What is Artificial intelligence (and when will it take my job)?

Seminar 1

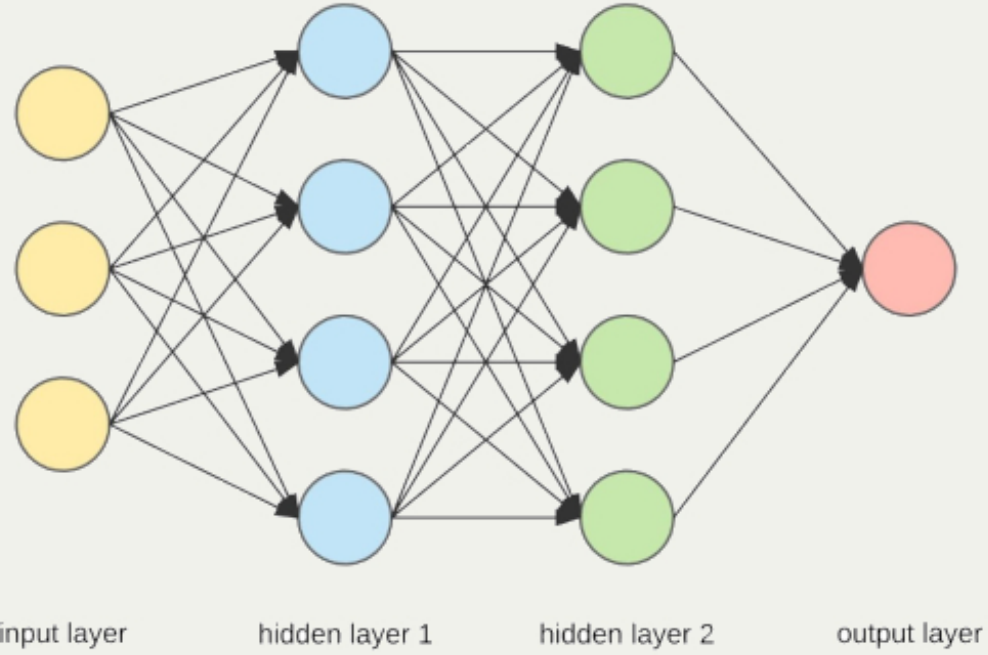
Ciarán O'Kelly

c.okelly@qub.ac.uk

8 June 2022

About me

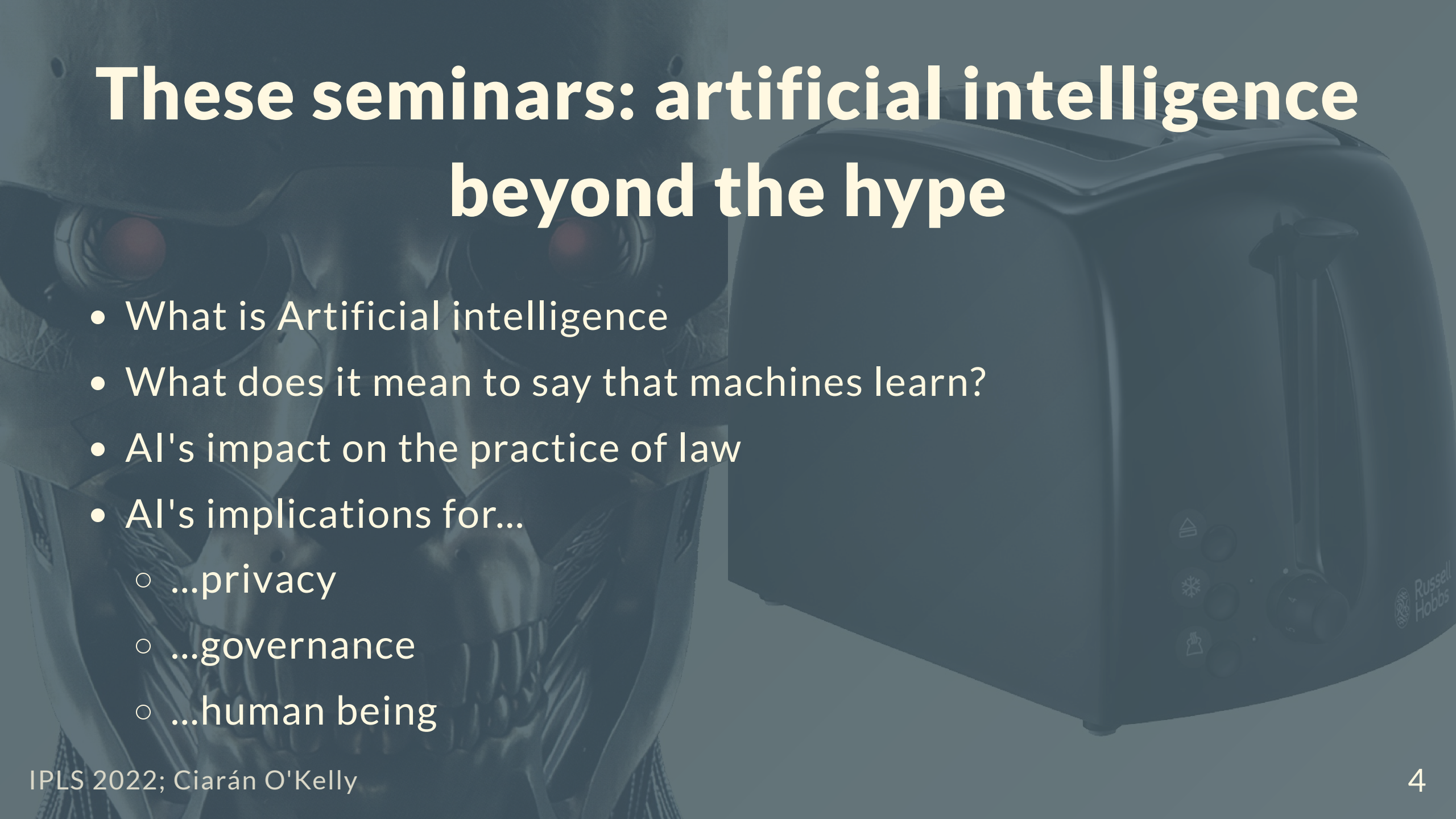
- Senior lecturer, QUB School of Law
- Coordinator LLM Law and Technology
- Research interests in accountability and moral reasoning in governance
- Interested in computational methods in law
  - Natural language processing
  - Corpus linguistics



Artificial Intelligence and its impacts on:

- how we take exams
- how we interact
- how we wage war
- how we practice law
- how we accumulate; label and protect ***data***

These seminars: artificial intelligence beyond the hype

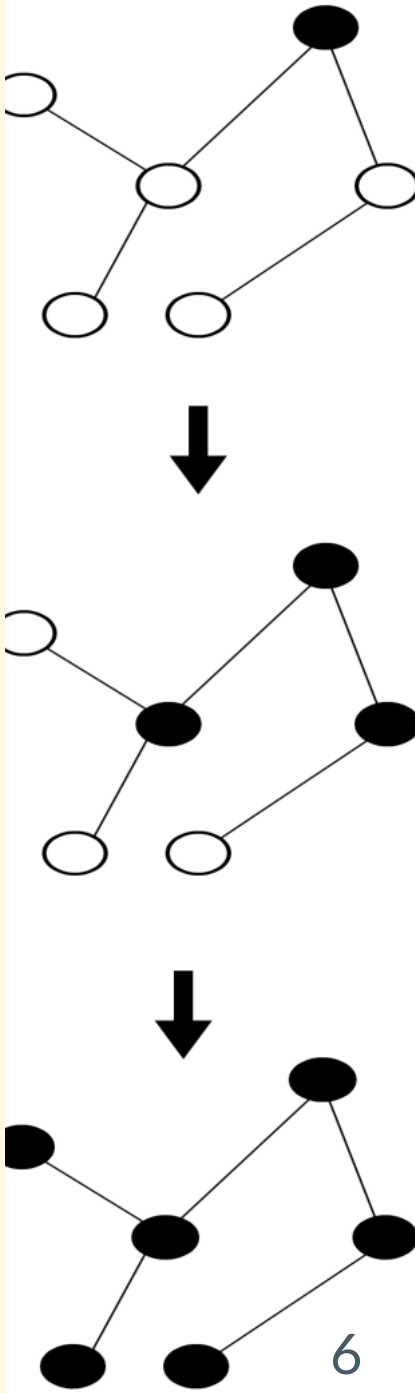


- What is Artificial intelligence
- What does it mean to say that machines learn?
- AI's impact on the practice of law
- AI's implications for...
  - ...privacy
  - ...governance
  - ...human being

The future of (your) work

- Which jobs will go?
- Character of work
- Value of work

1. **Thinking about coding and computational reasoning**
    1. Using Python and 'computational notebooks'
    2. How do/can we ask questions of computers?
2. **AI and its workflows**
    1. Input (data)
    2. Process (algorithm)
    3. Output (impact)
3. **AI and Law**
    1. Summers and Winters
    2. Privacy etc
    2. AI and legal practice



In short:

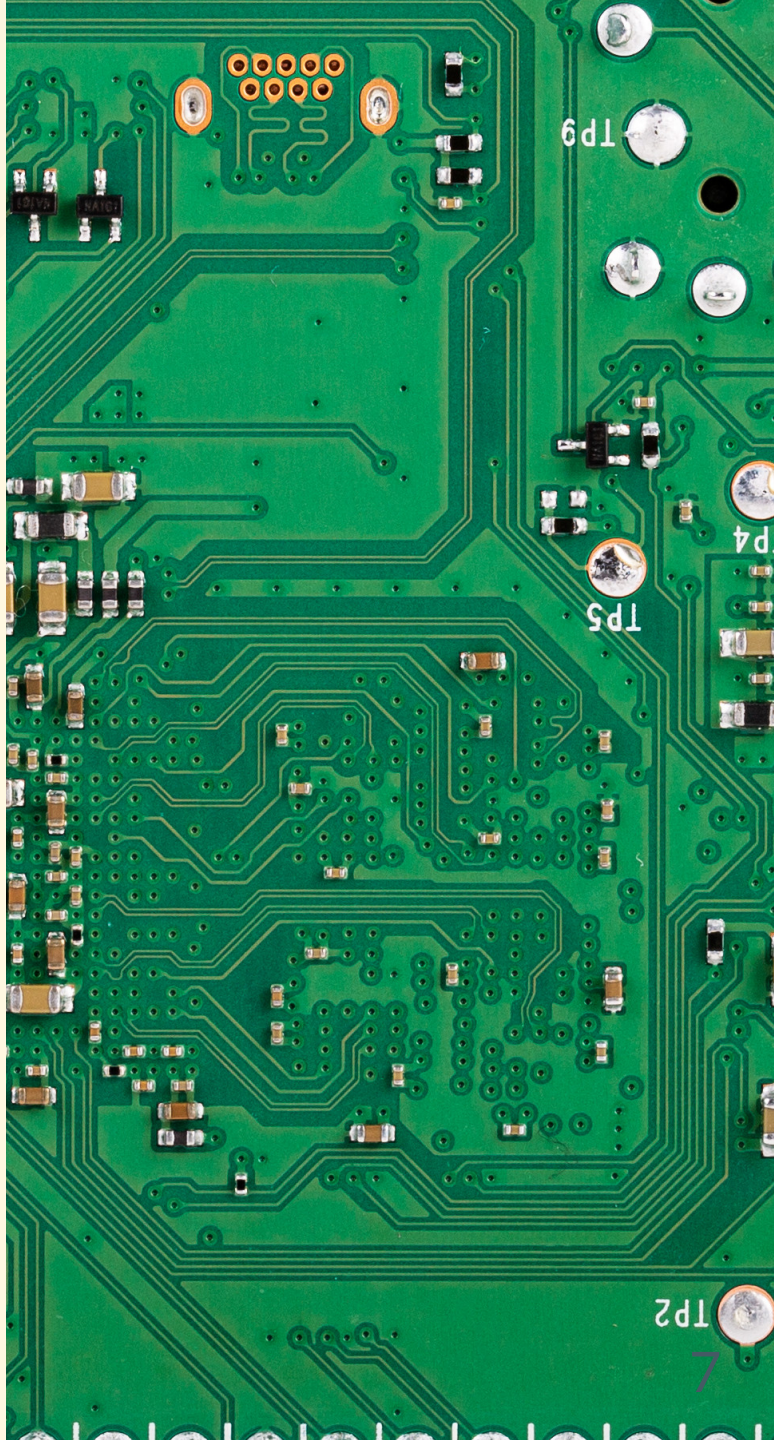
Do some coding and data analysis

↓

Think about 'computational reasoning'

↓

Return to AI's implications and impacts



Introducing Python

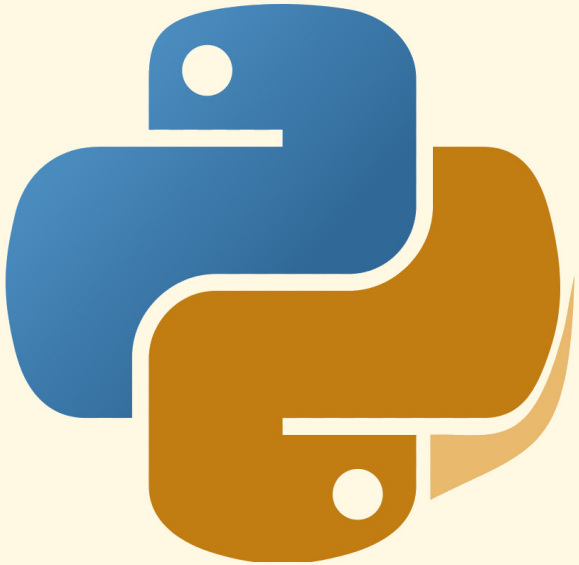
What is Python?

Python is a 'high-level' 'object-oriented' computer language that was first released in 1991.

Object-oriented means that everything in the language is a logical 'object.'

Objects can be created, stored and manipulated.

If we write `pi = 3.14159` in Python then we have created an object with the name `pi` and the value `3.14159` (which of course is not *quite* pi).



Python 2 and Python 3

There are two types of Python. They are not mutually compatible in every respect.

Python 2 is deprecated as of January 2020 and is no longer actively developed. It is still active because many applications were written in it.

Python 3 was released in 2008.

In this module we will use Python version 3.7.6.

Why python?

- Python works at a 'higher' level than a number of other object-oriented languages, for instance `c++`, which is the a core language in many applications (including the apps you are using today).
- Python figures out what kinds of objects you are creating and working out other parts of your operations without you explicitly stating your intentions.
- Python is a slower historically than lower-level programmes because it does more work in the background.
    - The payoff is that it is easier to read than other programmes because much of the inner workings are hidden.

IDEs (and Jupyter): the apps where programming work is done

- *Integrated Development Environments* are programmes that allow you to write, document and run code in the same place.
- Sometimes an IDE runs different panes performing different tasks in the same place (Microsoft's Visual Studio Code is one example of this).
- Others, such as Jupyter notebooks intersperse code, graphics and formatted text.

Noteable

In this module we will use Jupyter notebooks through the University of Edinburgh's Noteable service. Noteable is a container for running notebooks and will be available to you for the year.

Unfortunately QUB Canvas will not allow access to Noteable but you can access it by enrolling here:
https://canvas.instructure.com/enroll/XXXX

Markdown

Jupyter notebooks intersperse code and data plots with Markdown, a highly simplified syntax for marking up text:

- **bold**
- *italics*
- [a link](https:qub.ac.uk)
- etc.

So you write your text and leave all but the simplest formatting to your computer.

Basic Python Concepts

Python is a relatively straightforward computing language but the best way to learn is to use it for solving problems. Nonetheless here are some basic elements of Python.

Syntax

Every computer language has its own syntax that people must learn. Different object types are flagged using syntax (see below) and we operate on specific objects using syntax.

Why Python: an example

Let's write a small piece of code that will take a sequence of radii for circles (1cm - 1,000,000cm) and return the requisite circumference for each radius and export to a file called 'circumferences_<programming_language>.txt'

Recall that the formula to calculate the circumference of a circle is: $2\pi R$

In the next slides I outline script in C++ then Python that:

1. calculate the circumferences of a million circles with radii ranging from 1cm to 1km, in increments of 1cm.
2. return the array of circumferences to a text file called `circumferences.txt`

- You do not need to understand all that's happening here, though you should be able to get a sense of the Python script

```cpp
#include <fstream>
#include <algorithm> // for std::transform
#include <cmath>     // for M_PI
#include <iostream>  // for std::cout etc
#include <numeric>   // for std::iota
#include <vector>    // for awesome

// Script based on https://stackoverflow.com/questions/41030911/
    how-to-multiply-a-vector-and-scalar-in-c

int main() {
  std::vector<int> vec1(1000000);
  std::iota(vec1.begin(), vec1.end(), 0);

  int pi = 3.14159;

  std::vector<double> vec2(vec1.size()); // vec2 needs to be as big or bigger than vec1

  std::transform(vec1.begin(), vec1.end(), vec2.begin(),
                 [pi](int i) { return i * pi * 2; });


std::ofstream outFile("circumferences.txt");
    // output to file
    for (const auto &e : vec2) outFile << e << "\n";
}
```

In Python:

```
pi = 3.14159

radii = range(1, 1000000, 1)

circumferences_list = []

for element in radii:
    circ = element*(pi*2)
    circumferences_list.append(circ)

with open("circumferences_python.txt", 'w') as file_handler:
    for item in circumferences_list:
        file_handler.write("{}\n".format(item))
```

Let's try it

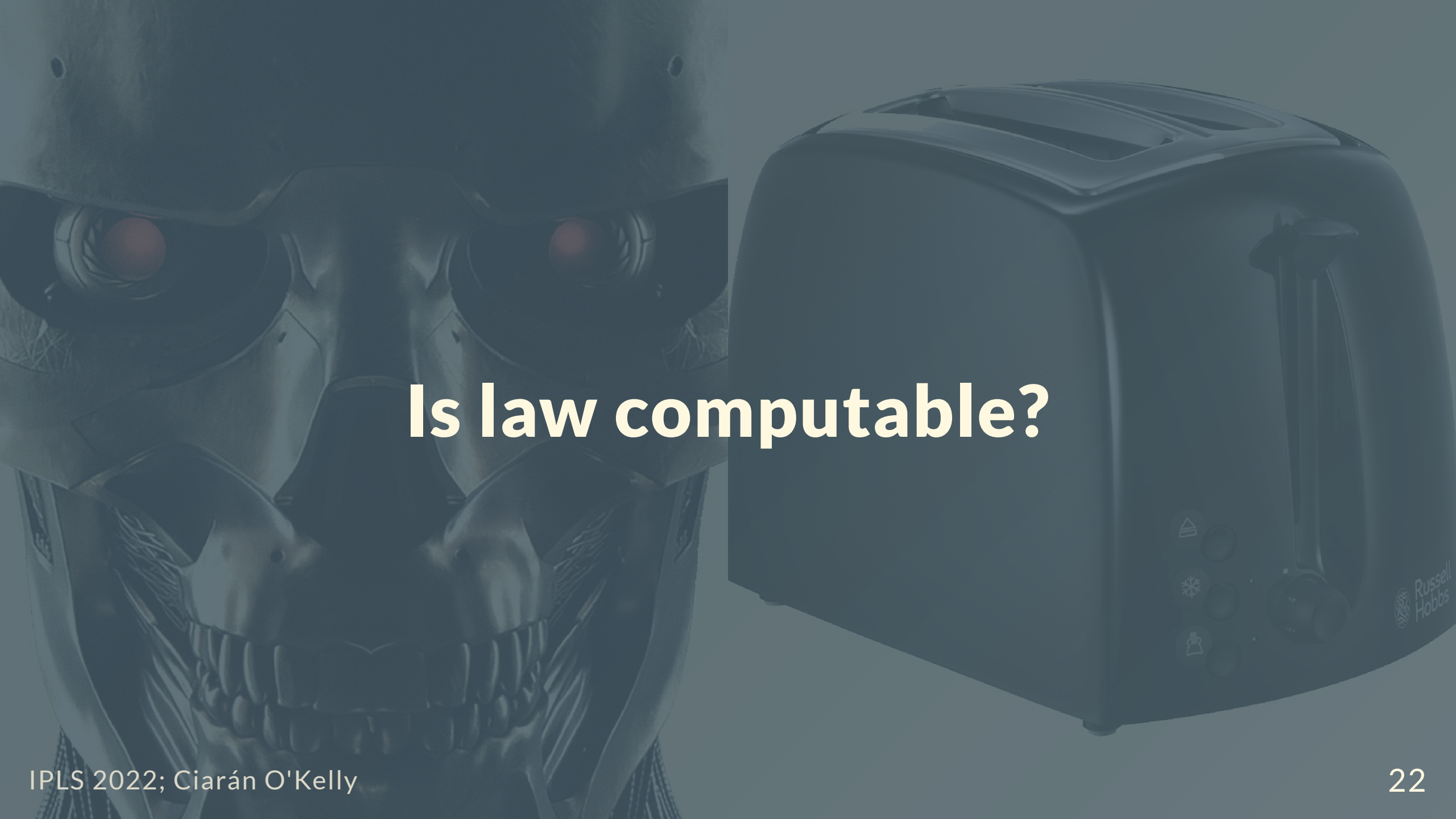
(**hopefully by switching to a Jupyter notebook**)

Is law a series of rules that can be applied 'algorithmically' as we did with our simple chatbot?

Jennifer Cobbe says we cannot teach computers to 'do' law in this way:

> “ law functions as a reflexive societal institution; a construct of society, it not only reflects society, but incorporates the assumptions, priorities, and values of those who act within it (lawyers, legislators, judges, legal academics, and so on), and reproduces that society along those lines...Legal AI proponents misunderstand the nature of both the technology and the law, with the result that they mistakenly believe that the former could adequately and reliably replace the latter ”

Cobbe J, ‘Legal Singularity and the Reflexivity of Law’ in Simon Deakin and Christopher Markou (eds), Is Law Computable? Critical Perspectives on Law and Artificial Intelligence (Hart 2020)

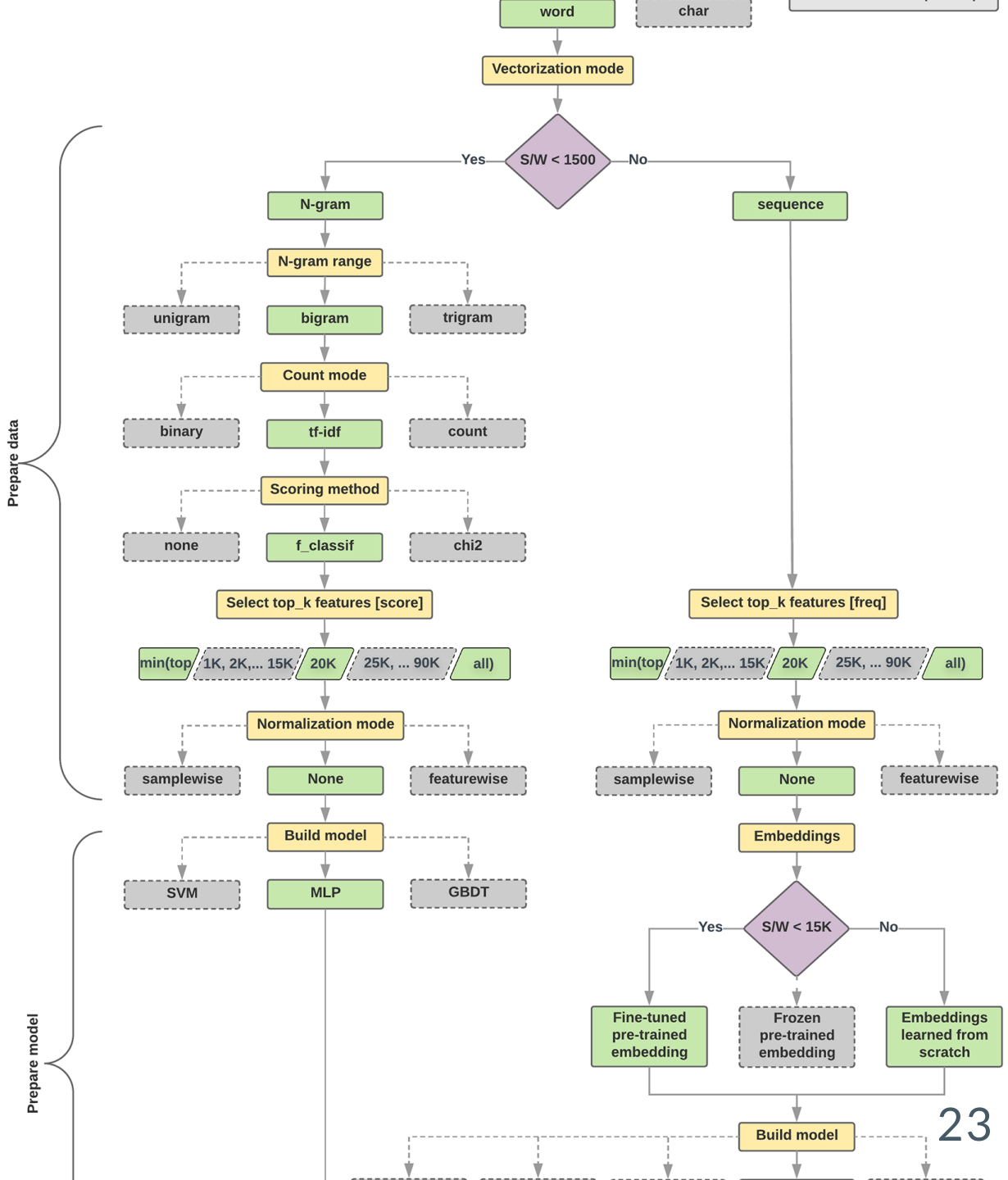


Is law computable?

Tomorrow

AI and its workflows

1. Input (data)
2. Process (algorithms)
3. Output (impact)



Data types

Data comes in many types and it is worth thinking about what they tell us about what computers do

Numeric data

These are numbers in essence, specifically numbers that you can perform statistical operations on (mean, median etc).

In Python, numbers are stored as 'integers' (whole numbers: 1, 2, 3 etc) or 'floating point numbers' (1.25, 3.14159 etc)

Categorical/Factoral Data

Categorical data may be stored in Python as integers (`int`) or as textual 'strings' (`str`).

Text:

- tall
- short

Ordinal

- tall = 1
- short = 0

Boolean

- True
- False

Beware ordinal data especially. Imagine a list of results from a survey of people's heights:

Categorical/Factoral

```
example_list = ["oak", "sycamore", "fir", "ash", "alder", "beech"]
type(example_list[0])

str
```

You cannot perform (most) statistical operations on categorial data. There is no average to get because the numbers are simply symbols for the categories.

That said, computer programmes very often represent categorical data as ones and zeros etc. Care is needed to understand what the data is.

You are endlessly scored and categorised

Textual and natural language data

Strings can also be stored and manipulated in Python.

```
example_list = ["oak", "sycamore", "fir", "ash", "alder", "beech"]

for i in example_list:
    print("The tree is of species", i)
```

```
The tree is of species oak
The tree is of species sycamore
The tree is of species fir
The tree is of species ash
The tree is of species alder
The tree is of species beech
```

Python Object Types: Lists, Tuples, Dicts etc

Python has a number of object types. As we see above, variables can be stored in objects called **lists**. Lists are denoted by square brackets.

```
example_list = ["oak", "sycamore", "fir", "ash", "alder", "beech"]
```

Lists are mutable and extensible

```
example_list.append("chestnut")
print(example_list)
```

```
["oak", "sycamore", "fir", "ash", "alder", "beech", "chestnut"]
```

Python Object Types: Lists, Tuples, Dicts etc

Tuples are *immutable*. That means that they you can't change them within a programme:

```
example_tuple = (1, 2, 3, 4, 5)

example_tuple[0] = 6
```

```
    TypeError                                 Traceback (most recent call last)

    ~\AppData\Local\Temp/ipykernel_6072/2285122387.py in <module>
    ----> 1 example_tuple[0] = 6


    TypeError: 'tuple' object does not support item assignment
```

Dictionaries

Dictionaries are *named* lists or tuples. They are important, for instance in the construction of Pandas dataframes (which are a lot like Excel spreadsheets).

```
trees = {"species": ["Oak", "Sycamore", "Fir", "Ash", "Alder", "Beech"],\
         "age": [240, 82, 34, 22, 39, 102],\
         "height_metres": [9, 8, 12, 11, 5, 12]}
```

Thank you