Python: lists and other demons

Paul Bradshaw

This week:

- **Storing** data: lists and dictionary variables
- Recipes for problem-solving: **functions**
- Books of recipes: **libraries**

First: the notebooks

- What problems did you have (oh goody)

Some highlights...

`.upper()`

`.lower()`

`.replace()`

`.count()`

`.find()`

`.replace()` (Sam)

```
[ ]  mylist = [3,5,7,9,11,13,15]
```

```
print(mylist)
```

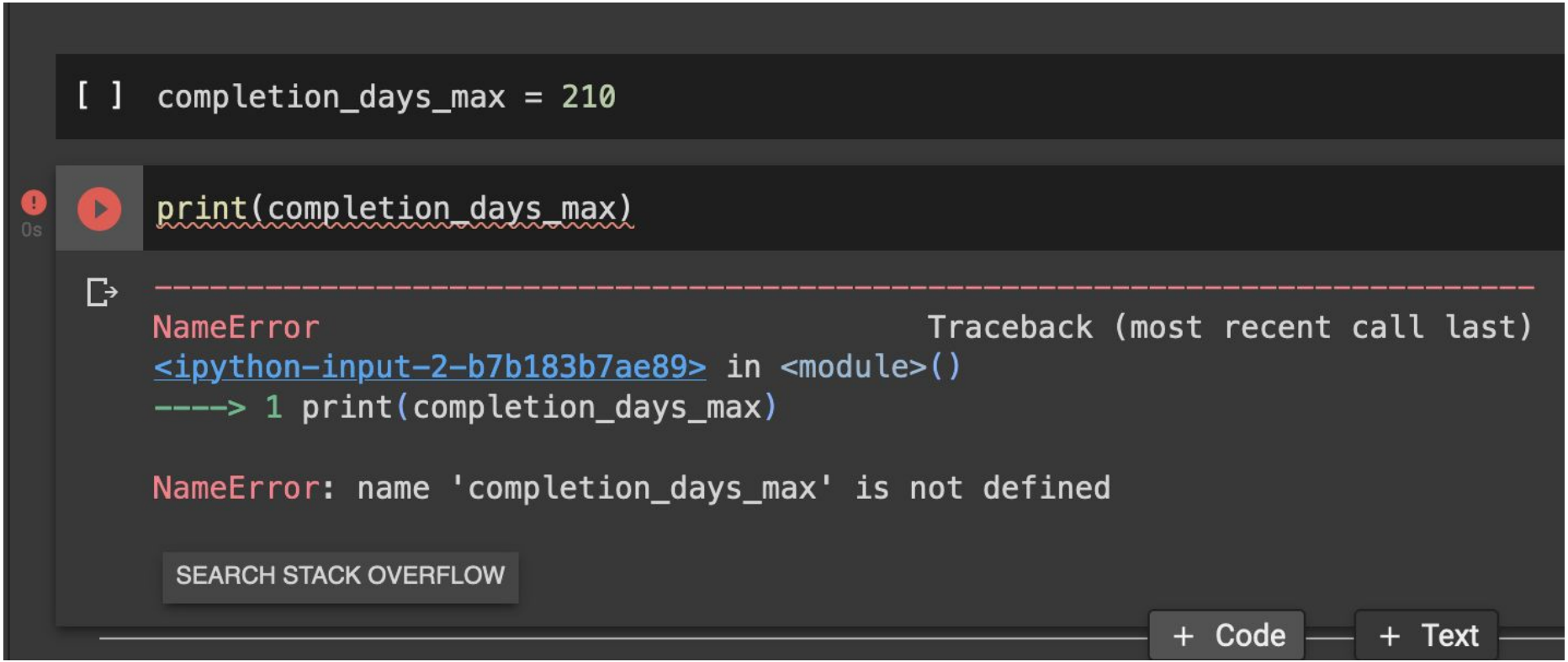
```
[3, 5, 7, 9, 11, 13, 15]
```

```
[ ]  listoflawfirms = ("Schillings, Carter Ruck, Freshfields, Slaughter & May, Hogan Lovells")
```

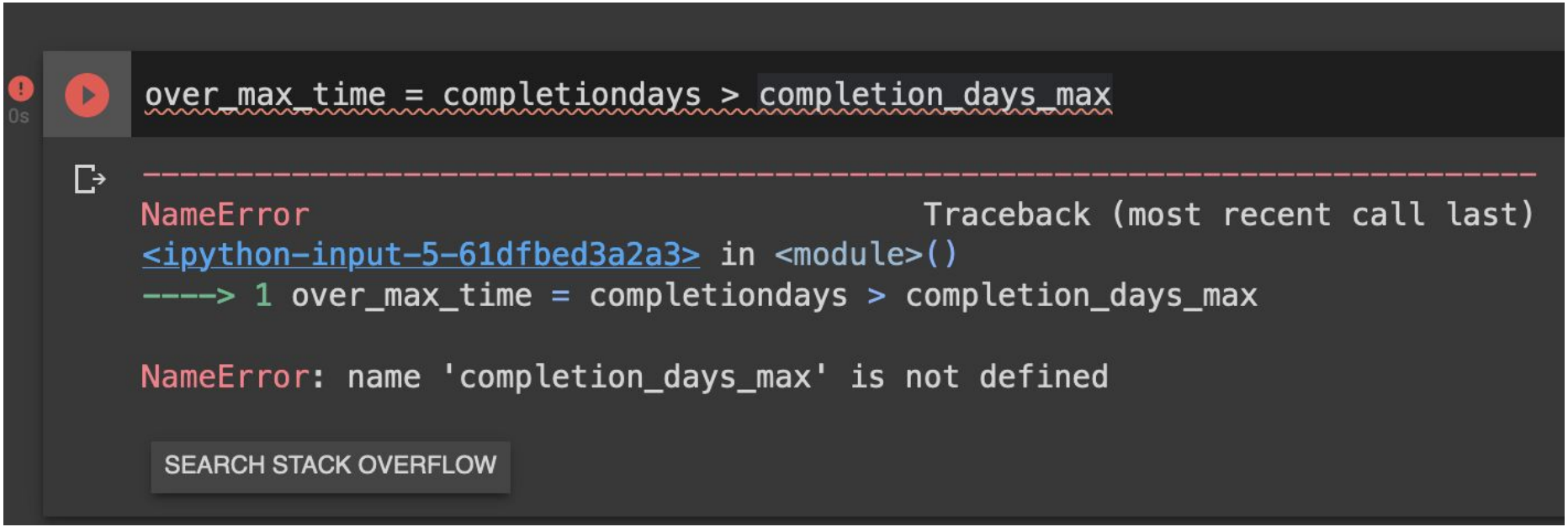
```
[ ]  print(listoflawfirms)
```

```
Schillings, Carter Ruck, Freshfields, Slaughter & May, Hogan Lovells
```

Ed: https://colab.research.google.com/drive/14zBjjFt6qKR1wzNMx9kbxCOZYcEiBvXq

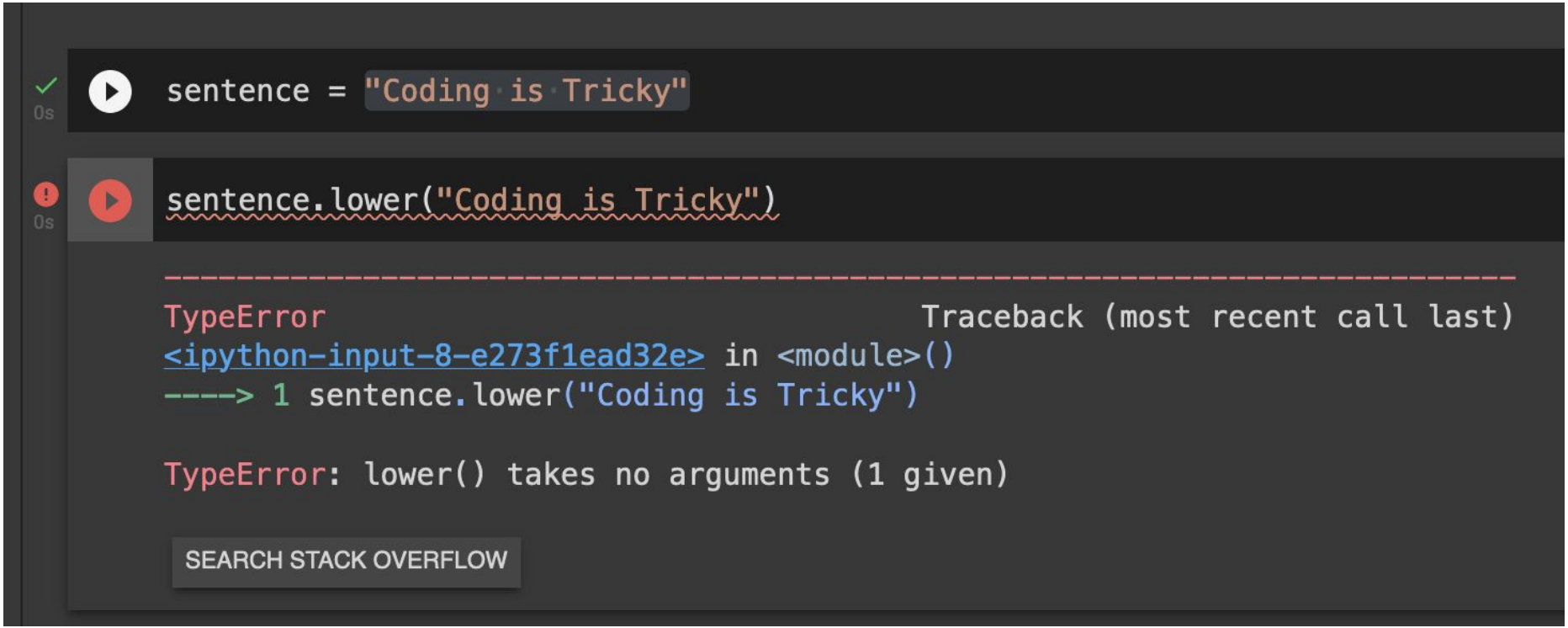


Vicky:
https://colab.research.google.com/drive/1HMdqR9vAcE7mo4_NdWr7BFjAxAmphAM6#scrollTo=qww-tQ0NBfhl

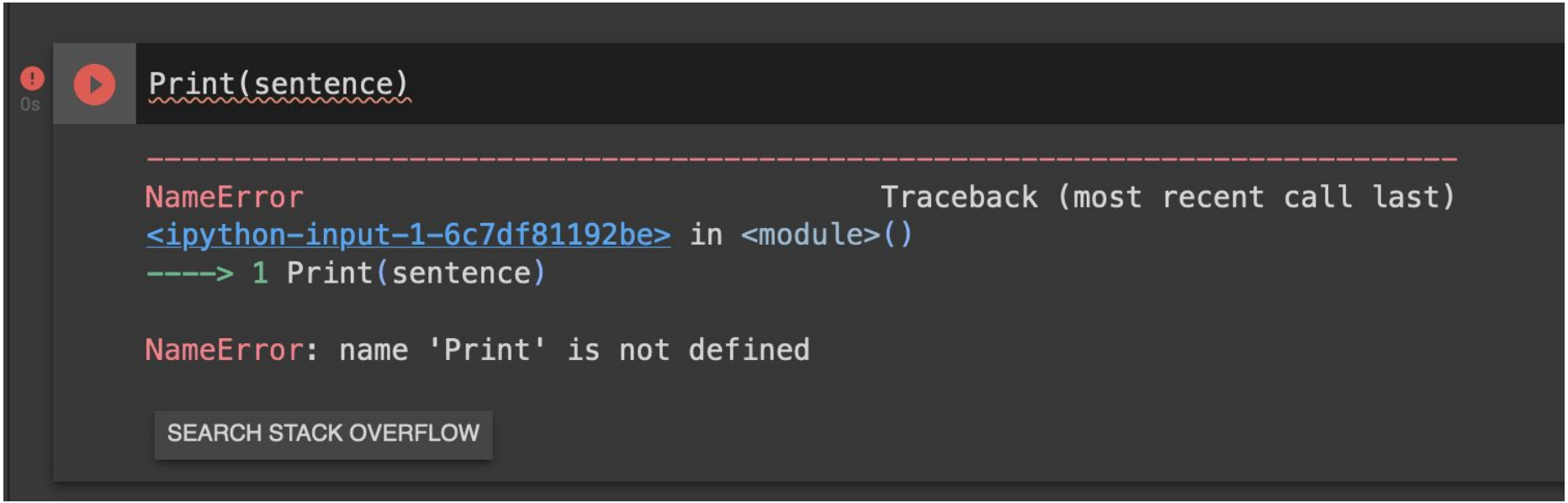


Vicky:
https://colab.research.google.com/drive/1HMdqR9vAcE7mo4_NdWr7BFjAxAmphAM6#scrollTo=qww-tQ0NBfhl

Vicky:
https://colab.research.google.com/drive/1HMdqR9vAcE7mo4_NdWr7BFjAxAmphAM6#scrollTo=qww-tQ0NBfhl



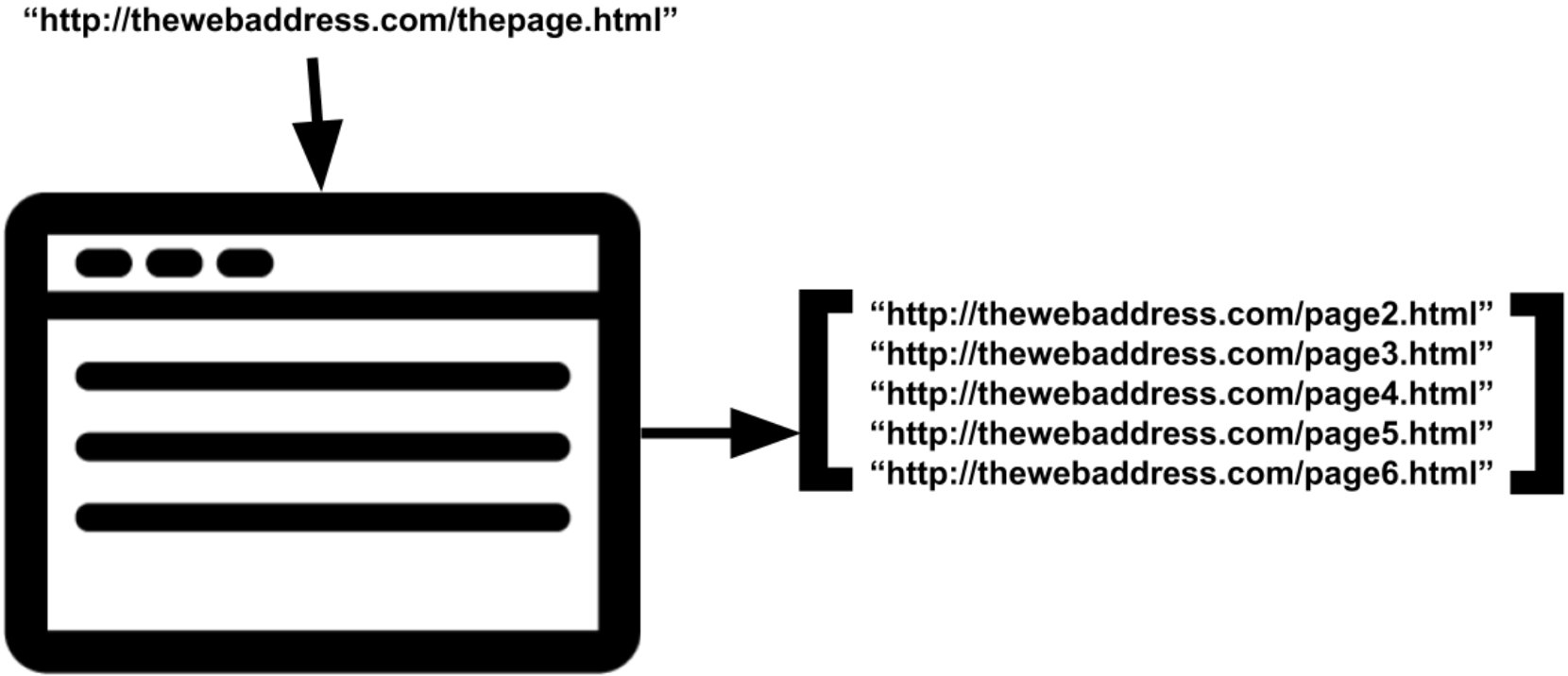
Vicky:
https://colab.research.google.com/drive/1HMdqR9vAcE7mo4_NdWr7BFjAxAmphAM6#scrollTo=qww-tQ0NBfhl



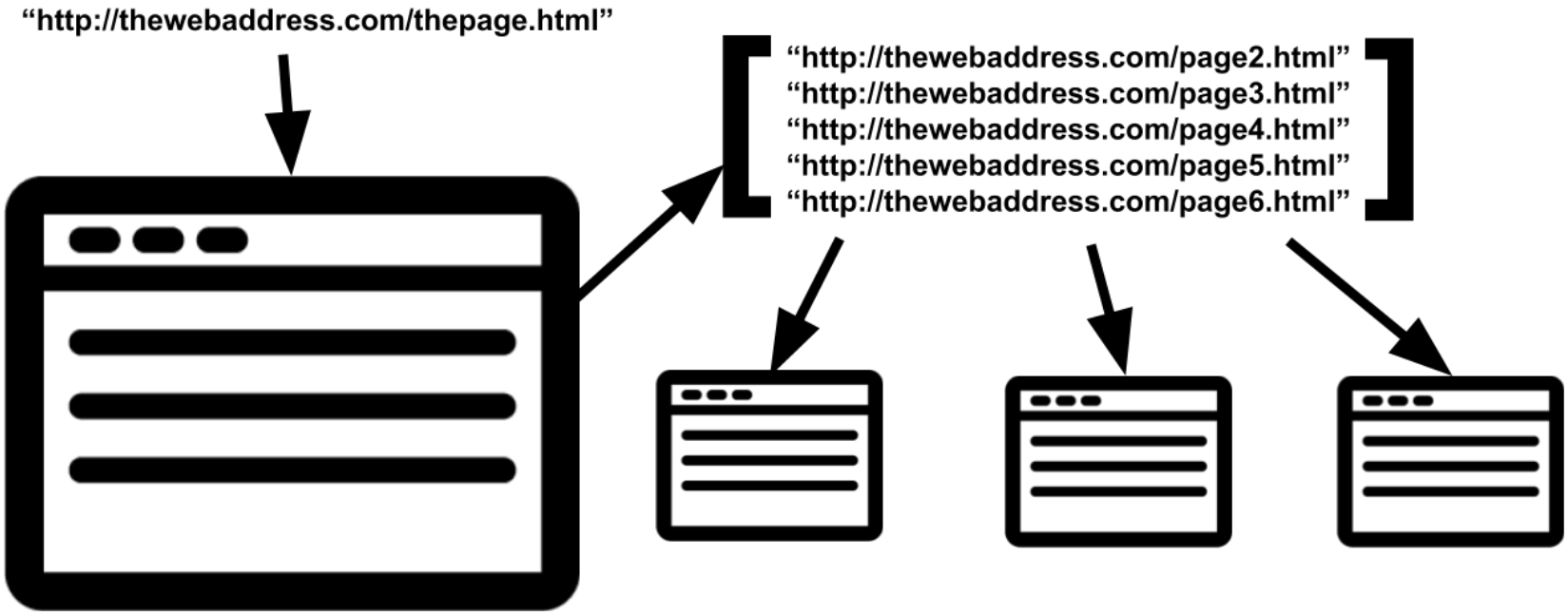
Vicky:
https://colab.research.google.com/drive/1HMdqR9vAcE7mo4_NdWr7BFjAxAmphAM6#scrollTo=qww-tQ0NBfhl

Lists

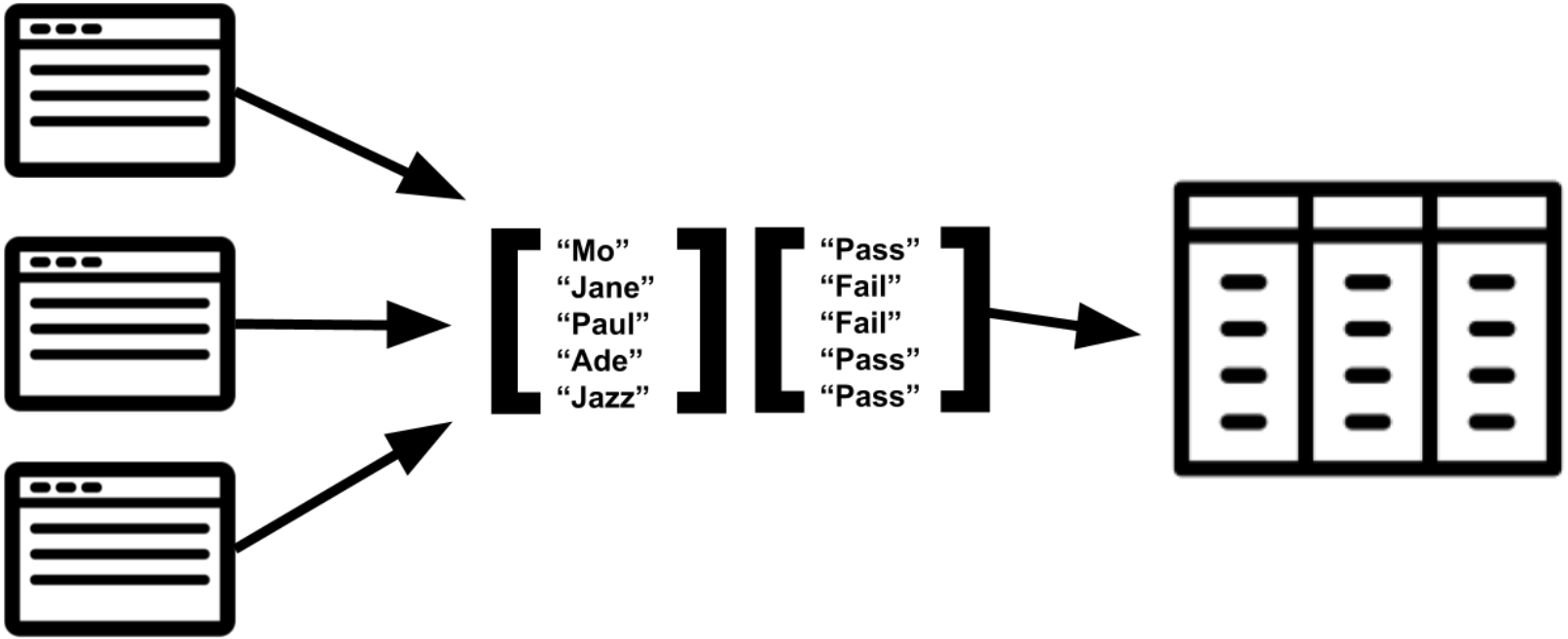
- A list starts and ends with square brackets
- `mydata = [“Kim”, “Jim”, “Tim”]`
- For storing multiple items
- Can be list of strings, integers, etc.



Go to a page -> scrape HTML -> extract links from HTML



Go to a page -> scrape HTML -> extract links from HTML -> loop through links -> scrape HTML



Loop through each page -> scrape HTML -> extract specific info from HTML -> add row to table -> download

Lists are **for** looping!

- Loop through a list using **for**

```
for i in mylist:
```

- The ‘i’ is a name you are giving each item
- The colon introduces intended commands

Indent what happens for each item

```
for i in mylist:
    print(i)
```

- Indented lines run as many times as items
- Doing something with ‘i’ each time, e.g. scrape

Dictionaries (‘dicts’)

- A dictionary starts and ends with curly brackets
- Contains a list of pairs, joined by a colon
- `mydict = {“dept” : “DfE”, “fois”:5}`
- Called ‘key-value’ pairs because used for storing labels (keys) and data (values)

[Accessing] items in dicts or lists

- Access items using a square bracket after the variable name: in a dictionary use a **key**; with a list use an **index** (position)

```
mydict['dept']
```

```
mylist[0]
```

Python uses a 0-based index

- Python starts counting at 0, so the first item is at ‘position 0’
- The second is at ‘position 1’
- And so on.

Functions

- Recipes for solving a particular problem
- E.g. `len` measures length of a string
- Followed by parentheses containing any ingredients
- Multiple ingredients separated by a comma

Head to the repo...

github.com/paulbradshaw/pythonin12parts

- Click on the part2 folder
- 03listsDictsFunctions.ipynb

Lists, dictionaries - and functions

In this notebook we look at two more common types of variables: **lists** and **dictionaries**, how to recognise them, how to create them, and what you might use them for as a journalist.

Lists are like columns of data

A list is a type of variable that allows you to store *multiple* values. It might be a list of numbers, a list of strings, a list of `True/False` values, or a mix of those.

You can even have a list of lists, but that's a bit too mind-bending to get into now.

To create a list, you need to put **square brackets** around your list of items, and **separate each one with a comma**, like so:

In [ ]:
```
#this is a list of numbers
refusals = [1, 11, 4, 0]
#this is a list of strings
orgs = ["AGO", "Cabinet Office", "DBEIS", "DCMS"]
#this is a list of Booleans
dept_tf = [False, False, True, True]
```

Those square brackets will also show when a list is printed - so it's a key way to recognise that you're dealing with a list (which might be the case if you've imported some data and then extracted one column).

In [ ]:
```
print(dept_tf)
```

```
[False, False, True, True]
```

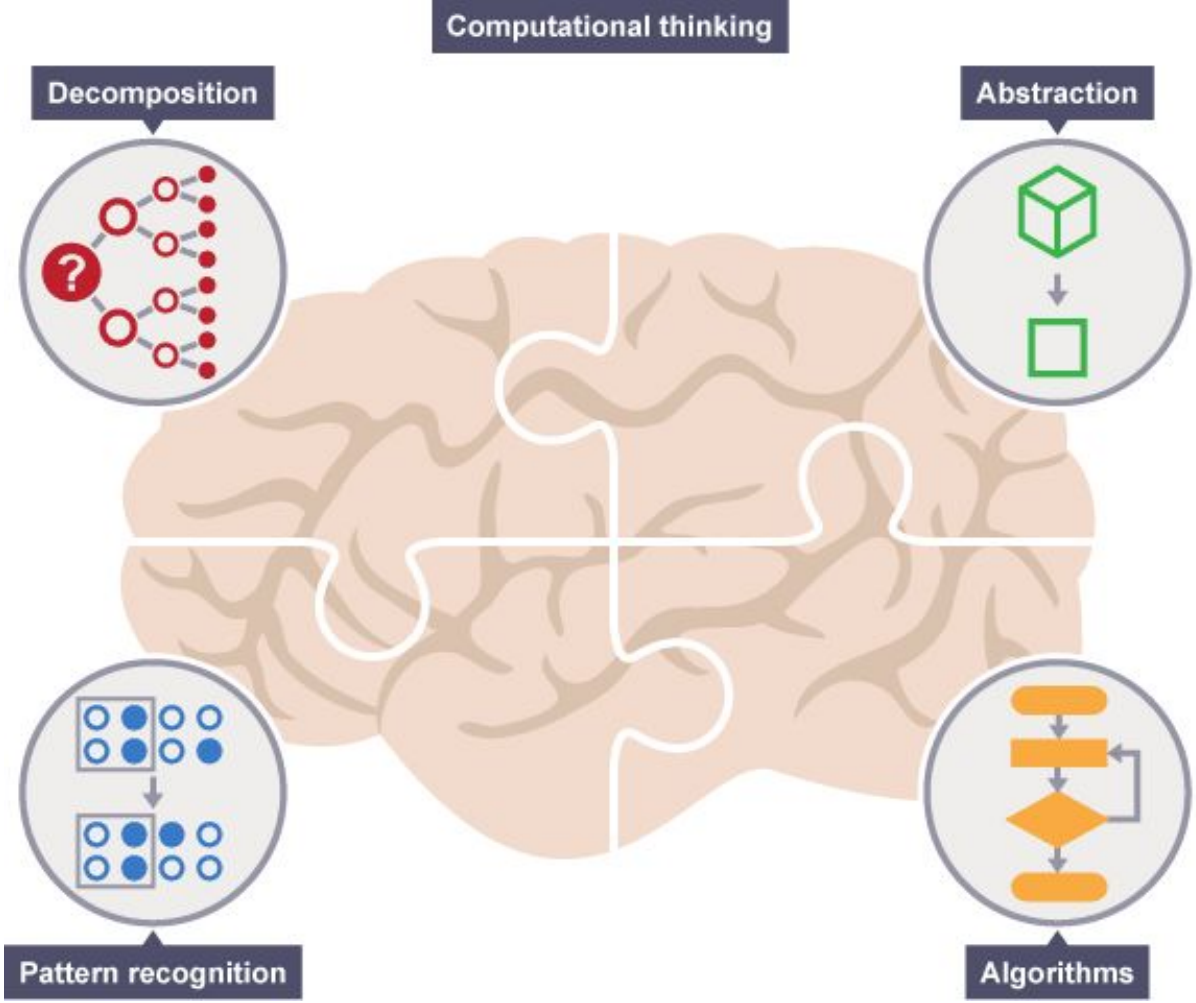
The data in the first two lists is from the Freedom of Information statistics: April to June 2021 bulletin - specifically the data tables in the sheet called '10_Exemptions'.

Libraries and functions

- Libraries give you access to more functions
- To use those functions, the name of the library is used too, e.g. scraperwiki.scrape

User-defined functions

- If you want to repeat a block of code you can store it in a function and ‘call’ it
- Create your own function with `def`



Read more here

An example:

I want the area codes for a column of phone numbers
The phone number is in two parts
The second part is always 7 digits
If I strip away those 7, I’m left with the area code

Creating algorithms with nested formulae

```
=LEFT(A2,LEN(A2)-7)
```

1. Measure length of A2
2. Subtract 7
3. Grab that many characters from A2, starting from the left

E.g. The scraping problem

1. Go to a URL and fetch the HTML
2. Explore the HTML and find things that match a pattern (URLs)
3. Store all of those in a list
4. Go to each URL and fetch the HTML
5. Explore the HTML and find matches to a pattern
6. Store those matches in a dictionary (or lists)
7. Combine those dictionaries/lists into a table (a dataframe)
8. Export as a CSV

Head to the repo...

github.com/paulbradshaw/pythonin12parts

- Click on the part2 folder
- 04libraries.ipynb

Using libraries in Python to access more functions

All of the functions in previous notebooks are built into Python. But you can access thousands more functions from **libraries** that people have created (sometimes called **modules**).

A **library** is a collection of functions and other code that someone has created - typically to solve a particular problem. For example:

- The `pandas` library was created to solve problems relating to data analysis
- The `matplotlib` library was created to add extra visualisation functionality to Python
- The `scraperwiki` library was created to solve scraping problems. It's not the only one: Beautiful Soup is another library for this, too.
- The `pdf2xml` library was designed to help deal with PDFs in Python (by converting them to xml)
- The `re` library provides functions to use **regular expressions** in Python, in order to describe patterns you might want to match (and fetch) in text or webpages.
- The `os` library was designed to add "operating system dependent functionality" such as reading or writing files.
- The `Scikit-learn` library allows you to use machine learning in Python
- The `NumPy` library has lots of functions to do more advanced mathematical processing

A good principle to bear in mind when using programming languages is that if you come across a problem, chances are that someone else has already come across a similar problem - and create a library to solve it. A bit of googling around can help you find that, and learn how to use it to help solve your problem.

To use a library's functions you need to **import** it.

This is done by using the conveniently-named `import` function, followed by the name of the library.

In [ ]:
```
#import the pandas library to use its functions
import pandas
```

Methods

- Like functions but exist within an ‘object’
- An analogy: the ‘carmaker’ library has functions that can make cars
- but the resulting cars themselves have methods that can do things like ‘drive’

Key points

- Use lists to store items and loop through it
- Use dictionaries to store data and access it
- Find functions to solve problems

Next...

- Create your own Colab notebook and create lists/dictionaries to store data
- Google how to access items from those variables
- Share the notebook with me!

Plus...

- Import some data using pandas
- Try to solve a problem - find a function or library that is supposed to fix it