INFINITE ITERATORS

`itertools.count` → lazy iterator

The `count` function is an infinite iterator

similar to `range` → `start`, `step`

different from `range` → no `stop` → infinite

→ `start` and `step` can be any numeric type

`float`

`complex`

`Decimal`

`bool` `False` → `0`
`True` → `1`

Example

```
count(10, 2)        → 10, 12, 14, …

count(10.5, 0.1)    → 10.5, 10.6, 10.7, …

takewhile(lambda x: x < 10.8, count(10.5, 0.1))
                    → 10.5, 10.6, 10.7
```

`itertools.cycle` → lazy iterator

The `cycle` function allows us to loop over a finite iterable indefinitely

Example

`cycle(['a', 'b', 'c'])` → `'a', 'b', 'c', 'a', 'b', 'c', …`

Important

If the argument of `cycle` is itself an iterator → iterators becomes exhausted

cycle **will still** produce an infinite sequence

→ does not stop after the iterator becomes exhausted

`itertools.repeat` → lazy iterator

The `repeat` function simply yields the same value indefinitely

```
repeat('spam')    → 'spam', 'spam', 'spam', 'spam', …
```

Optionally, you can specify a count to make the iterator finite

```
repeat('spam', 3) → 'spam', 'spam', 'spam'
```

Caveat

The items yielded by `repeat` are the same object

→ they each reference the same object in memory

Code Exercises