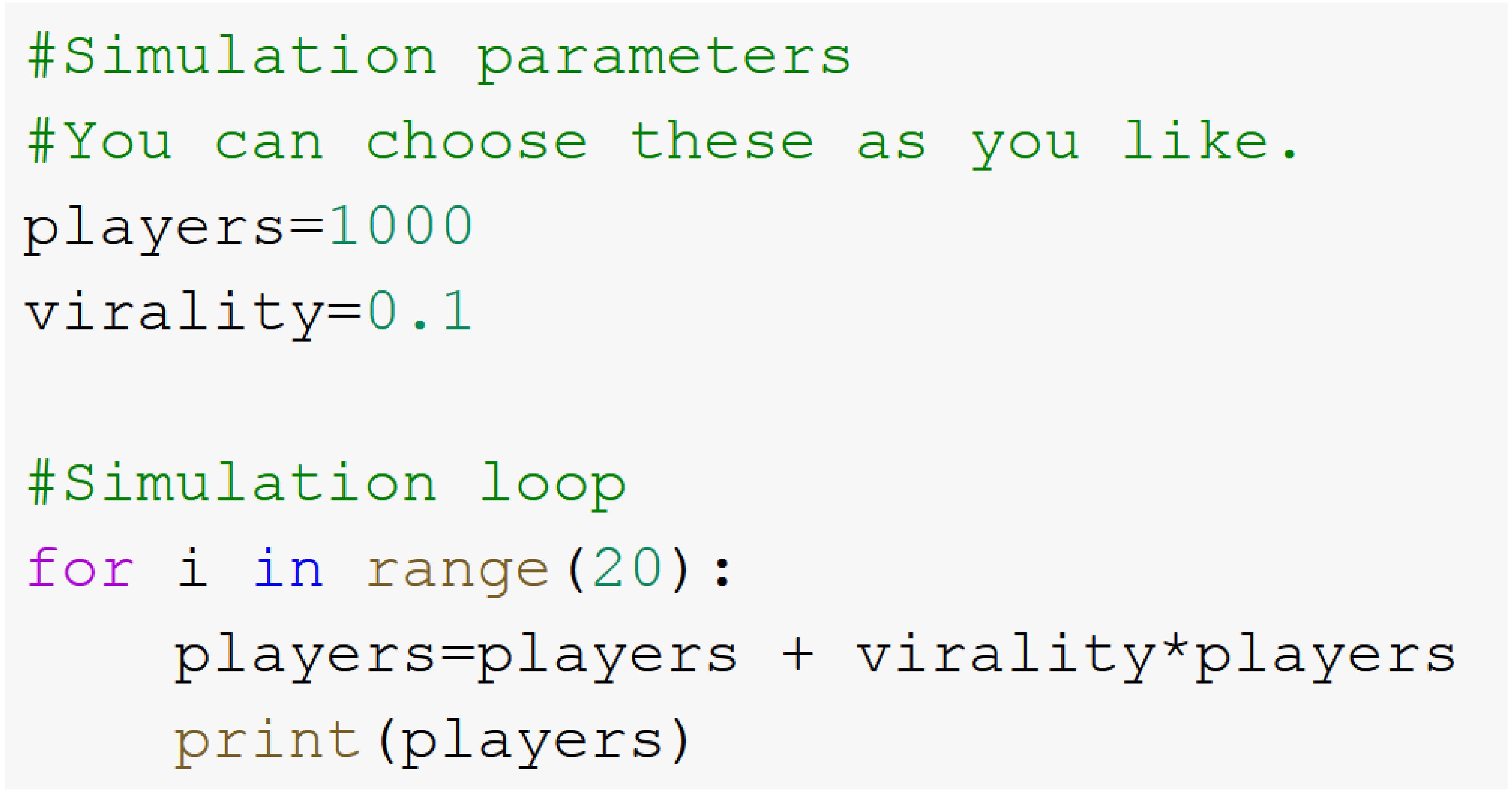
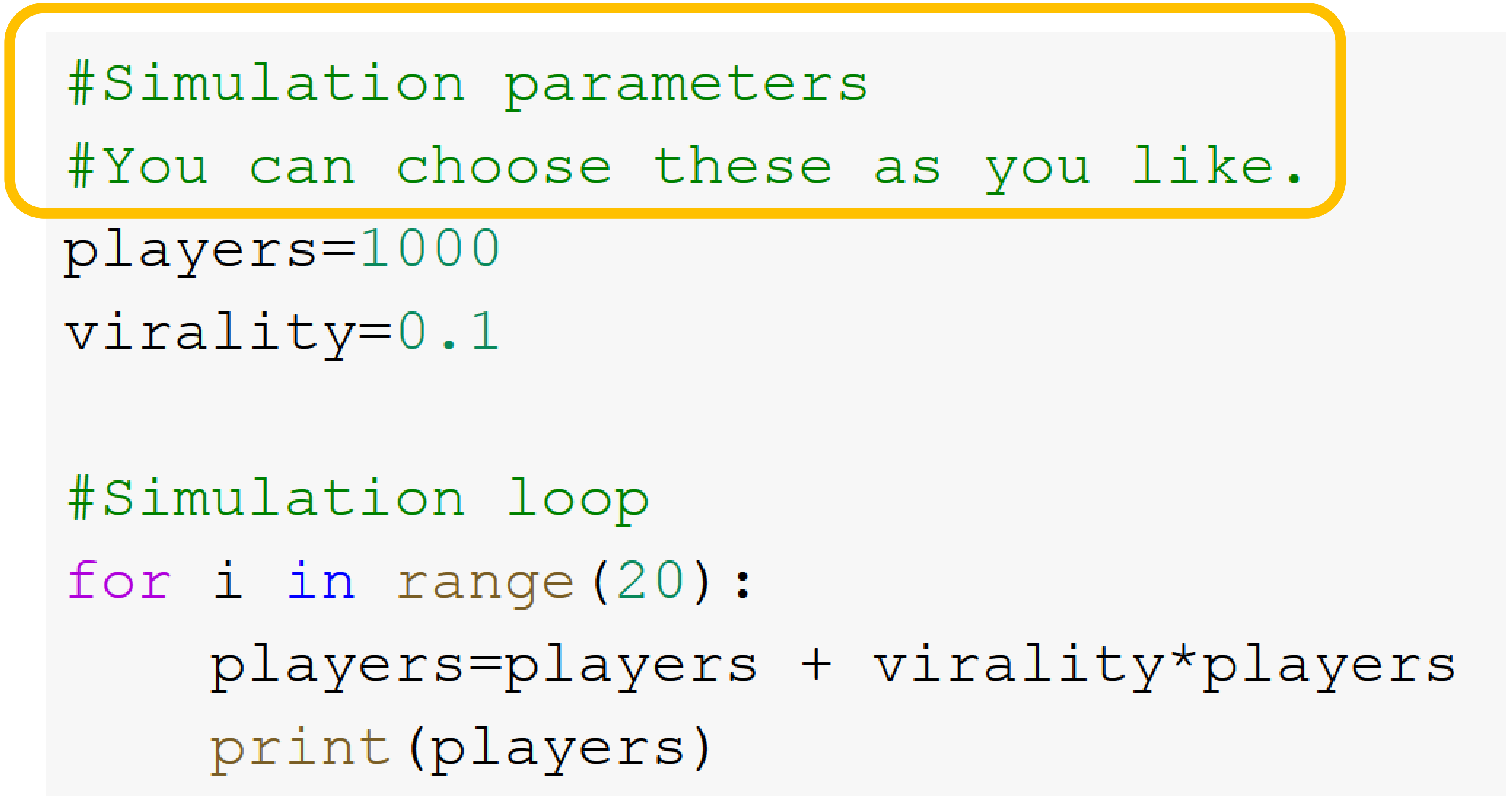
This is the first Python simulation in the RetentionAndVirality notebook. Let’s investigate it more closely to learn Python syntax...



```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

# means that the rest of the line is comments that explain the code.
The computer running the code ignores these.

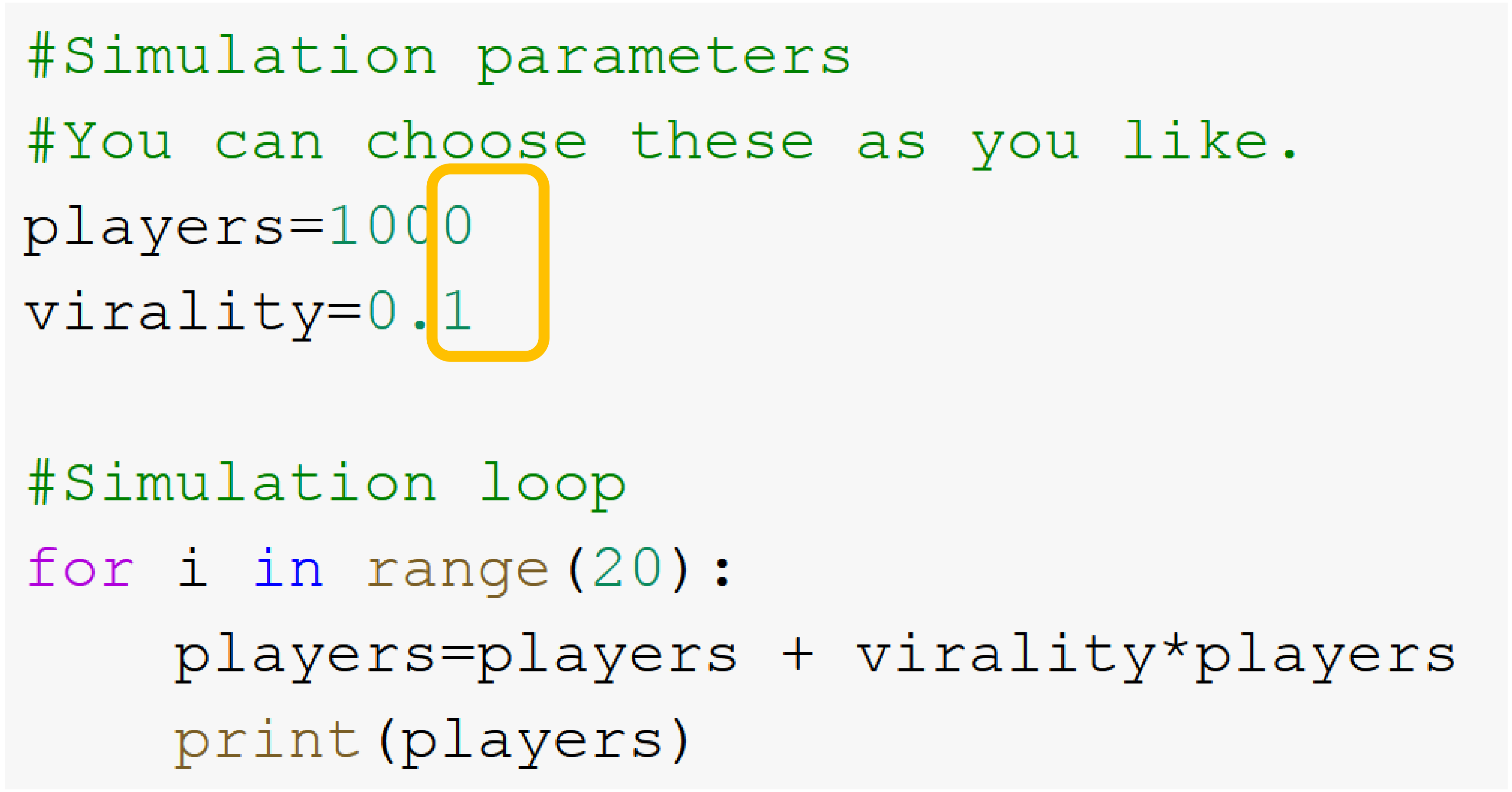


```
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

This introduces a variable called ”players” and sets its value to 1000. Python does not need type specifiers like C#’s ”int players=1000”

```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

In Python, you don’t need to add the ”;” at the end of a line.



```
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

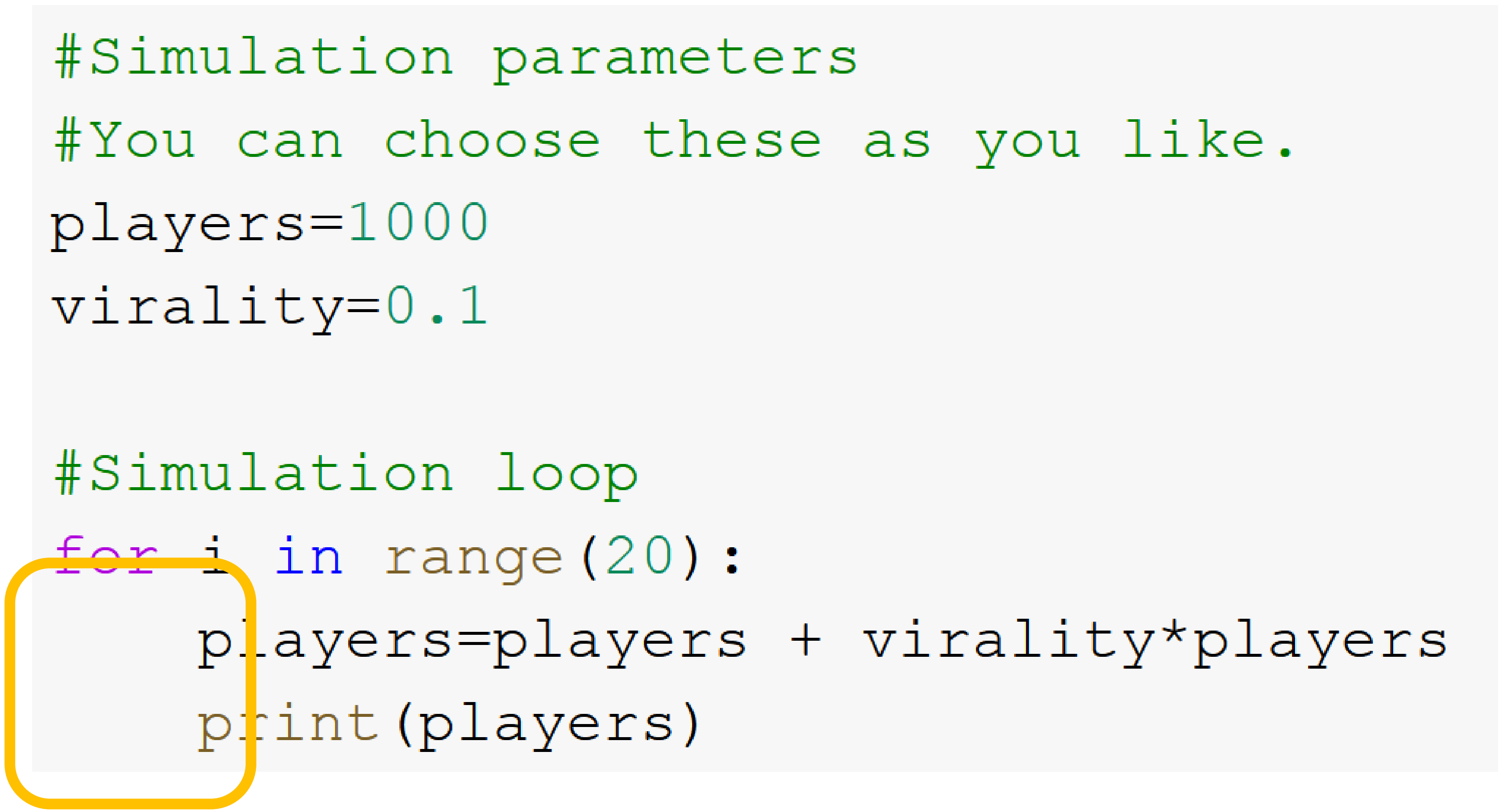
This declares a loop: All code inside the loop is run for 20 times.
In C#, this would be ”for (int i=0; i<20; i++)”

```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

For loop declaration ends with ":"

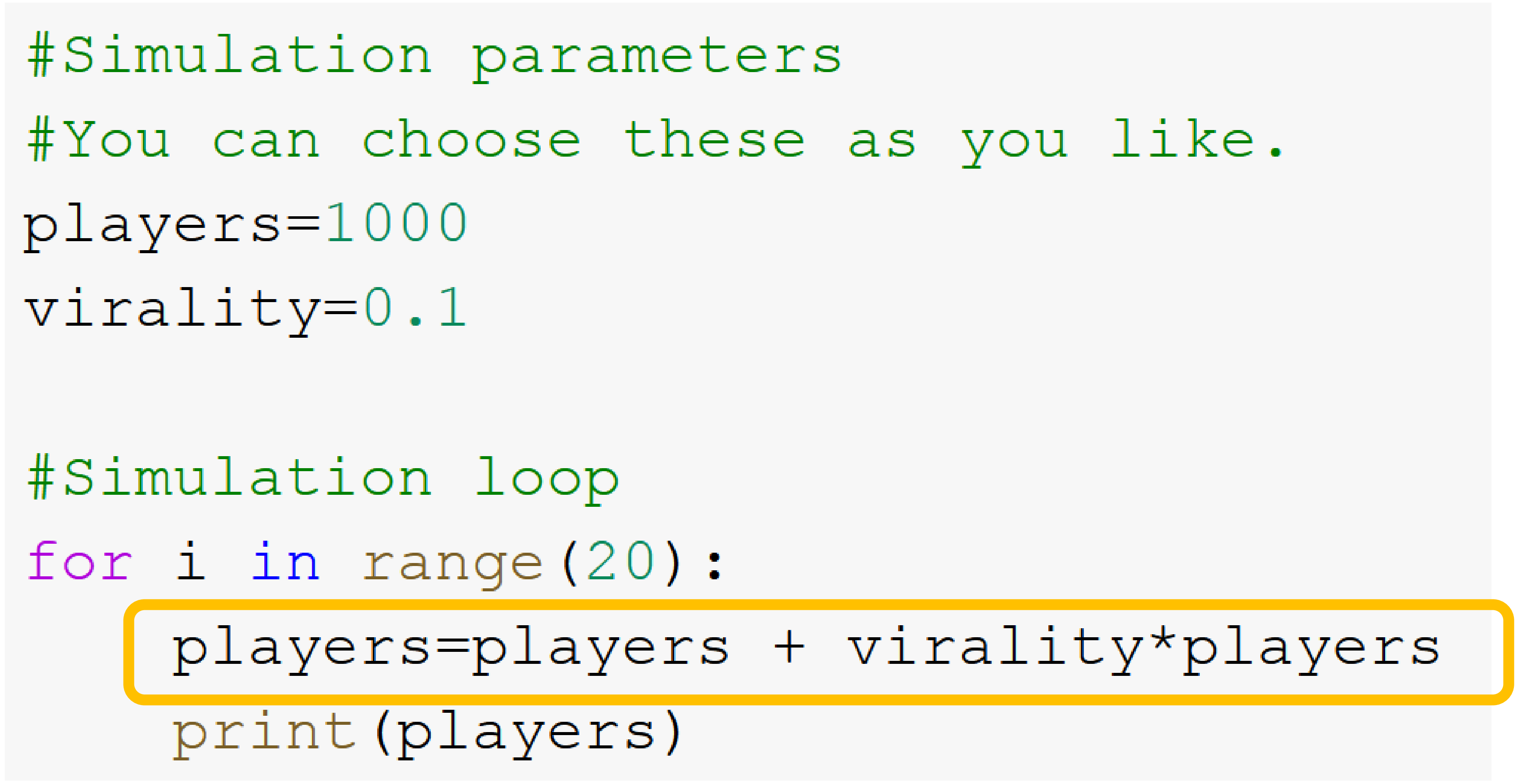
```
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

Everything after the ":" that is intended belongs inside the loop. In C#, the loop contents would be inside curly brackets.



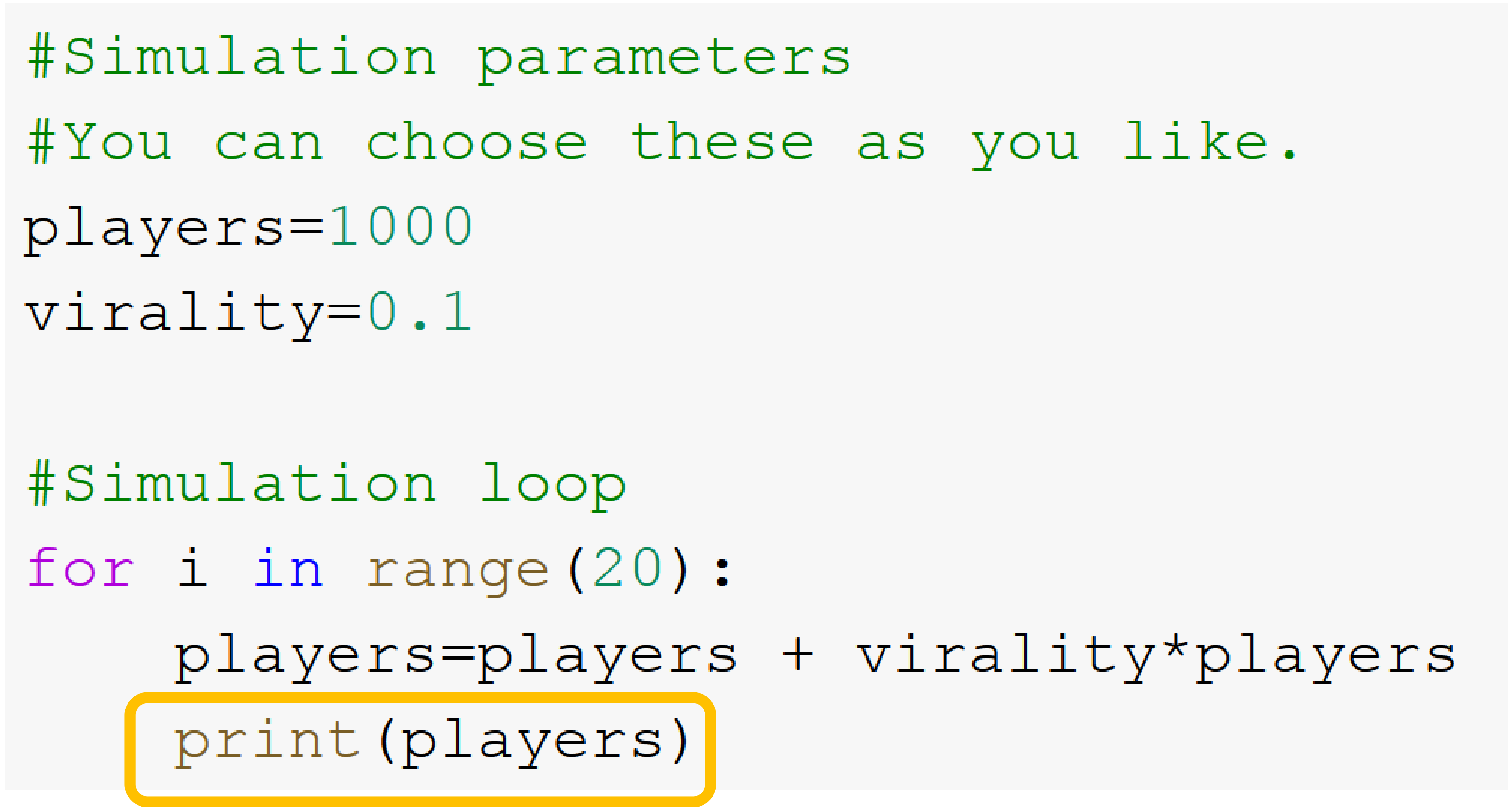
```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

This assigns the variable ”players” a new value based on its old value and the virality parameter. Mathematical expressions follow standard rules: Multiplication is executed before addition.



```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```

The print function prints things into the Colab output. In Python, print usually works no matter the variable type.



```python
#Simulation parameters
#You can choose these as you like.
players=1000
virality=0.1

#Simulation loop
for i in range(20):
    players=players + virality*players
    print(players)
```