Generator Functions

- Use the **`yield`** statement
- Act as an iterable
- Can be used in a **`for`** statement
  - Almost anywhere that a collection is used
- Won't create big list objects
- Lazy — they don't compute anything unless forced to by another function consuming results
  - `list(some_function())`
  - Builds a list object by consuming values from the generator

The Syracuse Generator

```
def syracuse_iter(n):
    yield n
    while n != 1:
        n = syracuse(n)
        yield n
```

- Will *yield* the number,
- While n != 1, will apply the function and *yield* each subsequent value.
- And it's lazy!
  - Won't create a big list if stopped early