balancing different needs, values, and expectations. This can, in turn, create problems. For every new system, questions such as these must be answered:

- Who is it for?
- Who will benefit from it?
- Are any group being further marginalized?
- Does the system work?
- Who is responsible for automatic processes and decisions?

In this chapter, we will look at some of the principles that engineers, philosophers, and social scientists have been asking about computing. We will explore topics such as freedom of information and human rights.

First, let's consider the concept of ethics.

What is ethics?
Ethics describes the principles that govern our behavior and the laws that govern our lives. It is the set of rules and regulations—both formal and not—that guide our behavior. Although sometimes people think of ethics as being only about being good or bad, ethics are more like a personal and contextual belief about what is right and wrong or just and unjust.

Considering outcomes
When trying to decide whether something is right or wrong, we often start by looking at its outcomes or consequences. For instance, we can say that social media is good because it offers many positive opportunities for social contact, public services, entertainment, and so on. Similarly, restricting media is, or making it expensive may have negative consequences for people.
But...

• All students need this information, not just those that choose to study CS (at second- or third-levels).

• All students.

• What do we do about that?
Nearly 80% of British teenagers have used generative AI

Ofcom report on digital habits finds YouTube has overtaken Facebook as UK's most visited website
How not to do AI education at second-level

All second-level students need to be aware of:

• What is AI? What is not AI?
• Where and how will they come across AI, and how to identify it.
• What are the safety (social and technical) risks involved in AI use?
• What are the potential ethical risks involved in AI use?

Then, the CS students can get into technical details... BUT! All students are going to be using Gen AI. What do we do about that from an educational perspective?
But for those that do study CS at third-level

- There are already students at third-level learning to program with GenAI from day 1
- Those who did so at second level could be at a distinct advantage
  - Not only because they know how to use GenAI, but they may be better, more informed, less misguided programmers because they were taught with GenAI.

https://www.manning.com/books/learn-ai-assisted-python-programming
So what can we do at second-level?

• Use GenAI to:
  • Lower barriers to programming (for all students)
  • End up with a more diverse group of students who
    • Aren’t afraid of programming
    • Are engaged in programming

‘LLMs lower the barrier for programming and may help us bring in a broader and more diverse group of students and professionals to the field’ – Leo Porter*

*today.ucsd.edu/story/in-this-era-of-ai-will-everyone-be-a-programmer
How *not* to do it

• a *biased* tool example
• One way may be new tools that leverage GenAI and control GenAI.

• Tools that bridle the raw power (and dangers, including educational ‘pitfalls’) of GenAI and allow GenAI to be controlled and tuned for education.
We have developed:

- A new type of programming problem for education: **Prompt Problems**
- A tool that can be used by students to practice Prompt Problems: **Promptly**
- *Is this a good idea?*
- *What else can be done in this arena?*
• “Prompt Problems” are designed to help students learn how to write effective prompts. It’s more than a copy/paste of the problem itself.

• **Promptly** is a tool that hosts a repository of Prompt Problems and supports the automated evaluation of the prompt-generated code.

• The design (ideally) encourages students to specify and decompose the problem, read the code generated, compare it with the test cases to discern why it is failing (if it is), and then update their prompt accordingly.
Enter your name: Bob
Hello Bob

Write me a Python program that asks the user to enter their name, and then prints the word “Hello” followed by a space, followed by their name.

```python
print("Hello " + input("Enter your name: "))
```

ChatGPT response:

```python
print("Hello " + input("Enter your name: "))
```

Code Running response:

You pass \( (^o^) /! \)
• Students described engaging in metacognitive aspects of learning such as planning their problem solving approach and monitoring whether they understood what they were doing.

• This increased awareness was also exemplified by students who described how the tool might better support reflecting on their learning.

• (We think) Prompt Problems are a useful way to teach programming concepts and encourage metacognitive programming skills.
• Although so-far only tested on first-year students at third-level, I believe this tool would be entirely appropriate for late-stage second-level students.

• Further, a modified tool could be used for even younger students.
Think about this though...
This also changes the way that we (instructors and students) interact.

Is that a good thing?
A typical programming exercise

- **Instructor**
  - **Problem**
  - **Specification**
  - **Implementation**

- **Student**

- **Auto-grader**
  - **Evaluation**
A typical programming exercise

- **Problem**
  - Instructor
  - Student
  - Auto-grader

- **Specification**

- **Implementation**

- **Evaluation**
  - "This is good"
  - "This is ****"
  - "good"
A typical programming exercise

Instructor

Problem

Specification

Student

Implementation

Auto-grader

Evaluation

"This is good" ⟷ "This is ****" "good"

A sentence can be “censored” by having all banned words removed. Define a function called `censor_sentence()` which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with “*”. 
A typical programming exercise

**Problem**

Instructor

**Specification**

Student

**Implementation**

Auto-grader

**Evaluation**

Instructor: "This is good"  Student: "This is ****"  Auto-grader: "good"

A sentence can be "censored" by having all banned words removed. Define a function called `censor_sentence()` which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with "*".

```python
def censor_sentence(sentence, banned_words):
    sentence = sentence.split()
    for word in sentence:
        if word in banned_words:
            sentence[sentence.index(word)] = "*" * len(word)
    return " ".join(sentence)
```
A typical programming exercise

**Problem**

Instructor → Student

**Specification**

*This is good* → *This is *****
*good*

**Implementation**

A sentence can be "censored" by having all banned words removed. Define a function called `censor_sentence()` which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with "*".

```python
def censor_sentence(sentence, banned_words):
    sentence = sentence.split()
    for word in sentence:
        if word in banned_words:
            sentence[sentence.index(word)] = "*" * len(word)
    return " ".join(sentence)
```

**Evaluation**

Auto-grader
A new programming exercise

**Problem**

**Specification**

A sentence can be “censored” by having all banned words removed. Define a function called `censor_sentence()` which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with “*”.

```python
def censor_sentence(sentence, banned_words):
    sentence = sentence.split()
    for word in sentence:
        if word in banned_words:
            sentence[sentence.index(word)] = "*" * len(word)
    return " ".join(sentence)
```

**Implementation**

**Evaluation**
Try solving Prompt Problems with Promptly!

https://promptly-sigcse.web.app/

Warning: This is still a research prototype, and the above example has limited pre-populated problems, and may be buggy!
What else should we do?
How not to do it

- a biased resource example
Students need to be informed about using these tools.
Also includes a “Student Guide” on GenAI including ethical implications, that can be adapted for your students (in any module/class) – and I’d say it’s applicable to second-level
D STUDENT GUIDE

Generative AI refers to a kind of artificial intelligence software that is capable of generating information in response to prompts. The software is trained on source data, and uses that training data as input to a sophisticated model that predicts the appropriate response to the prompt. It does not understand the prompts, but it produces a convincing simulation of understanding. Examples of generative AI systems that use text include ChatGPT and Bard, and generative AI models capable of generating images include MidJourney and DALL-E.

Generative AI tools can be used in ways that increase productivity and help you to learn. However, they may also be used in unsupervised ways that provide answers without helping you to learn.

Policy on generative AI:
- You may use AI tools to help you study during lab exercises and assignments.
- You will never be permitted to use AI tools in secure assessments (i.e., the Test and Exam).

Examples of productive use

Generative AI tools are used in industry so you will be likely to use them regularly in your future work after graduation. Therefore, you should learn to use them appropriately to receive the most long-term benefit. As a student, effective use of generative AI tools are centered on helping you understand course material, and may include asking generative AI to:
- Explain a given topic, or to provide an example of how programming constructs are used.
- Explain your program one line at a time.
- Produce an example that is similar to assignment questions.
- Explain the meaning of error messages.
- Generate code to complete tasks that you have already mastered from previous coursework.

Examples of inappropriate use

Some uses of generative AI do not typically help you learn, and such uses are likely to result in worse long-term outcomes (e.g., you will not be able to complete Test and Exam questions, or to continue following courses that expect a mastery of early programming concepts). Examples of these uses are:
- Using AI tools on official assessments where it has been forbidden.
- Asking generative AI to complete laboratory questions or assignments for you.
- Asking generative AI to debug code that has errors.
- Writing a code solution in a language you know and then asking an AI tool to translate that code into the language required for the assignment.

Risks of generative AI

There are many risks associated with the use of generative AI.
- Accuracy: If you are using generative AI tools for learning then you should always double check the content. For example, if you are assigned to write a program that uses a specific algorithm, AI tools may generate a solution that arrives at the correct answer but does not use the required algorithm. If you use generative AI to assist in the creation of assessed content then you are responsible for the accuracy and correctness of the work that you submit.
- Quality: Content generated by generative AI may alter your understanding and may contain bugs. It is important that you understand how any generated code works and you evaluate the quality of the content.
- Learning: Generative AI can be a powerful productivity tool for users who are already familiar with the topic of the generated content because they can evaluate and revolve the content as appropriate. Tasks assigned by your teachers are designed to help you learn, and relying on AI tools to complete tasks denies you the opportunity to learn, and to receive accurate feedback on your learning.
- Over-reliance: Using AI tools to do your work for you may achieve the short-term goal of assignment completion, but consistent over-reliance on AI tools may prevent you from being prepared for later examinations, subsequent coursework, or future job opportunities.
- Motivation: You may experience lack of motivation for tasks that generative AI can complete. It is important to understand that you need to master simple tasks that generative AI cannot complete. Stay motivated!

Impact on others

There are many consequences to inappropriate usage of AI tools. Some of these consequences may be unintended, and could potentially harm others. For example:

Other students: You could expose other students to harm by presenting their learning or including content in a group assignment that violates academic integrity.
- Faculty: Violating academic integrity standards through the use of AI tools may result in serious and potentially harmful consequences.
- Institutional: Including code from AI tools that you do not understand could expose the university to loss of reputation or even financial harm through lawsuits.

Academic misconduct

Using generative AI in ways that are not permitted will be treated as academic misconduct. This will have serious consequences.

---

Final appendix (last page) of The Robots are Here paper, available at brettbecker.com/publications

Or

iticse23-generative-ai.github.io
But this isn’t just about programming

• In the Real World, *everyone* is going to be using GenAI
  - Students of all disciplines
  - Workers in all fields
  - *All members of society*

How do we deal with this from an educational context?
A context where we will also, thanks to GenAI, have:

• Personalised learning
• Automated Mastery Learning
• Virtual “personal” Teaching Assistants
• Virtual “personal” Teachers?
Over to you to answer my questions 😊