

Exercise walk-through

4.1

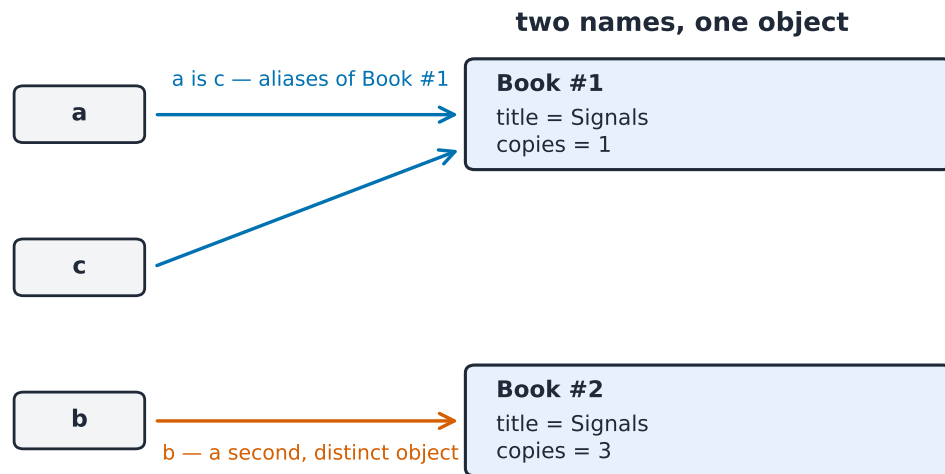
Objects and classes

University Computer Science

Objects and classes: blueprints, names, identity

- 1 **Blueprint** A class is a blueprint; an object is one thing built from it; the constructor seeds the **instance attributes**, methods take self.
- 2 **Lookup** Attribute lookup walks outward: **instance** dict, then class dict, then bases - assigning through an instance shadows the class value.
- 3 **Trace** Two Books made, take twice through alias a: copies reads 1 for a and c (one object, two names), 3 for b; the class counter reads 3.

Object and reference graph



Two names on one object: a and c alias the same Book, b is a second, distinct object.

Exercises 2.1-2.2

2.1 A class counts its instances in a class attribute starting at 1. Two instances are created. State where the counter lives and what it reads. [3]

2.2 After `d = c` on two Book references, `c.take()` runs once. State what `d.copies` shows and why. [3]

Exercises 2.1-2.2

2.1 A class counts its instances in a class attribute starting at 1. Two instances are created. State where the counter lives and what it reads. [3]

2.1

- On the class (class attribute) ✓.
- Reads 3: incremented once per construction ✓.

2.2 After `d = c` on two Book references, `c.take()` runs once. State what `d.copies` shows and why. [3]

2.2

- One less than before – the same object ✓.
- `d` is an alias of `c`: assignment copies the reference, not the object ✓.

Exercise 2.3

2.3 Two distinct Books hold identical fields. State the result of `==` (no `__eq__` defined) and of `is`, and name what each compares. [3]

Exercise 2.3

2.3 Two distinct Books hold identical fields. State the result of `==` (no `__eq__` defined) and of `is`, and name what each compares. [3]

2.3

- `==` is `False`: with no `__eq__`, equality is identity ✓.
- `is` is `False`: two distinct objects ✓. Identity compares objects; equality compares values once defined.

Exercise 3.1

3.1 A Copy class starts as:

```
class Copy:
    total = 0
    def __init__(self, title):
        self.title = title
        Copy.total += 1
```

Three copies are created; then `x.total = 99` is executed on the first of them. State the value of `Copy.total`, the value of `x.total`, and the value of `.total` on the other two copies, explaining your answers by the lookup order. Explain in one line why the shared counter belongs on the class, not in each instance.

[8]

Exercise 3.1

3.1 A Copy class starts as:

```
class Copy:
    total = 0
    def __init__(self, title):
        self.title = title
        Copy.total += 1
```

Three copies are created; then `x.total = 99` is executed on the first of them. State the value of `Copy.total`, the value of `x.total`, and the value of `.total` on the other two copies, explaining your answers by the lookup order. Explain in one line why the shared counter belongs on the class, not in each instance. [8]

3.1

- `Copy.total` is 3 ✓.
- `x.total` is 99 – the assignment shadows in the instance dict ✓.
- The other two read 3: their instance dicts are empty, lookup falls to the class ✓.
- One counter must agree across all instances – shared state lives where it is shared, on the class ✓.

Exercise 3.2

3.2 This program runs:

```
p = Book('Signals', 3)
q = Book('Signals', 3)
r = p
p.take()
```

Give the value of `.copies` read through each of `p`, `q` and `r`, with a reason for each. State the outcome of `p == q`, of `p is r`, and of `q is p`, naming what equality and identity each compare. [7]

Exercise 3.2

3.2 This program runs:

```
p = Book('Signals', 3)
q = Book('Signals', 3)
r = p
p.take()
```

Give the value of `.copies` read through each of `p`, `q` and `r`, with a reason for each. State the outcome of `p == q`, of `p is r`, and of `q is p`, naming what equality and identity each compare. [7]

3.2

- `p.copies` is 2, `r.copies` is 2 – same object, one take ✓.
- `q.copies` is 3 – a second object, untouched ✓.
- `p == q` is `False` (default equality is identity); `p is r` is `True` (one object); `q is p` is `False` (distinct objects) ✓. Equality compares values, identity compares objects ✓.