ZIPPING ITERABLES

The `zip` Function → lazy iterator

We have already seen the `zip` function

It takes a variable number of positional arguments – each of which are iterables

It returns an iterator that produces tuples containing the elements of the iterables, iterated one at a time

It stops immediately once one of the iterables has been completely iterated over

→ zips based on the shortest iterable

```
zip([1, 2, 3], [10, 20], ['a', 'b', 'c', 'd'])
```

→ `(1, 10, 'a'), (2, 20, 'b')`

```
itertools.zip_longest(*args, [fillvalue=None])
```

Sometimes we want to zip, but based on the longest iterable

→ need to provide a default value for the "holes" → `fillvalue`

```
zip([1, 2, 3], [10, 20], ['a', 'b', 'c', 'd'])
    → (1, 10, 'a'), (2, 20, 'b')

zip_longest([1, 2, 3], [10, 20], ['a', 'b', 'c', 'd'])
    → (1, 10, 'a'), (2, 20, 'b'), (3,None, 'c'), (None, None, 'd')

zip_longest([1, 2, 3], [10, 20], ['a', 'b', 'c', 'd'], -1)
    → (1, 10, 'a'), (2, 20, 'b'), (3,-1, 'c'), (-1, -1, 'd')
```

Code Exercises