What we did

```
class Book:
    def __init__(self, title, year):
        self.title = title
        self.year = year

    def is_antique(self):
        return self.year <= 1920
```

- Define class like `class Book:`
- Define methods with `def` inside
- Create instance like `Book('Emma', 1815)`
- Special method `__init__` to handle parameters

Objects

- With *objects* we can model objects we want our programs to handle.
- They have *attributes/properties*, some of them being *methods*. (Terminology sometimes differs.)
- There are some special methods (we have seen `__init__`)

The method `__str__`

The method `__str__` says how an object should be displayed. (What the function `str` should make of it.)

The method `__str__`

The method `__str__` says how an object should be displayed. (What the function `str` should make of it.)

```
class Book:
    ...

    def __str__(self):
        return self.title
```

The method `__str__`

The method `__str__` says how an object should be displayed. (What the function `str` should make of it.)

```
class Book:
    ...

    def __str__(self):
        return f"'{self.title}' by {self.author}"
```

Balls

...

Reading in *Thinking Python*

In *Think Python* Chapter 15–17 covers what we have done.

Reading in *Thinking Python*

In *Think Python* Chapter 15–17 covers what we have done.

Chapter 16, “Classes and functions”, talks about the difference between functions that *modify* some of its arguments and “pure functions” that act more like mathematical functions, just giving a new value without anything being changed.

This is an important distinction, but not really new just because we are talking about object-oriented programming.

Videos on their way

Also some material that I originally planned to say now will instead be in videos to appear on Studium today or tomorrow: Watch this week!