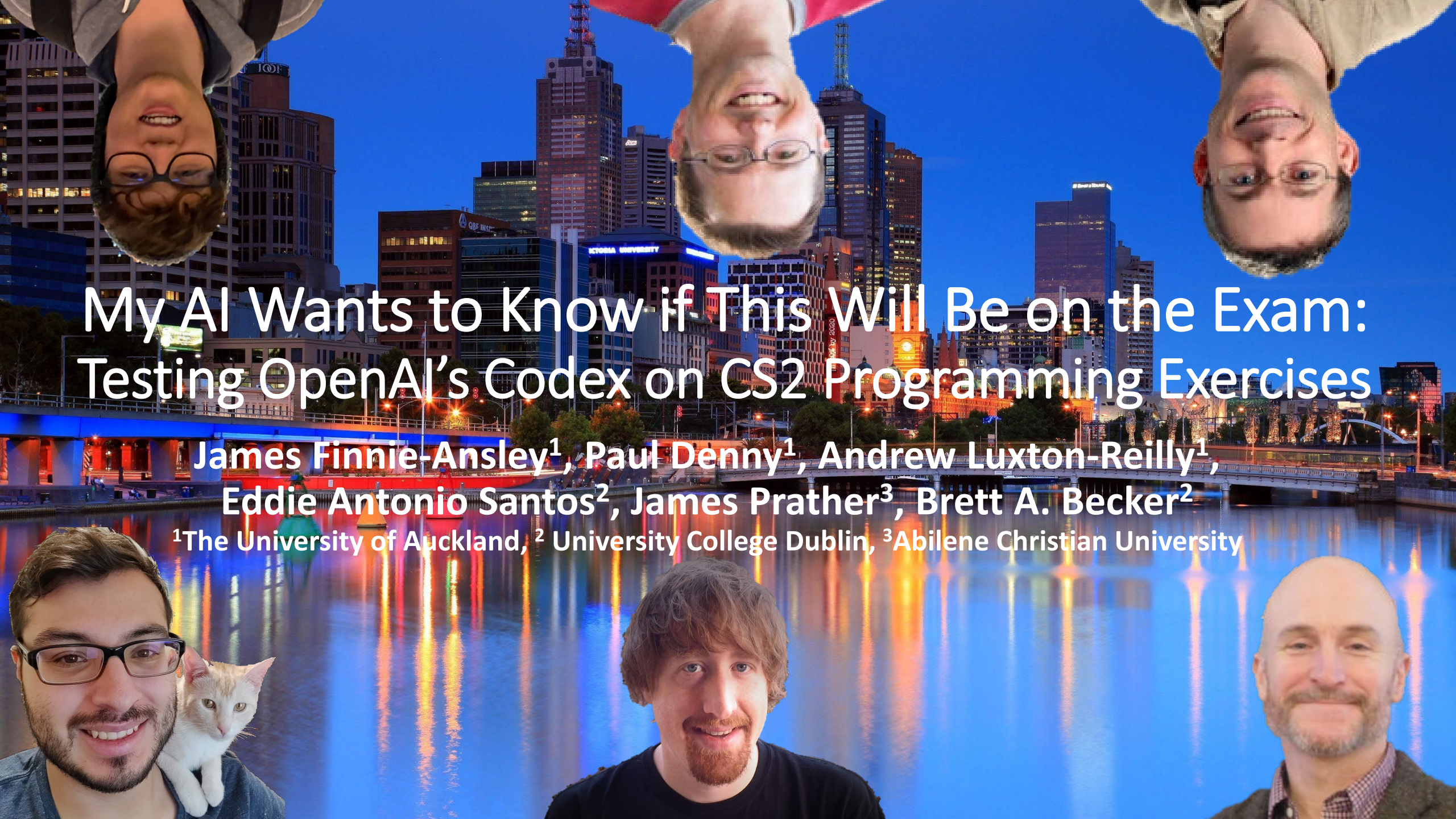
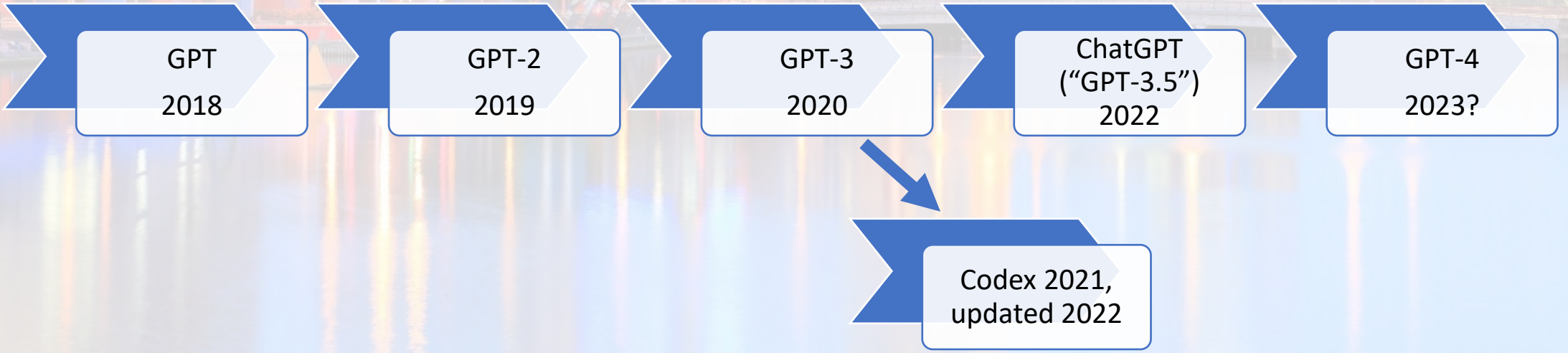


My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises

James Finnie-Ansley[1], Paul Denny[1], Andrew Luxton-Reilly[1],
Eddie Antonio Santos[2], James Prather[3], Brett A. Becker[2]
[1]The University of Auckland, [2] University College Dublin, [3]Abilene Christian University

AI code generation

- Long sought… only in the last 1.5 years realised (publicly, usefully)
- Based on natural language models, e.g. OpenAI GPT-3, with additional training on programming languages -> Codex and ChatGPT.



AI code generation

- GPT-3, ChatGPT, Codex are dynamic in terms of free/paid access
- GitHub Copilot (IDE plugin powered by Codex) made free to students in early summer 2022 and then to teachers a few months later
- Other AI code generation models exist
  - Amazon CodeWhisperer, DeepMind AlphaCode, others
- Together these are proficient in dozens of languages, can translate between programming languages, can explain code in English, can generate code from English (and other languages), provide code (Big-O) complexity, and more.

Nascent work on AI code generation in CS *education*

- Concentrated on the introductory programming course/sequence (CS1)
  - **Finnie Ansley et al. (ACE 2022)**: Codex performs in the top quartile of University of Auckland students on CS1 exams, also decent at Rainfall doi.org/10.1145/3511861.3511863
  - **Leinonen et al. (SIGCSE TS 2023)**: GPT-3 proficient in explaining programming error messages in natural language, often with correct fixes doi.org/10.1145/3545945.3569770
  - **MacNeil et al. (ICER 2022)**: Generating Diverse Code Explanations Using the GPT-3 Large Language Model. doi.org/10.1145/3501709.3544280
  - **Sarsa et al. (ICER 2022)**: GPT-3 proficient in creating programming problems, solutions, test cases doi.org/10.1145/3501385.3543957

Is Codex limited to CS1-level?

- **RQ1**: How does Codex perform on CS2 assessments compared with students?
- **RQ2**: How does Codex perform on CS2 assessments compared with CS1 assessments?
- **RQ3**: What question characteristics appear to influence the performance of Codex?
  - Relevant here because CS2 questions do not only differ in content when compared to CS1 exams, but because the style of questions is often different also

RQ1: Codex vs students in CS2: Method

- 26 programming questions from 2 invigilated (proctored) lab-based CS2 Python tests at the University of Auckland in 2019
  - Questions included problem statement, starter code/function headers, example test case(s)
- This CS2 covers: efficient data organization & manipulation, sorting & searching, writing software that uses & implements common ADTs (e.g. lists, stacks, queues, dictionaries, & binary trees)
- CS1 & CS2 courses use the online Runestone textbooks, cover standard CS1 & CS2 content aligned with the ACM Curriculum
- Compared Codex performance to 264 real students on the same questions
- Automated assessment (CodeRunner) – for students and Codex
- We did not engage in prompt engineering – we simulated students copying and pasting exam questions into Codex

Example question as seen by students (and fed to Codex)

Write a function called `create_string_len_tuple(words)` which takes a list of strings as a parameter and returns a list of tuples. Each tuple contains the **string** and the **length** of the string. Note: you can assume that the parameter list is not empty.

For example:

Test	Result
`my_list = ['A', 'Big', 'Cat']` `print(create_string_len_tuple(my_list))`	`[('A', 1), ('Big', 3), ('Cat', 3)]`
`my_list = ['Free', 'f1', 'f2', 'f3', '']` `print(create_string_len_tuple(my_list))`	`[('Free', 4), ('f1', 2), ('f2', 2), ('f3', 2), ('', 0)]`

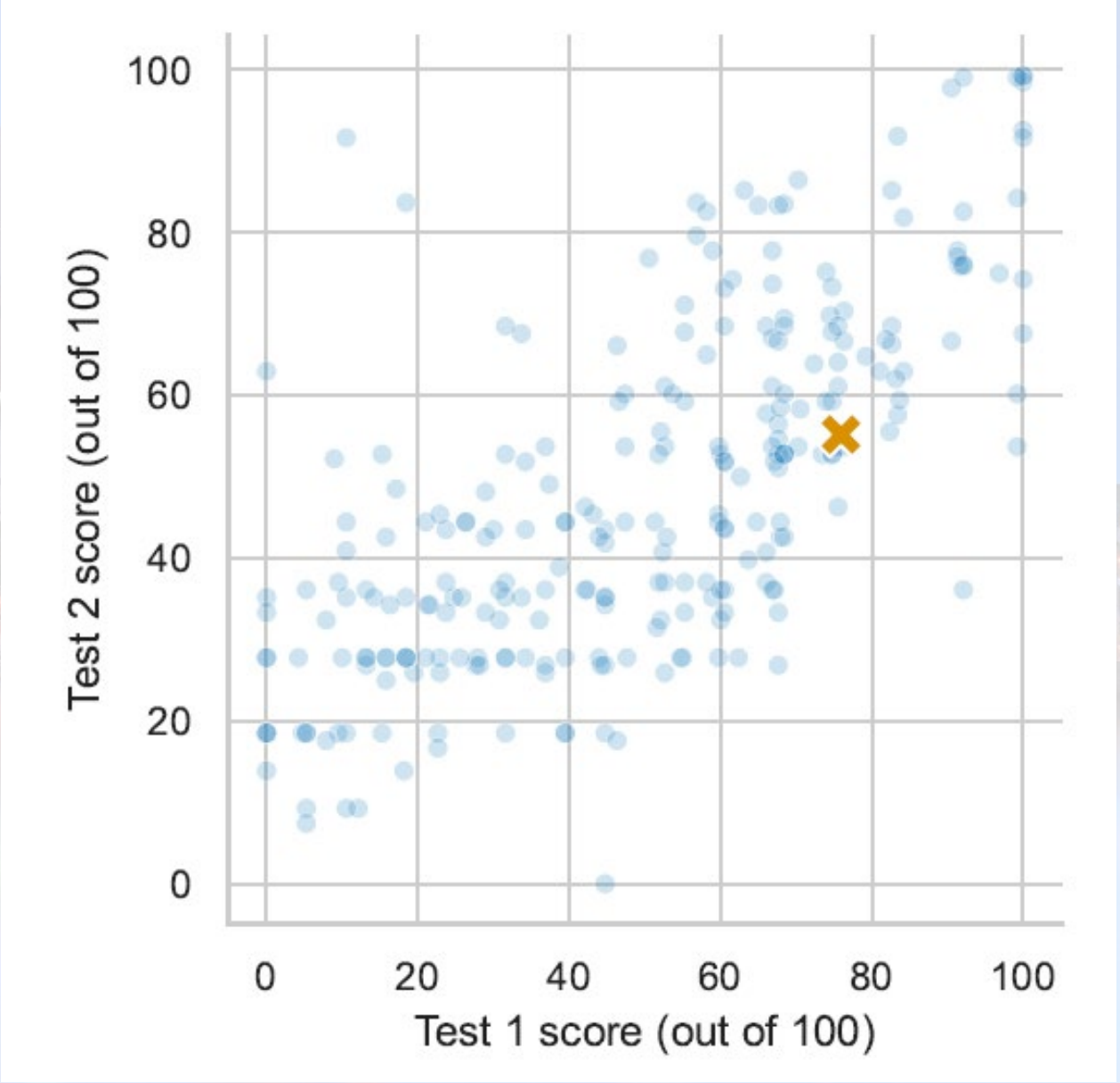
Answer: (penalty regime: 0, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 %)

```
1 def create_string_len_tuple(words):
2
```

RQ1: Codex vs students in CS2: Results

- Codex outperformed students on 19/26 questions
- All questions equally weighted, Codex scored 66% vs average of 48% for students
  - Codex scored >=90% on 12/26 questions
  - Students scored an average of >=90% on 3/26 questions
- Overall Codex ranked 66th place among the 264 students – just in top quartile – *very* similar to the 2022 ACE CS1 study (Finnie-Ansley, et al.)

RQ1: Codex vs students in CS2: Results

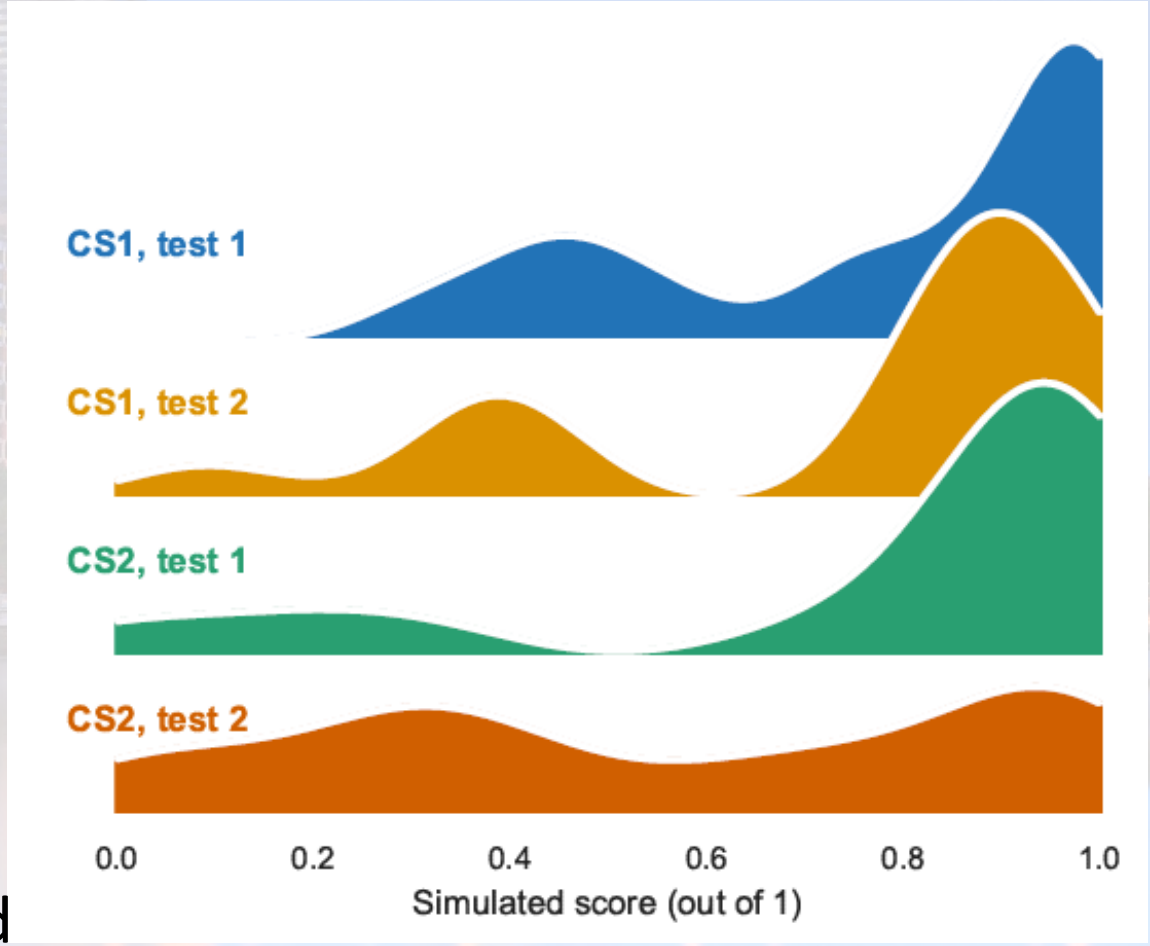


RQ2: Codex performance in CS2 vs CS1: Method

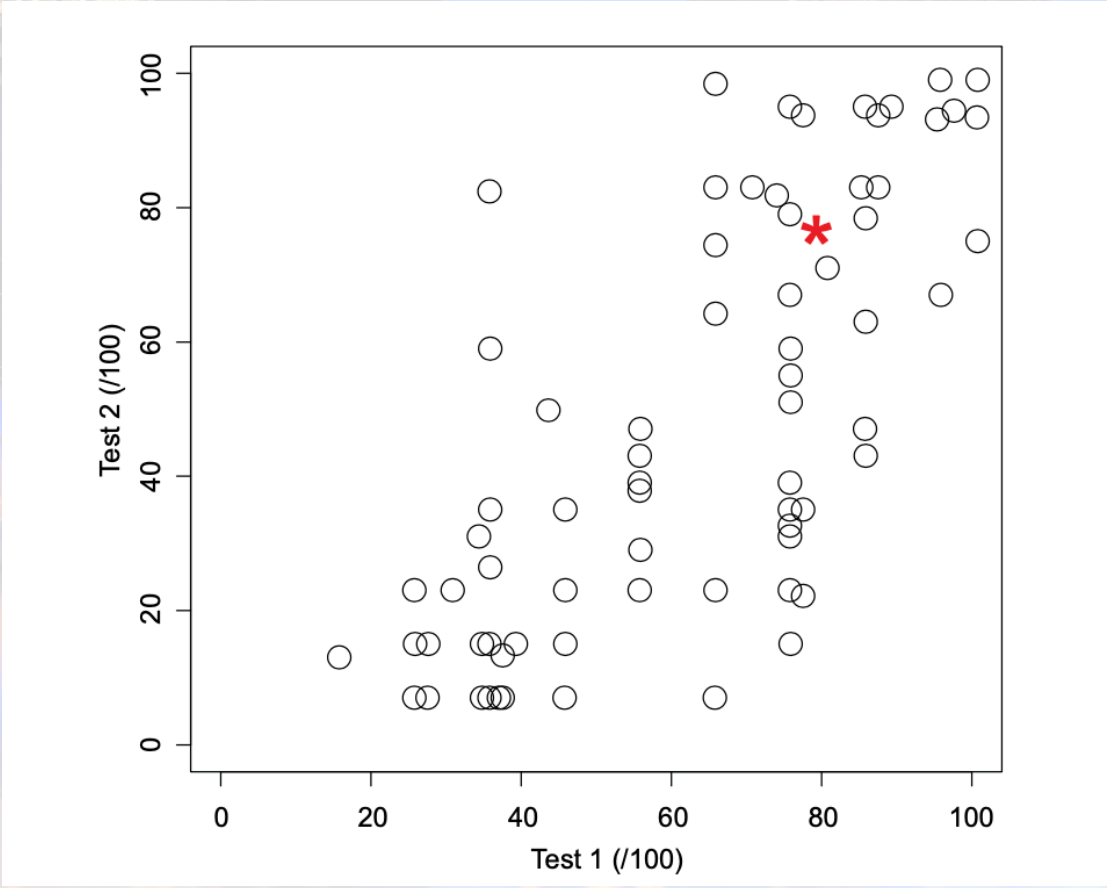
- Compared Codex performance on 2 CS1 exams (ACE 2021) to the Codex results from RQ1

RQ2: Codex performance in CS2 vs CS1: Results

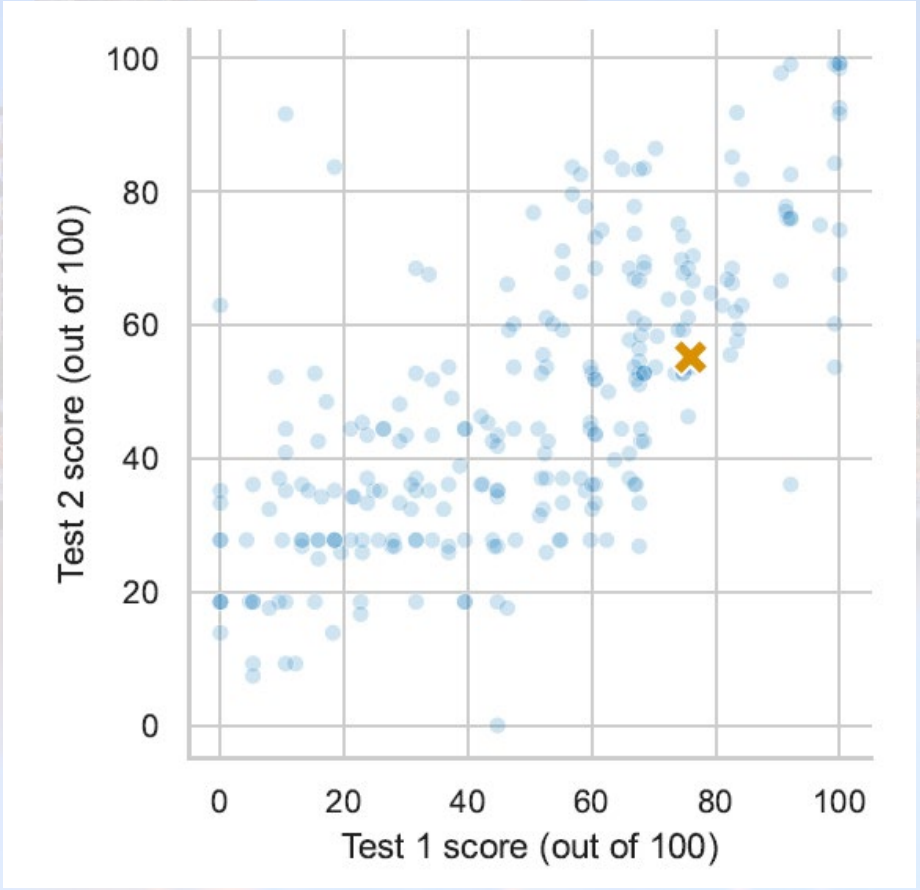
- CS1 tests 1&2 and CS2 test 1 have similar profiles on a kernel density plot (higher peak -> more questions received score in that region)
  - Codex is bimodal, but more hit than miss
- CS2 test 2 is more uniform – Codex exhibits a wider variety of correctness
  - This seems to be due to CS2 test 2 having much longer problem descriptions
- Codex seems to do better with 'blank slate' questions with explicit, well-defined requirements



RQ2: Codex performance in CS2 vs CS1: Results



CS1 (ACE 2022)



CS2

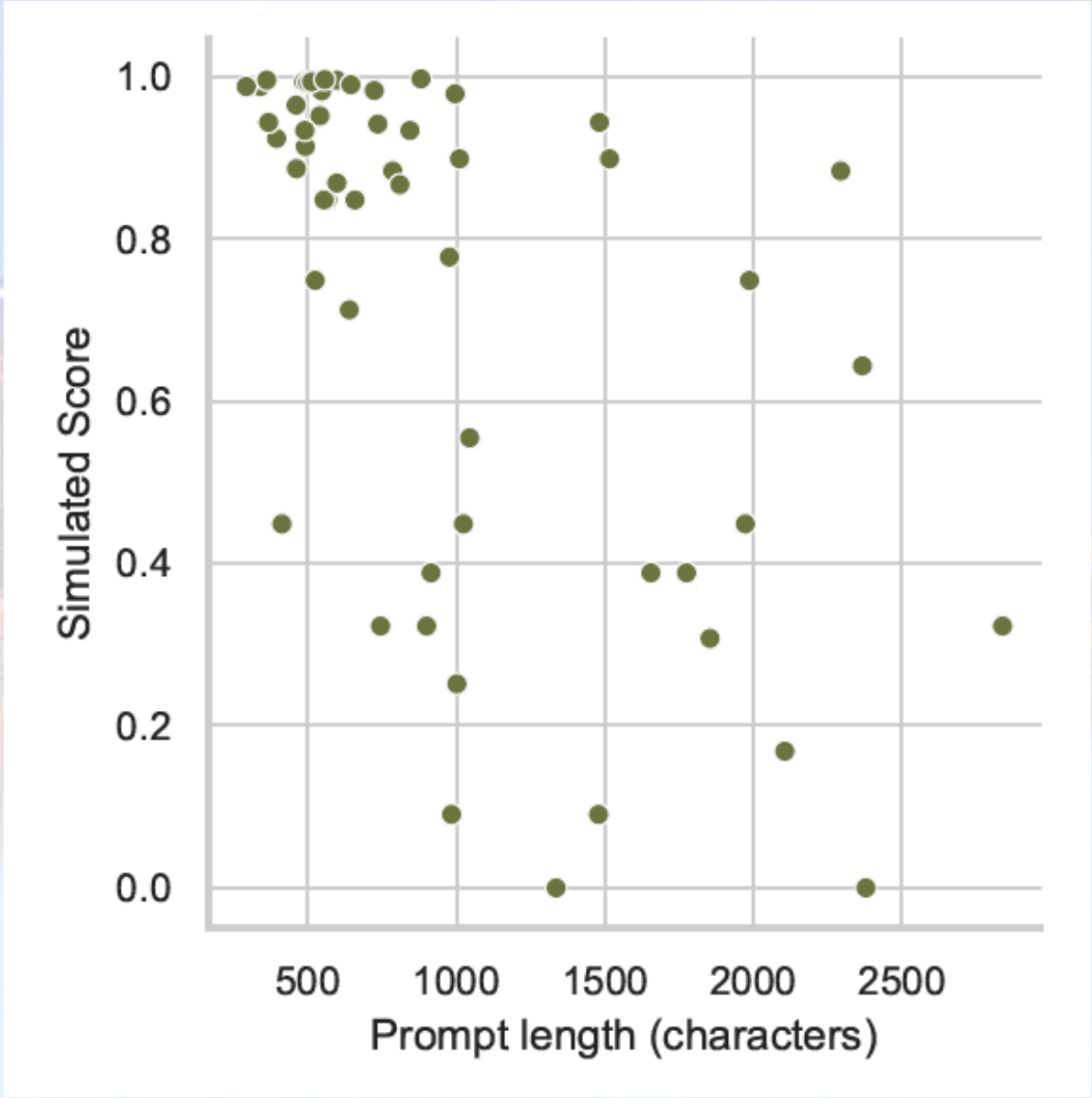
RQ3: What question characteristics appear to influence the performance of Codex? Method

- Divided CS1 & CS2 questions into two groups: Those where Codex scored above the average score **(best performing)**, and those where Codex scored below the average score **(worst performing)**.

RQ3: What question characteristics appear to influence the performance of Codex? Results

- Prompt (question) length
  - **Best performing** questions: mean of 742 characters
  - **Worst performing** questions: mean of 1443 characters
- Possible explanations:
  - More complex questions are longer
  - Codex performance goes down as the number of ‘building blocks’ in questions (complexity) increases – prior work found that this performance degradation could be exponential
  - When questions contain code to be edited or used, performance goes down. 26% of **worst performing** questions exhibited this while only 6% of the **best performing** questions did.

RQ3: What question characteristics appear to influence the performance of Codex? Results



RQ3: What question characteristics appear to influence the performance of Codex? Results

- Possible explanations (continued):
  - **Best performing** questions tend to be posing simple problems that require:
    - the application of standard algorithmic patterns (e.g., filtering, mapping, etc.) or
    - computing common mathematical operations (multiplying numbers, computing prime factorizations, etc.)
  - **Worst performing** questions tend to be:
    - those with implicit edge cases (e.g., not explicitly stating words might contain uppercase letters)
    - those that operate on complex data (nested structures, 2D lists, etc.)
    - Those that need specific output formatting

Speculation!

- The existence of AI code generation tools complicates the delivery of programming education
  - students have ready access to uniquely generated solutions that are frequently correct, but not curated (i.e., could be flawed, or use programming constructs/idioms inconsistent with course instruction)
  - Temptation to use these tools on marked assessment will be high and could have negative impacts
  - Educational effort is perhaps best directed at better supporting students to understand the code to which they are exposed
    - this may emphasize reading over writing, which is consistent with some existing approaches

Speculation!

- We have limited understanding of how these technologies will impact student behavior or how they might impact computing education practices
- Regardless, there will be an impact and we need to understand how to best mitigate the drawbacks and leverage the potential benefits
  - SIGCSE TS 2023 paper: **Programming is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation**
  - In-person and Hybrid via “authors’ corner”
  - Sneak Peek: brettbecker.com/publications (near top)

In conclusion…

- Codex is able to solve most CS2 questions, performing similarly to students in the top quartile of the class
- We find evidence that Codex may perform better on questions that are more precisely defined, succinctly written, have fewer edge cases, and do not require adapting existing code.
- This work confirms that Codex is capable beyond the complexity of CS1 problems. It is unknown at what point the complexity of questions will markedly impact Codex performance
- How educators should adapt to this new technology remains an open question.
- More work is needed in this rapidly emerging area so educators can best adapt their classroom practices in ways that continue to benefit student learning


Thanks! Questions?