درس ۹

Py

کپسوله‌سازی و وراثت

```python
class Savings(
    BankAccount):
  def __init__(self, o):
    super().__init__(o)
```

یک کورس رایگان از MyWebUniversity

اسکریپت‌نویسی Python و OOP

درس ۹

OOP: کپسوله‌سازی و وراثت

نام‌های _private، @property، وراثت، super() و بازنویسی method



کپسوله‌سازی با رسم

Python کلمهٔ private ندارد. به جای آن، نامی که با آندرسکور شروع می‌شود یعنی داخلی، از بیرون ...

دو آندرسکور تغییر نام را فعال می‌کند، که دسترسی تصادفی را سخت‌تر می‌سازد. Python به ...



01

@property: دسترسی کنترول‌شده

property اجازه می‌دهد یک مقدار را مثل attribute بخوانید در حالی که یک method پشت صحنه اجرا ...

این پنهان‌سازی واقعی دیتا است: کود بیرونی account.balance می‌نویسد، اما قواعد شما همیشه ...

property.py

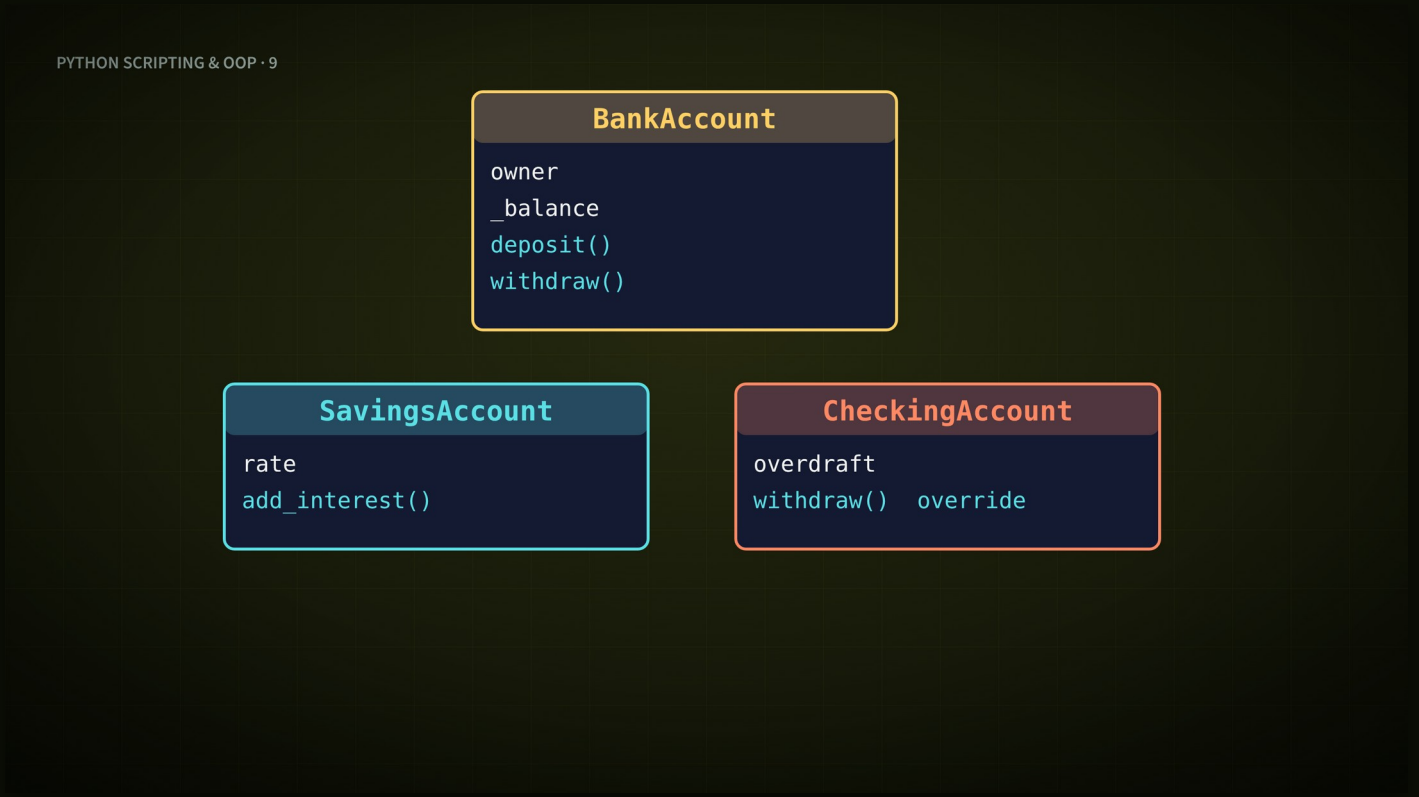
```python
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self._balance = balance
    @property
    def balance(self):
        return self._balance
    @balance.setter
    def balance(self, value):
        if value < 0: raise ValueError("balance cannot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

02

وراثت

یک کلاس فرزند همه چیز را از والد خود به ارث می‌برد و آنچه لازم دارد اضافه یا تغییر می‌دهد. ...

یک SavingsAccount یک BankAccount است، به علاوهٔ سود. همهٔ کود والد را بدون کاپی دوباره ...



03

super() و بازنویسی

super نسخهٔ والد یک method را صدا می‌کند. در __init__، اول super را صدا کنید تا والد بخش ...

بازنویسی یعنی فرزند یک method با همان نام تعریف می‌کند، و Python برای آبجکت‌های فرزند نسخهٔ ...

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
        ...
```

04

وراثت در برابر ترکیب

وراثت را تنها برای یک رابطهٔ واقعی «هست-یک» استفاده کنید. وقتی یک آبجکت فقط از دیگری ...

یک Bank حساب‌های زیادی دارد، خودش حساب نیست. ترکیب طراحی‌ها را ساده و انعطاف‌پذیر نگه می‌دارد.

bank.py

```python
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)
```

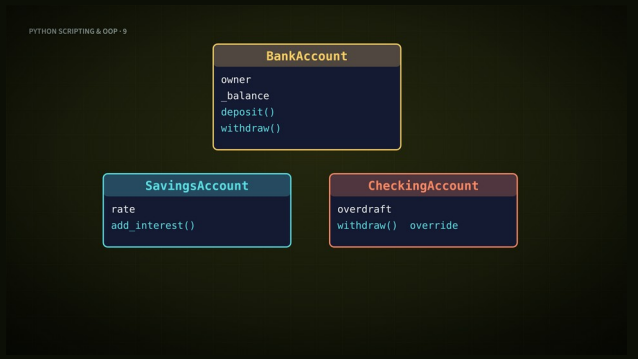
05

نقشهٔ درس


1
کپسوله‌سازی با رسم
2
دسترسی @property: کنترول‌شده
3
وراثت
4
super() و بازنویسی
5
وراثت در برابر ترکیب

منابع و مطالعهٔ بیشتر





تشکر از تماشای شما

MyWebUniversity.com

آموزش رایگان تکنالوژی از سال ۲۰۰۰ — انگلیسی و دری

@MyWebUniversity-Dari · @MyWebUniversity