Mini-Lab: From Properties to a Descriptor

Goal: Start with three plain `@property` validators, then refactor them into a single reusable **descriptor**. Two questions, one progression.

Q1 — 3 properties

Implement `PersonV1` so that:

- `name`: non-empty `str` (after `.strip()`), stores as given.
- `age`: `int` ≥ 0 (treat `bool` as invalid since it's a subclass of `int`).
- `email`: must contain `'@'`; **normalize** with `value.strip().lower()` on set.

> Do this with three `@property` pairs (get/set). No descriptor yet.

Quick checks (will also run from the skeleton):

```
pv1 = PersonV1("Amit", 44, "  AMIT@EXAMPLE.COM ")
assert pv1.name == "Amit"
assert pv1.age == 44
assert pv1.email == "amit@example.com"
```

Q2 — Extract to a descriptor

Create a minimal **data descriptor** `Field` with:

- `__set_name__(self, owner, name)` — remember the attribute name.
- `__get__(self, instance, owner)` — return `self` if `instance is None`; else read `instance.__dict__[name]`.
- `__set__(self, instance, value)` — optional `coerce` then `validator`, then store in `instance.__dict__[name]`.

Then refactor into:

```
class Person:
    name  = Field(validator=ensure_str_non_empty)
    age   = Field(validator=ensure_int_ge0)
    email = Field(coerce=normalize_email, validator=ensure_email)
```

with the same semantics as `PersonV1`.

Quick checks:

```
p = Person("Amit", 44, "  AMIT@EXAMPLE.COM ")
assert p.name == "Amit"
assert p.age == 44
assert p.email == "amit@example.com"
```

Notes / Hints

- Store **per-instance** values in `instance.__dict__`, never on the descriptor object.
- Because `Field` implements `__set__`, it is a **data descriptor** and will take precedence over `instance.__dict__` values on access.
- Keep this minimal—no defaults or delete logic required for this mini-lab.

Run

```
# run skeleton (will fail until you implement)
python lab_skeleton.py

# compare with reference
python lab_solution.py
```

Extra (optional, 5 min)

- Add `__delete__` to the descriptor to clear a stored value.
- Add a `doc` string to `Field` and set `self.__doc__ = doc` so `help(Person)` shows per-field docs.