INDEX BASE AND SLICE BOUNDS

RATIONALE

Valid Questions

Why does sequence indexing start at **0**, and not 1?

Why does a sequence slice `s[i:j]` include `s[i]`, but exclude `s[j]`?

this is not just an arbitrary choice → there are rational and practical reasons behind doing so

We want to determine how we should handle sequences of consecutive integers

→ represent positions of elements in a sequence

```
['a', 'b', 'c', 'd']
  1    2     3    4
  0    1     2    3
```

Slice Bounds

Consider the following sequence of integers `1, 2, 3, …, 15`

How can we describe this range of numbers without using an ellipsis (…)?

a) `1 <= n <= 15`

b) `0 < n <= 15`

c) `1 <= n < 16`

d) `0 < n < 16`

(b) and (d) can become odd at times.

Suppose we want to describe the unsigned integers `0, 1, 2, …, 10`

Using (b) or (d) we would need to use a signed integer for the lower bound:

b) `-1 < n <= 10`

d) `-1 < n < 11`

Now consider this sequence: `2, 3, …, 16`

a) `2 <= n <= 16`

c) `2 <= n < 17`

How many elements are in this sequence? 15

Calculating number of elements from bounds in (a) and (c)

a) `15 = 16 – 2 + 1` `# = upper – lower + 1`

c) `15 = 17 – 2` `# = upper – lower`

So, (c) seems simpler for that calculation

We'll get to a second reason in a bit, but for now we'll use convention (c)

Starting Indexing at `0` instead of `1`

When we count elements we naturally start counting at `1`, so why start indexing at `0`?

Consider the following sequence:

	`2, 3, 4, …, 16`	sequence length: `15`
index `n` (1 based)	`1, 2, 3, …, 15`	`1 <= n < 16` upper bound = length + `1`
index `n` (0 based)	`0, 1, 2, …, 14`	`0 <= n < 15` upper bound = length

For any sequence `s`, the index range is given by:

`0` based: `0 <= n < len(s)`

`1` based: `1 <= n < len(s) + 1`

So, `0` based appears simpler

Another reason for choosing 0 based indexing

Consider this sequence:

a, b, c, d, … z

1 based 1, 2, 3, 4, …, 26

0 based 0, 1, 2, 3, …, 25

How many elements come before d? 3 elements

1 based index(d) → 4 4-1 elements

0 based index(d) → 3 3 elements

So, using 0 based indexing, the number of elements that precede an element at some index

→ is the index itself

Summarizing so far…

choosing `0` based indexing for sequences

describing ranges of indices using `range(l, u)` → `l <= n < u`

we have the following results

the indices of any sequence `s` are given by: `range(0, len(s))` `[0 <= n < len(s)]`

first index: `0` last index: `len(s)-1`

number of indices before index `n`: `n`

the length of a `range(l, u)` is given by: `l - u`

```
    0   1   2       25
s = [a, b, c, …, z]   len(s) → 26
```

indices → `range(0, 26)`

`n` elements precede `s[n]`

Slices

Because of the conventions on starting indexing at `0` and defining ranges using `[lower, upper)`

inclusive exclusive

we can think of slicing in these terms:

Each item in a sequence is like a box, with the indices between the boxes:

First 2 elements: `s[0:2]` `s[:2]`

Everything else: `s[2:6]` `s[2:]`

In general we can split a sequence into two with k elements in the first subsequence: `s[:k]` `s[k:]`