

Object-oriented programming · supported focus

Supported focus · SL

Supported scope

- Supported SL focus. First assessment 2027 target; official PDF returns 403; older acquired brief is final assessment 2026. Remaining guide, assessment and practical requirements retain their recorded holds.
- State model conditions and distinguish fictional data from measurements.

Oop · 1.1

Reasoning

Two bank-account model objects can respond to the same deposit operation while holding different balances. Start with a prediction. State the quantities or features you would compare, then decide what evidence could distinguish two explanations.

Oop · 2.1

Reasoning

A class defines a type with state and behaviour. An object is an instance .

Encapsulation controls access through an interface so operations can preserve invariants. Inheritance models an appropriate type relationship; composition models an object containing or using another object. encapsulation: Controlling access to state through an interface; instance: An individual object of a class.

Oop · 3.1

Reasoning

A method call acts on a particular instance. State changes should satisfy preconditions and postconditions. Polymorphism lets code use a common interface with different implementations when the contract is respected. Implement a small synthetic account or inventory model. Test two independent instances, rejected invalid operations and boundary values. Keep the model away from real financial accounts and credentials.

Oop · 4.1

Reasoning

State the known values and their units. Choose the relation because its assumptions fit this case, then rearrange before substitution. Known: object A starts at 40 units and receives 15; object B starts at 20 and receives 5. Final balances are 55 and 25. Their total is 80, but neither individual object balance is 80.

Oop · 4.2

Reasoning

An instance starts at 12 units, adds 8 and removes 3. Find its final value. Use the same sequence: known quantities $\rightarrow$ model $\rightarrow$ relation $\rightarrow$ substitution $\rightarrow$ unit and interpretation.

Oop · 5.1

Reasoning

A class is not the same thing as one instance. Inheritance is not automatically better than composition. Merely hiding a field does not prove that all operations preserve valid state. Return to the original observation. Explain what the result supports, which conditions it assumes, and one way to test a competing explanation.

Oop · 5.2

Reasoning

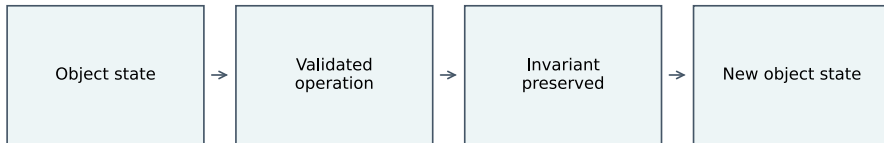
Two instances of a class must always share one balance. This claim is false: A class is not the same thing as one instance. Inheritance is not automatically better than composition. Merely hiding a field does not prove that all operations preserve valid state.

Oop · 5.3

Reasoning

Objects: state, behaviour and an interface: A method call acts on a particular instance. State changes should satisfy preconditions and postconditions. Polymorphism lets code use a common interface with different implementations when the contract is respected.

Oop · diagram



- Original teaching diagram: Objects: state, behaviour and an interface
- Use the labels and the stated model together.

Oop · original transfer

Your turn

Independent accounts start at 10 and 20. Credit 5 to the first, then try to withdraw 30 under a no-overdraft rule. Give final states and the invariant.

Oop · transfer reasoning

Method and interpretation

States are 15 and 20. Rejected withdrawal leaves state unchanged. Instance attributes are independent; validated methods preserve $\text{balance} \geq 0$.

Oop · executable trace

- Trace the state changes and boundary guard.
- Run the boundary case and compare the printed output.

```
class Account:
    def __init__(self, balance):
        self.balance = balance
    def withdraw(self, amount):
        if amount < 0 or amount > self.balance:
            return False
        self.balance -= amount
        return True
a, b = Account(15), Account(20)
print(a.withdraw(30), a.balance, b.balance)
print(a.withdraw(15), a.balance, b.balance)
```

Check yourself

- Explain instance using the stated model.
- Explain encapsulation using the stated model.
- Change one value in the original transfer; explain which assumption still matters.