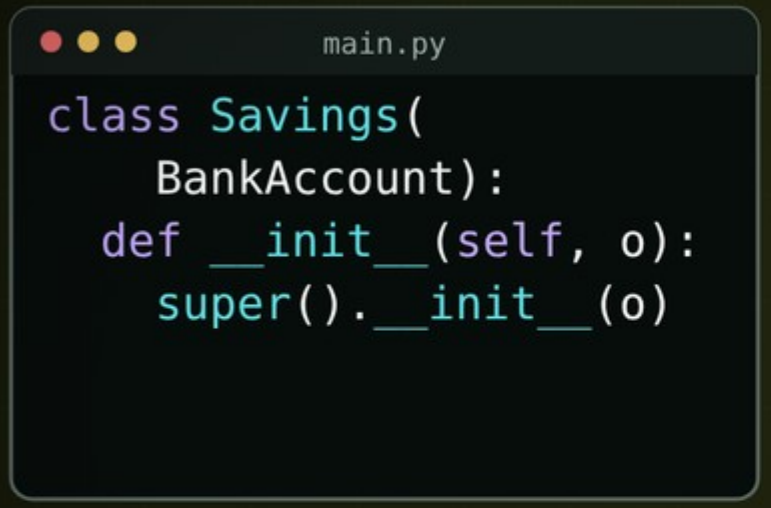
LESSON 9



Py

Encapsulation & Inheritance

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

PYTHON SCRIPTING & OOP

LESSON 9

OOP: Encapsulation and Inheritance

_private names, @property, inheritance, super() and method overriding

Encapsulation by Convention



- Python has no private keyword. Instead, a name starting with an underscore means internal, …
- Two underscores trigger name mangling, which makes accidental access harder. Python trusts …

01

@property: Controlled Access

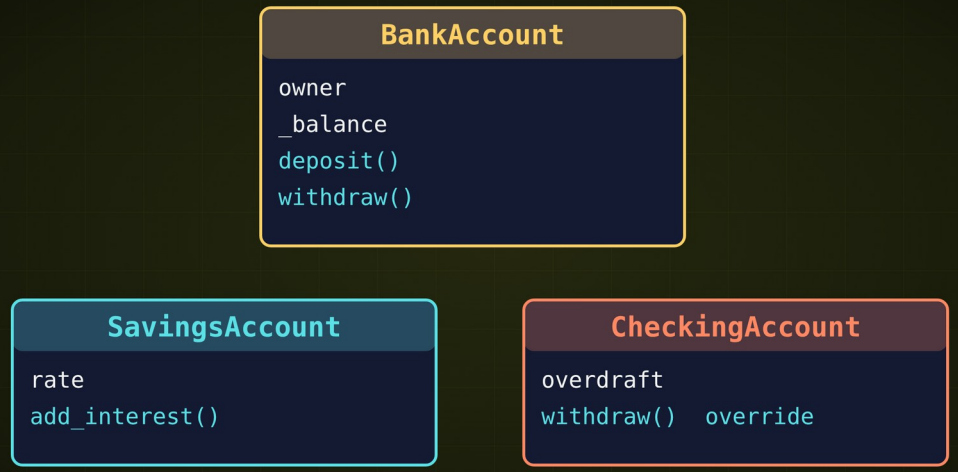
property.py

```python
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self._balance = balance
    @property
    def balance(self):
        return self._balance
    @balance.setter
    def balance(self, value):
        if value < 0: raise ValueError("balance cannot be negative")
        self._balance = value
```

- property lets you read a value like an attribute while a method runs behind the scenes. A …
- This is real data hiding: outside code writes account dot balance, but your rules always …

02

Inheritance



- A child class inherits everything from its parent and adds or changes what it needs. Write …
- A SavingsAccount is a BankAccount, plus interest. You reuse all the parent's code without …

03

super() and Overriding

savings.py

```python
class SavingsAccount(BankAccount):
    def __init__(self, owner, balance=0, rate=0.04):
        super().__init__(owner, balance)
        self.rate = rate
    def add_interest(self):
        self.balance += self.balance * self.rate

class CheckingAccount(BankAccount):
    def withdraw(self, amount):          # override
        ...
```

- super calls the parent's version of a method. In double underscore init, call super first …
- Overriding means the child defines a method with the same name, and Python uses the child's …

04

Inheritance vs Composition

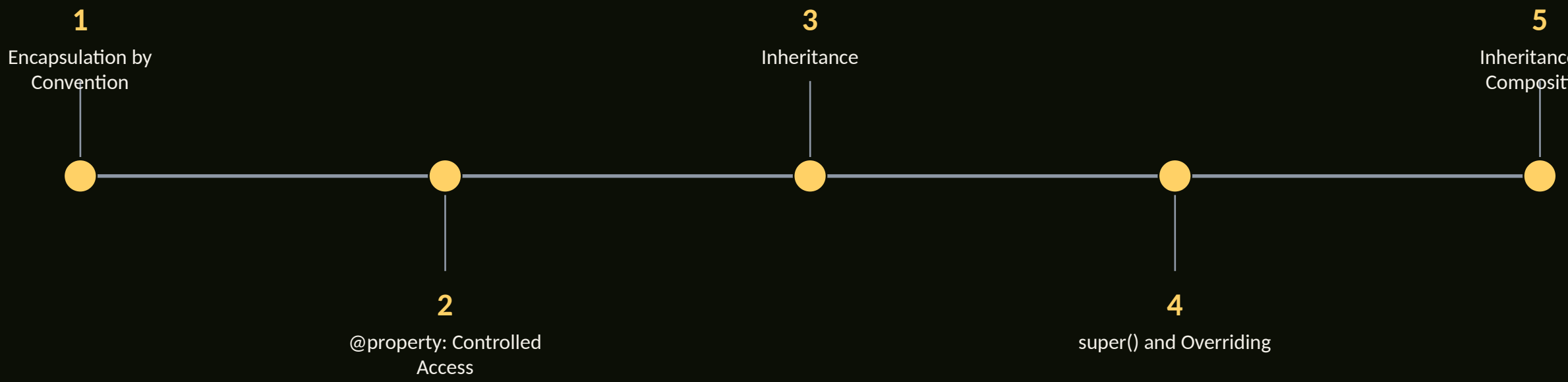
bank.py

```
class Bank:
    def __init__(self, name):
        self.name = name
        self.accounts = []            # has-a: composition
    def open(self, account):
        self.accounts.append(account)
    def total(self):
        return sum(a.balance for a in self.accounts)
```

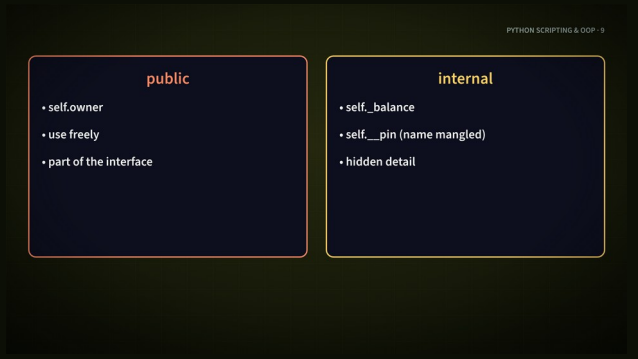
- Use inheritance only for a true is-a relationship. When one object simply uses another, …
- A Bank has many accounts, it is not an account. Composition keeps designs simple and flexible.

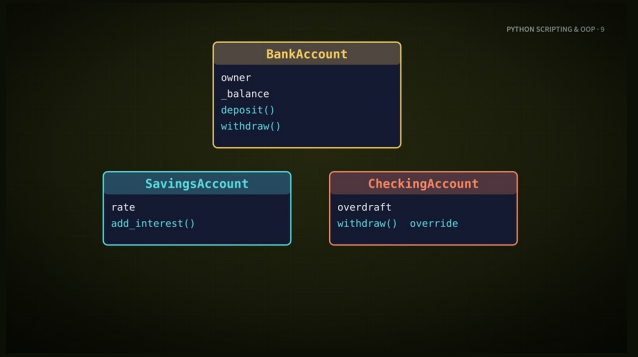
05

Lesson Map


1
Encapsulation by Convention
2
@property: Controlled Access
3
Inheritance
4
super() and Overriding
5
Inheritance vs Composition

Sources and Further Reading





Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari