The Robots Are Coming:

Exploring the Implications of OpenAI Codex on Introductory Programming

James Finnie-Ansley*, Paul Denny*, Brett A. Becker**,
Andrew Luxton-Reilly*, James Prather***
The University of Auckland*, University College Dublin**, Abilene Christian University***

The Turing Test: Create a Node chain

Which code snippet was written by: a machine, a student, and me?

Student Solution:

A

```
def from_list(data_list):
    if len(data_list) == 1:
        head = Node(data_list[0])
        return head
    elif len(data_list) == 2:
        current = Node(data_list[1])
        head = Node(data_list[0], current)
        return head
    elif len(data_list) != 0:
        current = Node(data_list[1])
        head = Node(data_list[0], current)
        count = 2
        while count != len(data_list):
            next = Node(data_list[count])
            current.set_next(next)
            current = next
            count += 1
        return head
```

My Solution:

C

```
def from_list(data_list):
    head = None
    for value in reversed(data_list):
        head = Node(value, head)
    return head
```

Machine Solution:

B

```
def from_list(data_list):
    if len(data_list) == 0:
        return None
    head = Node(data_list[0])
    current = head
    for i in range(1, len(data_list)):
        current.set_next(Node(data_list[i]))
        current = current.get_next()
    return head
```

OpenAI Codex

- A powerful tool based on the GPT-3 natural language processing model (at the time (2021) the largest NLP Transformer on Earth)

- Can generate code from English text prompts, translate code between languages, explain the function of a given piece of code in English (or, many other natural languages)

- Is used to power tools such as GitHub Copilot which is now becoming readily available (students DO already have this)

What Do We Do About This?

- Do we make peace and embrace these technologies?
- Declare war and prevent students from using it?
- Look the other way and hope for the best?

We Gave Codex a Test… literally

- We tested OpenAIs Codex against:
  - Two Exams given to students in an introductory programming course
  - Seven variations of Soloway's rainfall problem (including one unpublished variant)
- We wanted to find out:
  - How well Codex performed on these problems – particularly compared to students
  - How much variety there was in solutions generated by Codex

An Example – Palindrome

Playground

Load a preset... Save

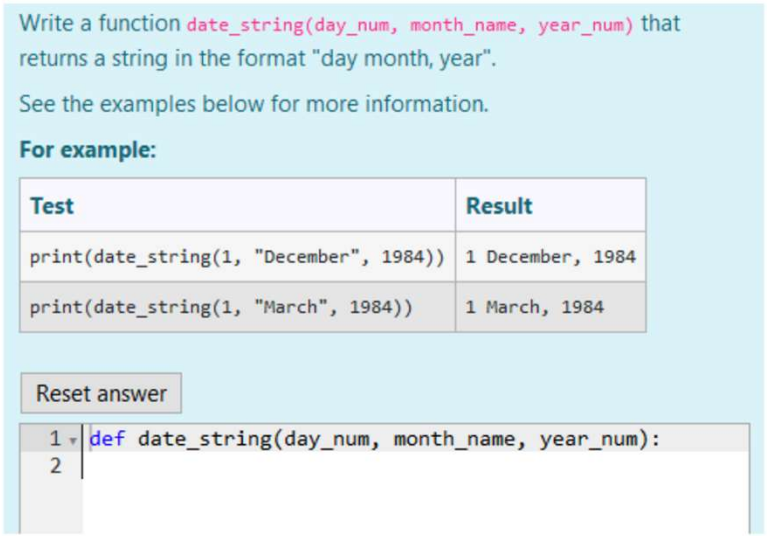
```
"""
Complete the make_palindrome() function that takes a string parameter, a_string. The function processes the string as follows:

All alphabetical characters are made lowercase.
Any characters that are not alphabetical or numerical are removed.
If the processed string is a palindrome, it is returned by the function. A palindrome is a sequence of characters that reads the same, backwards and forwards. For example, the string racecar is a palindrome.

If the processed string is not a palindrome, the function turns it into a palindrome by concatenating to it, the same set of characters in the processed string, but in reverse order. For example, the string damir is not a palindrome. To make it a palindrome we can concatenate to it the string rimad to get damirrimad, which is a palindrome.
"""

def make_palindrome(a_string):
```

Two Invigilated Tests



Write a function `date_string(day_num, month_name, year_num)` that returns a string in the format "day month, year".

See the examples below for more information.

For example:

Test	Result
`print(date_string(1, "December", 1984))`	`1 December, 1984`
`print(date_string(1, "March", 1984))`	`1 March, 1984`

Reset answer

```
1 def date_string(day_num, month_name, year_num):
2
```

What Students Were Provided

```
"""
Write a function date_string(day_num, month_name,
year_num) that returns a string in the format
"day month, year".
>>> print(date_string(1, "December", 1984))
1 December, 1984
>>> print(date_string(1, "March", 1984))
1 March, 1984
"""
```

What Codex Was Provided

Types of Test Questions – Easy

Write a program that prompts the user to enter an integer value x.
Your program then displays the value of 4x.

Write a function named count_odd(my_list) that returns the number of odd integers in a given list.

Write a function date_string(day_num, month_name, year_num) that returns a string in the format "day month, year".
print(date_string(1, "December", 1984))
should print
1 December, 1984

Complete the sort_tuple(a_tuple) function that takes a tuple object as parameter. The function returns a tuple with the same contents ordered in ascending order (smallest to largest/A to Z). You can assume that the tuple will only contain one type of object.

Types of Test Questions – Medium

Write a program that keeps asking the user to enter integers until a negative integer is given. Your program will then output a message that indicates the sum of all the numbers entered but not including the negative number.

Write the check_the_words(words_list) function which takes a list of words as a parameter. The function returns True if all the words in the parameter list have fewer than 6 letters and start with a vowel. The function returns False in all other cases.
Vowels are the letters:
vowels = "aeiouAEIOU"
Note: you can assume that all the words in the parameter list contain at least one letter.

Write the get_rearranged_list() function which takes three parameters: a list of numbers, an index value and an integer (how_many). The function returns a new list which has exactly the same elements as the parameter list but with the elements rearranged so that the slice of the list starting from the index parameter is moved to the end of the new list. The length of the slice is given by the how_many parameter, e.g. if the parameter list is [15, 10, 5, 20], the index value is 1 and how_many is 2, then the two elements starting from index 1 (the elements 10 and 5) are moved to the end of the new list.
Note: you can assume that the index parameter is always a valid index and that there are always enough elements in the list to allow the rearrangement without error.

Types of Test Questions – Hard

Complete the get_max_keys(data_dict) function that takes a dictionary object, data_dict, as a parameter. The data_dict parameter contains string keys and integer values. The get_max_keys() function returns a tuple of keys that have the largest integer value. The keys in this tuple should be sorted in ascending order (smallest to largest/A-Z). Note that if data_dict is empty, the get_max_keys() function returns an empty tuple.

Complete the function get_words(filename) that takes a string filename as a parameter. The file is formatted as follows:

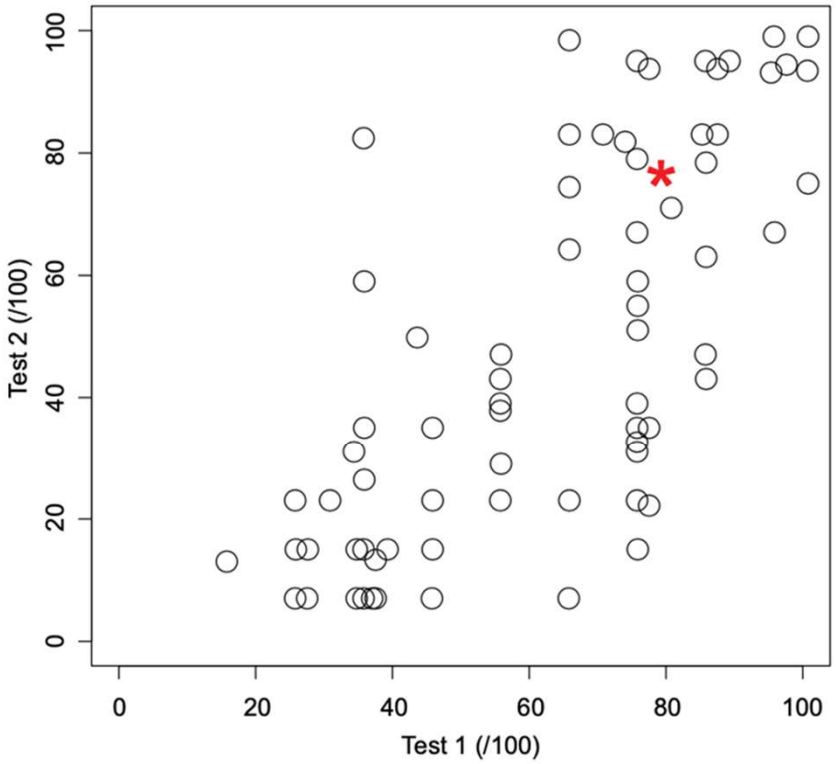
The first line is a single letter.
The second line is a numerical value

From the third line onward we have a list of words, one word per line, separated by a newline character.
The get_words() function will identify all of the words in this list that begin with the letter specified in the first line of the file, and are of the length specified in the second line of the file, and add them to a list. The function returns a tuple consisting of the target letter, the target length and the list of words.

Two Invigilated Tests – Results

- Codex Got:
  - Test 1: 78.5%
  - Test 2: 78%

- Ranked 17th out of 71 students



Student scores on invigilated tests (Test 1 and Test 2), with performance of Codex (plotted as red asterisk).

There were some things Codex struggled with

- Formatting: leaving out full stops in output

- Restrictions: problems preventing the use of certain functions such as .split()

- Questions with specific formatting requirements, e.g.

```
>>> print_triangle_numbers(5)

    1
   222
  33333
 4444444
555555555
```

```
>>> data_dict = {"A":5,"B":4,"C":2}
>>> draw_bar_graph(data_dict,5)
 ____
|    | ____
|    ||    |
|    ||    | ____
|    ||    ||    |
|    ||    ||    |
-----------------
```

The rainfall problem

- Seven variations of the rainfall problem[1]
  - One of which is unpublished
  - The unpublished variant uses tonnes of apples harvested as the contextual setting.

- We generated 50 solutions for each input and tested them against some basic test cases.
- We then evaluated the general approach taken in each solution

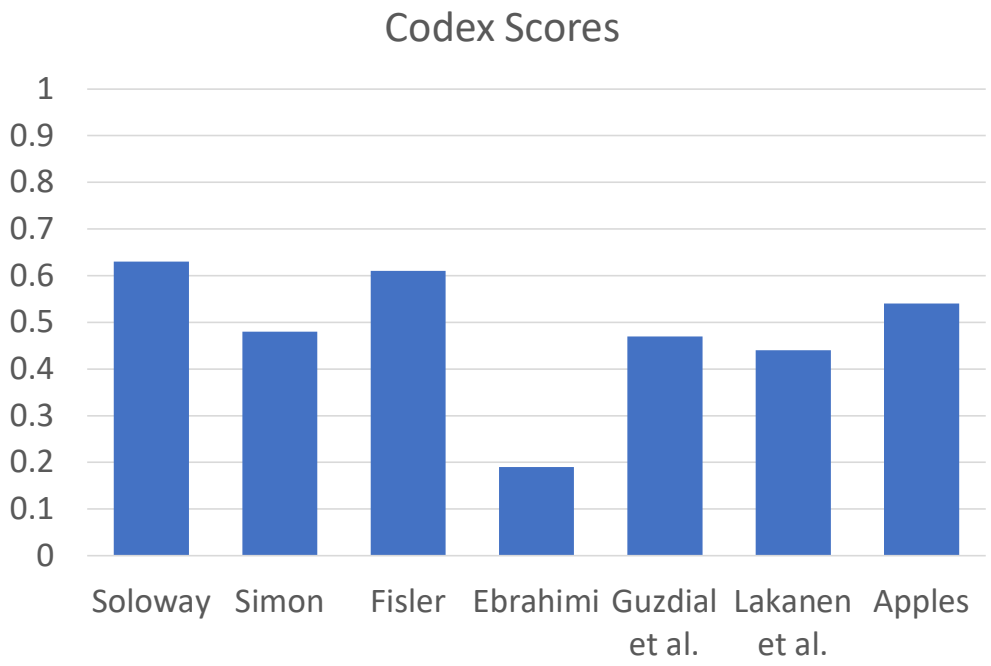
[1]For background on the rainfall problem, see https://web.cs.wpi.edu/~kfisler/Pubs/icer14-rainfall/icer14.pdf

The rainfall problem – results

- Not amazing but not too bad
- All but one (Lakanen et al.) got at least one solution that passed all tests

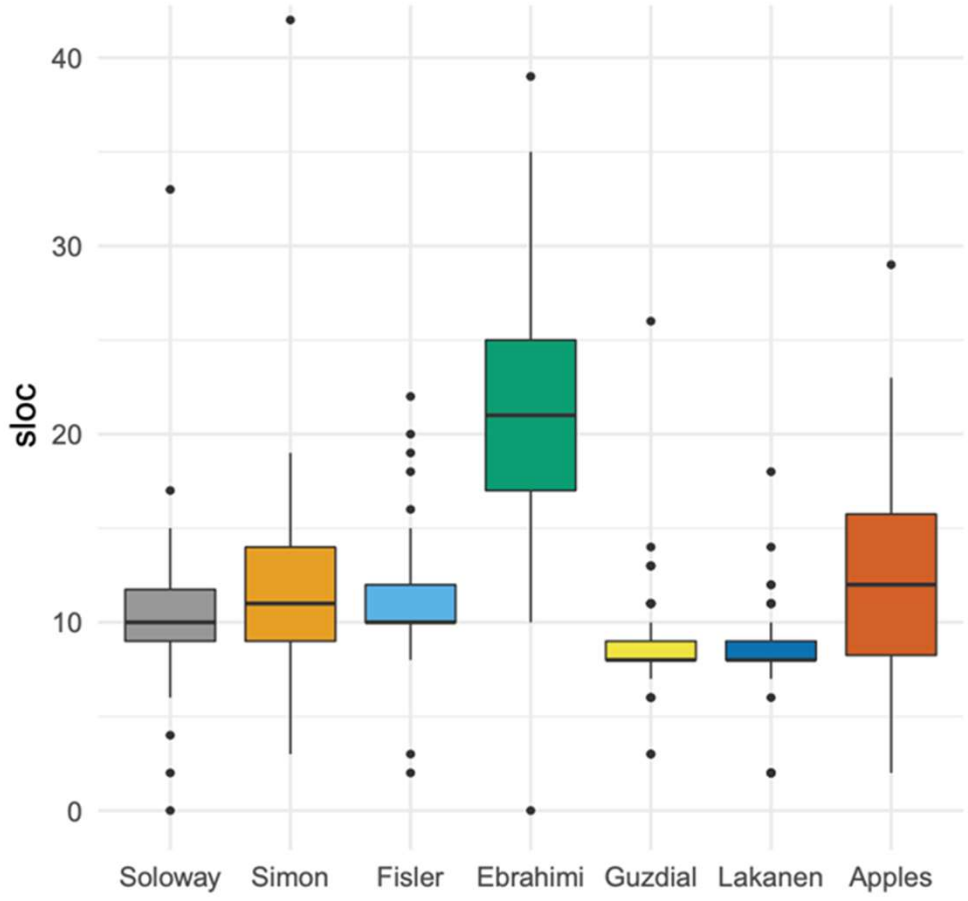
The apples variant handled cases where no valid input was provided (e.g. blank lists, all negative values etc)

- Other variants did not explicitly state what to do in this case



Variation of Solutions

- Source lines of code indicates solutions are at least different.
- Looking at the method used indicates some variety in approach:
  - Most solutions used a single `for` or `while` loop
  - Some used built-in functions or multiple loops



Source lines of code (sloc) per variant.

Summary

- We looked at how codex performed on CS1-appropriate programming problems that were also given to students and found it was pretty good.
  - Not a superstar A+ student but it is up there
    - top quartile on the two tests
    - Much better than the average student (in our class)
- Codex handles variation in the wording of problem prompts pretty well and some boundary conditions (blank lists, etc.)
- Codex is not just spitting out the same code with a different wig on – the approaches it takes to problems are varied, and often sound, sometimes elegant.

What does it all mean?!

Cats outta the bag

Genie is out of the bottle

Codex is out there

Beans? Consider them spilled

Elvis has LEFT the BUILDING

What does it all mean?!

- Students are already using Codex and will continue to use it whether we like it or not
- We need to start acting
  - If we want to integrate this technology:
    - How?
    - How can Codex be leveraged to improve learning?
    - What will industry think of us teaching with this technology?
  - If we want to avoid this technology:
    - How? (again)
    - Can Codex solve other problem types: Parsons, MCQs, EiPE (explain in plain English), fact-based, ill-structured?
    - Can we detect plagiarism if students use Codex?
- All things to consider but all things yet to be fully understood.

Credits

- Variant of this was presented at the 2022 Australian Computing Education Conference
  - Received best research paper award
- Original slides prepared by James-Finney Ansley