Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

Each answer shows the steps, then the **result**.

2.1

`a.n` is 2: both calls changed one object

`b.n` is 2

`True`

2.2

`['sit']`

`tricks` is one list on the class, shared by every dog

fix: set `self.tricks = []` in `__init__`

2.3

`False` and `False`: by default `==` checks identity

with the `__eq__`, `p == q` is `True`; `p is q` stays `False`

2.4

```
def __repr__(self):
return f"Book({self.title!r}, {self.copies})"
```

3.1

(a)

```
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        if amount <= 0:
            raise ValueError("amount must be positive")
        self.balance += amount
        return self.balance

    def withdraw(self, amount):
        if amount <= 0 or amount > self.balance:
            raise ValueError("bad amount")
        self.balance -= amount
        return self.balance
```

`__init__` stores both attributes, with the default; `deposit` checks then adds; `withdraw` checks both conditions before changing anything; both return the balance

(b) `50`, `30`; the third call raises `ValueError`; the balance stays `30`

3.2

(a) ids 1, 2, 3; `Book.next_id` is 4

(b) `d.id` is 4 and `a.next_id` is 9; `Book.next_id` is 5

(c) assigning `a.next_id` makes an instance attribute on `a` that hides the class one for `a` only; `__init__` reads `Book.next_id`

3.3

(a) `['pen']`: the second cart already holds the first cart's item

(b) the default list is made once, when `def` runs; every cart made without an argument shares that one list

(c) `def __init__(self, items=None):` then `self.items = [] if items is None else list(items)`

(d) yes: the bug is silent and only shows when two objects exist, so a cheap automatic check saves long debugging

exercises.lol exercises.com.cn