Recall: indexing

Indices in math:

$$\text{Given } X = \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}$$

What is X_2? $X_0 = 0$, $X_1 = \frac{\pi}{2}$, $X_2 = \pi$, ...

Lists and strings (and more) can be *indexed*

```
X = [0, pi/2, pi, 3*pi/2]
s = X[2] / pi
```

Python indices

Start counting from 0

Will cause an error if *out of range*

Can be *negative!*

- start counting from the back
- range checks still apply!

Names

Example:

```
names = ['Jonathan', 'Anderson']
```

```
first_name = names[0]
```

```
initial = first_name[0]
another_way_to_write_initial = names[0][0]
```

What's the last letter of your last name?

Problem

Find the position of a name within a list of names

Think about:

1. Approach: how would *you* do this?
2. Data representation: what do we need? How is it stored?
3. Algorithm: how can you *explain* the approach?
4. Code: how can you express this in Python?

Slicing

Lists, strings (and more) can be *sliced*

$$\text{Given } X = \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}, s = \sum_{i=1}^{3} X_i$$

$$s = X_1 + X_2 + X_3 = \frac{\pi}{2} + \pi + \frac{3\pi}{2} = \frac{6\pi}{2} = 3\pi$$

```
X = [0, pi/2, pi, 3*pi/2]
s = sum(X[1:3])
```

Slices

A *slice* of something (list, range, string, etc.)

From a beginning (*inclusive*) to and end (*exclusive*)

Much like a `range`!

```
start = 1
end = 3

# The following are almost the same thing:
print(X[start:end])

for i in range(start, end):
    print(X[i])
```

Advanced slicing

Also like a `range`, can specify an *increment:*

```
X[0:4:2]
X[::2]
```

Can even specify a negative increment:

```
X[4:0:-2]
X[::-1]
```

Sequence functions

`sorted()`

`reversed()`

`sum()`

... all *built in*, no `import` necessary

Summary

Slicing and indexing

Reversing and sorting

Sorting and summing