



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

Introduction to Computation (CS1602) · Lecture 9

Objects and Encapsulation 对象与封装

Instructor: Tao Huang

Part 4: Abstraction and Organization · Fall 2026



Part IV · Abstraction and Organization

Organize related state and operations.

How can classes and modules support a larger program?



| Data, Interfaces and Verification 数据与接口

- **Contract 约定**: inputs and promised results
- **State 状态**: changes retained after a call
- **Evidence 证据**: examples, boundaries and invariants



| Two Position Records 两条位置记录

Record 记录	Coordinates 坐标	Tasks 操作
A	(0, 0)	Show the position; move A
B	(3, 4)	Measure the distance from A to B

| 先区分状态、操作与查询，不预先指定实现。

Quiz 1 · Record Updates



扫码作答 · 1 题

1 普通函数：状态修改与返回值

Submit this round once. 按已学规则分析。

<https://taohuang.info/cs1602/zh/quiz/9/>



| Lecture Structure

- | | | |
|----------|-------------------|---|
| 1 | Objects 对象 | Initialization, attributes and aliases 初始化与引用 |
| 2 | Methods 方法 | Receivers, locals and results 接收者与返回值 |
| 3 | Rules 规则 | Valid states and public interfaces 有效状态与接口 |
| 4 | Sharing 共享 | Class attributes and method kinds 类属性与方法类别 |
-

I · Objects and Initialization

Functions already operate on records.

How can one interface organize many records?

| Records and Operations 记录与操作

Coordinates and related operations

Two dictionary records

a = {"x": 0, "y": 0}

b = {"x": 3, "y": 4}

- **Show** a position
- **Move** one record
- **Measure** a distance

| 类不能代替公式与输入检查。



| Existing Types and Methods 既有类型与方法

code

```
1 print(type(42), type("abc"))
2 text = "hello world"
3 print(text.count("o"))
4 items = [3, 1, 2]
5 print(items.reverse())
6 print(items)
```

output

```
<class 'int'> <class 'str'>
2
None
[2, 1, 3]
```

| Classes and Instances 类与实例

code

```
1 class Point:
2     pass
3
4 p = Point()
5 print(type(p))
6 print(type(p) is Point)
```

output

```
<class '__main__.Point'>
True
```

| Instance Attributes 实例属性

code

```
1 class Point:
2     pass
3
4 x = 100
5 p = Point()
6 p.x = 3
7 p.y = 4
8 print(p.x, p.y, x)
```

output

```
3 4 100
```

| Missing Attributes 缺失属性

Predict both accesses before advancing. 先预测两次访问。

```
1  class Point:
2      pass
3
4  p = Point()
5  p.x = 3
6  q = Point()
7  print(p.x)
8  print(q.x)
```

Initialization with `__init__` 初始化

code

```
1 class Point:
2     def __init__(self, x: float = 0, y: float = 0) -> None:
3         self.x = x
4         self.y = y
5
6 p = Point(3, 4)
7 print(p.x, p.y)
```

output

3 4

I Default Coordinates 默认参数

code

```
1 class Point:
2     def __init__(self, x: float = 0, y: float = 0) -> None:
3         self.x = x
4         self.y = y
5
6 for p in [Point(), Point(3), Point(3, 4)]:
7     print(p.x, p.y)
```

output

```
0 0
3 0
3 4
```

| Class Body and Initialization 类体与初始化

How often does each message appear? 各打印几次?

```
1  class Trace:
2      print("class body")
3      def __init__(self) -> None:
4          print("init")
5
6  a = Trace()
7  b = Trace()
```

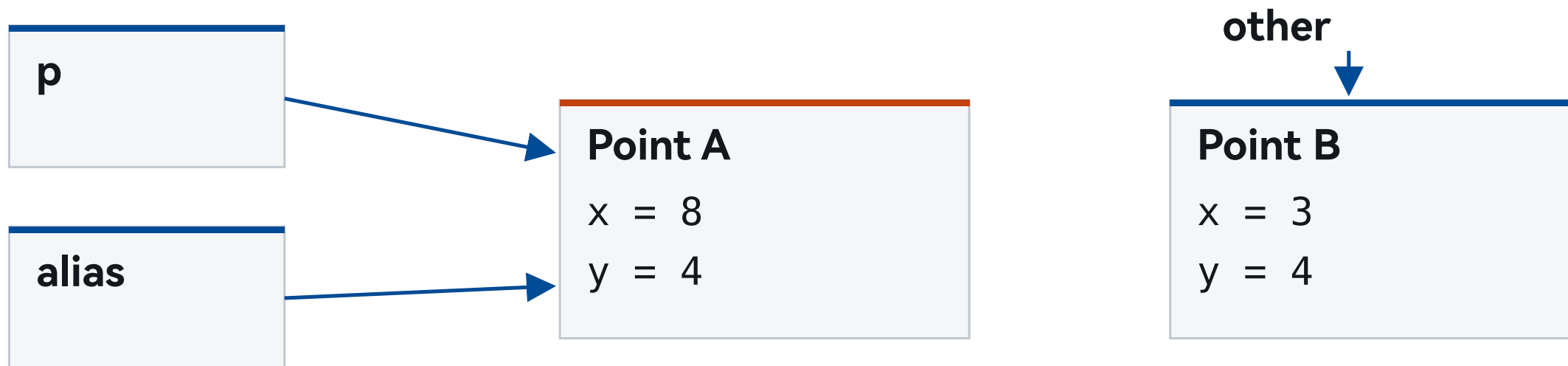
| Class Execution and Initialization Results

Actual output 实跑输出

```
class body  
init  
init
```

- **Class body 类体**: once in this program
- **Initialization 初始化**: once per construction
- **__init__ returns None**; the class call returns the instance

| Aliases and Separate Instances 别名与实例



Names → objects 名字与对象; logical references, not physical addresses.

| Initialization Contract 初始化约定

- Required attributes 必需属性
- Accepted values 输入范围
- Defaults 默认值
- Failure behavior 失败行为

| 属性存在不等于数值合法。

II • Instance Methods

Instances now have initialized state.

How does shared method code choose its receiver?

| Method Definitions 方法定义

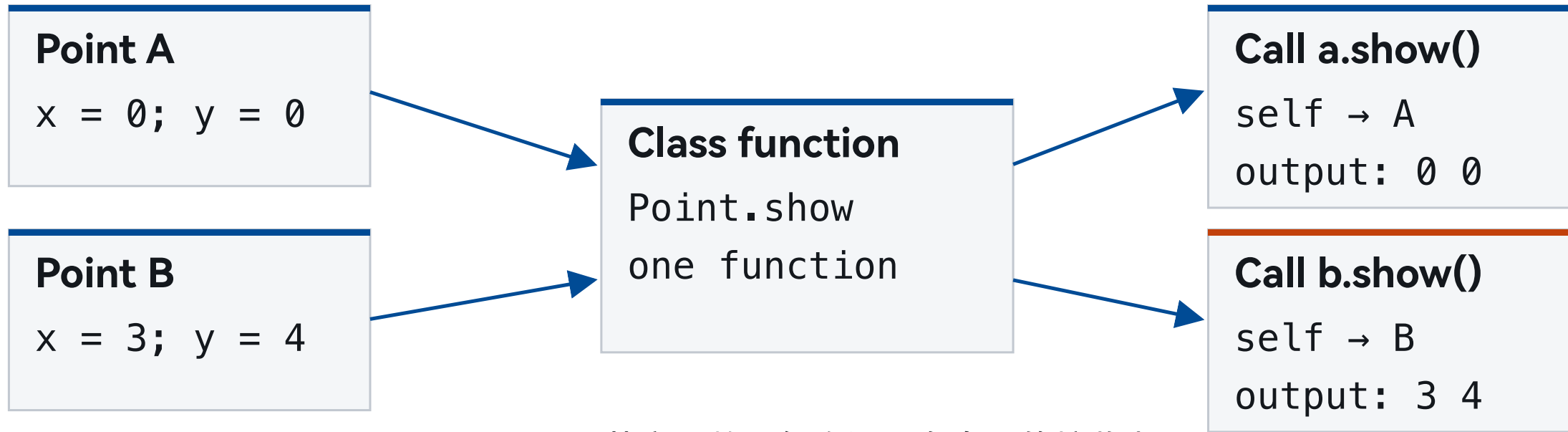
code

```
1 class Point:
2     def __init__(self, x: float = 0, y: float = 0) -> None:
3         self.x = x
4         self.y = y
5
6     def show(self) -> None:
7         print(self.x, self.y)
8
9 p = Point(3, 4)
10 p.show()
```

output

3 4

Receiver Binding 接收者绑定



Shared function; separate call frames 共享函数，每次调用有自己的接收者。

| A Missing Receiver Parameter

What does `p.show()` supply? 调用传入什么?

```
1 class Point:
2     def show():
3         print("hi")
4
5 p = Point()
6 p.show()
```



| Local Parameters and Attributes 局部与属性

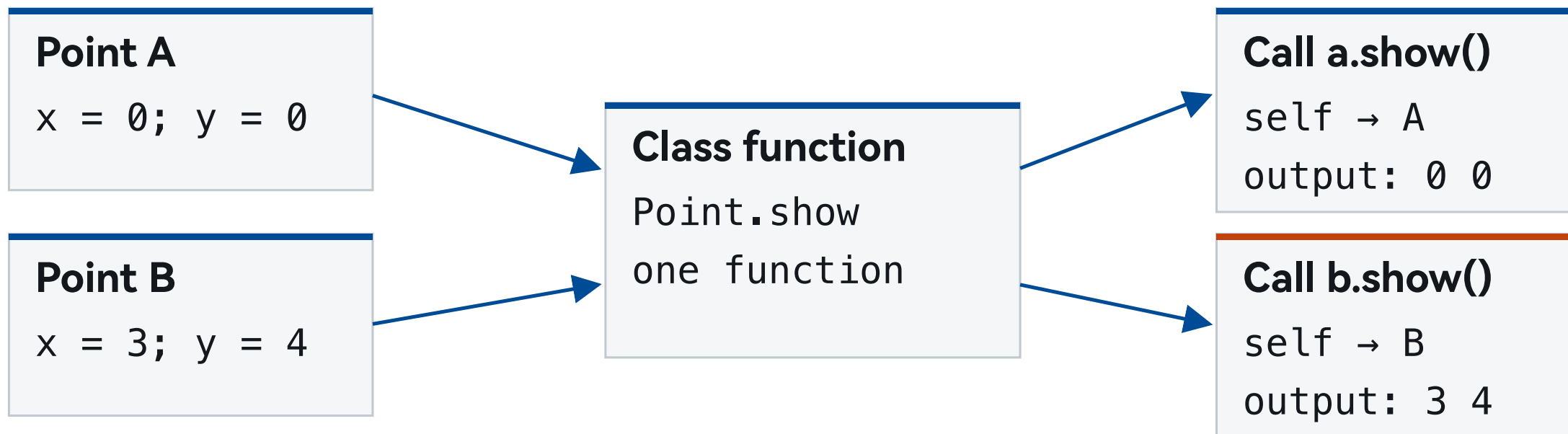
code

```
1 class Point:
2     def __init__(self, x: int) -> None:
3         self.x = x
4     def shift(self, x: int) -> None:
5         self.x += x
6
7 x = 100
8 p = Point(3)
9 p.shift(2)
10 print(p.x, x)
```

output

5 100

| One Method, Two Receivers 一个方法与两个接收者



Shared function; separate call frames 共享函数，每次调用有自己的接收者。

| Point Movement 点的平移

code

```
1  class Point:
2      def __init__(self, x: float = 0, y: float = 0) -> None:
3          self.x = x
4          self.y = y
5
6      def move(self, dx: float, dy: float) -> None:
7          self.x += dx
8          self.y += dy
9
10 a = Point(0, 0)
11 b = Point(3, 4)
12 a.move(1, 1)
13 print(a.x, a.y)
14 print(b.x, b.y)
```

output

```
1 1
3 4
```

| State Changes and Return Values 状态与返回值

code

```
1 def move(record, dx, dy):  
2     record["x"] += dx  
3     record["y"] += dy  
4  
5 p = {"x": 3, "y": 4}  
6 result = move(p, 1, 0)  
7 print(p["x"], result)
```

output

4 None

Calls Within a Method 方法内调用

code

```
1 def message() -> str:
2     return "global function"
3
4 class Record:
5     def __init__(self, text: str) -> None:
6         self.text = text
7     def message(self) -> str:
8         return self.text
9     def report(self) -> str:
10        return self.message()
11
12 r = Record("instance method")
13 print(message())
14 print(r.report())
```

output

```
global function
instance method
```

| Forward Type References 前向类型引用

- `def distance_to(self, other: "Point") -> float:`
- **The class name is bound after its body finishes.**
- **The string annotation defers the reference.**

| 先解释引号的作用，再使用引用当前类的类型提示。

| Distance Between Objects 对象间距离

code

```
1 class Point:
2     def __init__(self, x: float = 0, y: float = 0) -> None:
3         self.x = x
4         self.y = y
5
6     def distance_to(self, other: "Point") -> float:
7         dx = self.x - other.x
8         dy = self.y - other.y
9         return (dx * dx + dy * dy) ** 0.5
10
11 a = Point(0, 0)
12 b = Point(3, 4)
13 print(a.distance_to(b))
```

output

5.0

Quiz 2 · Initialization and Receivers



扫码作答 · 2 题

1 局部参数与实例属性 2 显式选择方法接收者

Submit this round once. 按已学规则分析。

<https://taohuang.info/cs1602/zh/quiz/9/>

Quiz Review · Attribute Creation

code

```
1 class Label:
2     def __init__(self, text: str) -> None:
3         text = text.strip()
4
5 a = Label(" hi ")
6 print(a.text)
```

Actual exception · final line 实际异常

AttributeError:
'Label' object has no attribute 'text'

Quiz Review · Receiver Selection

code

```
1 class Counter:
2     def __init__(self, start: int = 0) -> None:
3         self.value = start
4     def add(self, delta: int) -> None:
5         self.value += delta
6
7 a = Counter(1)
8 b = Counter(10)
9 Counter.add(b, 2)
10 print(a.value, b.value)
```

output

1 12

| The Point Interface Point 接口

Operation 操作	State effects 状态	Result / output
Point(x, y)	Initialize new coordinates	Class call → instance; init → None
show()	Coordinates unchanged	Printed coordinates; None
move(dx, dy)	Change this instance	None
distance_to(other)	Both points unchanged	Distance

| Instance-Method Reasoning 实例方法分析

- | | | |
|---|------------------------|--|
| 1 | Receiver 接收者 | Identify the object supplied as self. |
| 2 | Bindings 绑定 | Separate local arguments from attributes. |
| 3 | Calls 调用 | Follow ordinary call and return order. |
| 4 | Observations 观察 | Check state, returned value and output separately. |

| 同一个方法的两次调用，先确定各自的接收者和原状态。



HALFWAY

Break

Next: valid states, shared data and class methods.

III · Encapsulation and State Rules

Methods organize state changes.

Which rules must every public operation preserve?



| The Account Contract 账户约定

Item 项目	Accepted behavior 约定
State 状态	Nonnegative balance; integer cents
deposit(amount)	Positive integer; returns None
withdraw(amount)	Positive integer; enough funds; returns bool
Rejected calls 拒绝	Balance and history stay unchanged
statement()	Return a string; do not change state

| 先确定单位、输入与失败行为，再审查更新语句。

| A Withdrawal Candidate 取款候选

Predict `withdraw(-20)`. 候选, 不是正确示范。

```
1 class CandidateAccount:
2     def __init__(self, balance: int) -> None:
3         self.balance = balance
4     def withdraw(self, amount: int) -> bool:
5         if amount > self.balance:
6             return False
7         self.balance -= amount
8         return True
9
10 a = CandidateAccount(100)
11 print(a.withdraw(-20), a.balance)
```

| Withdrawal Validation 取款检查

code

```
1 class BankAccount:
2     def __init__(self, balance: int) -> None:
3         if balance < 0:
4             raise ValueError("negative opening balance")
5         self.balance = balance
6     def withdraw(self, amount: int) -> bool:
7         if amount <= 0 or amount > self.balance:
8             return False
9         self.balance -= amount
10        return True
11
12 a = BankAccount(100)
13 print(a.withdraw(-20), a.balance)
```

output

False 100

I Rules at Every Entry Point 入口与规则

Entry 入口	Validate 检查	On rejection 拒绝后
Constructor 构造	Nonnegative opening balance	ValueError; no valid instance returned
Deposit 存款	Positive integer amount	Balance/history unchanged
Withdraw 取款	Valid amount and enough funds	False; balance/history unchanged

| Account State Transitions 账户状态变化

Actual output 实跑输出

```
True 120  
False 120  
False 120  
0 []  
[('deposit', 50), ('withdraw', 30)]
```

- **100 → +50 → -30 → 120**
- **Rejected 1000 and -20** leave state unchanged
- **Separate account 独立账户**: zero balance, empty history
- **History** records accepted actions only

| The Temperature Contract 温度约定

Check 检查

Rule 规则

Domain 输入域

Finite real values; celsius ≥ -273.15

Initialization 初值

Invalid starting value $\rightarrow$ ValueError

set_celsius(value)

False and unchanged on rejection; True on acceptance

math.isfinite(value)

True for finite values, False for NaN/infinity

| 非法初值也要拒绝，不能只检查后来的 setter。

I Temperature Updates 温度更新

Excerpt · complete definition in notes

```
def set_celsius(  
    self,  
    value: float,  
) -> bool:  
    if (  
        not math.isfinite(value)  
        or value < -273.15  
    ):  
        return False  
    self._celsius = value  
    return True
```

- **Starting state:** 25 Celsius
- **First call:** set_celsius(-300)
- **Second call:**
set_celsius(100)
- **Predict each result and retained value.**

[Complete runnable example](#) 完整程序

Public and Nonpublic Interfaces 公开与非公开接口

Previous prediction · actual result

False 25

True 212.0

- **Public interface 公开接口**: supported operations and rules
- **_name 非公开约定**: implementation detail
- **Direct assignment remains possible**; no enforced privacy

下划线不提供权限控制。

| Name Mangling 名称改写

code

```
1 class Secret:
2     def __init__(self) -> None:
3         self.__hidden = 42
4
5 s = Secret()
6 print(s._Secret__hidden)
7 print(s.__hidden)
```

Actual exception · final line 实际异常

AttributeError:
'Secret' object has no attribute '__hidden'



| Underscore Naming 下划线命名

Form 形式	Meaning 含义	Access control?
<code>_x</code>	Nonpublic convention	No
<code>__x</code>	Name mangling in a class body	No
<code>__init__</code>	A documented special-method name	Not the <code>__x</code> mangling rule

Quiz 3 · Initial-State Rules



扫码作答 · 1 题

1 构造入口与有效状态

Submit this round once. 按已学规则分析。

<https://taohuang.info/cs1602/zh/quiz/9/>

IV · Shared Data and Method Kinds

An instance has its own attribute bindings.

Which objects and operations should the class share?

| Instance and Class Attributes 实例与类属性

code

```
1 class Point:
2     dimensions = 2
3     def __init__(self, x: float, y: float) -> None:
4         self.x = x
5         self.y = y
6
7 a = Point(0, 0)
8 b = Point(3, 4)
9 print(a.x, b.x)
10 print(a.dimensions, b.dimensions, Point.dimensions)
```

output

```
0 3
2 2 2
```

I Ordinary Attribute Lookup 普通属性查找

Predict a.label, b.label and Record.label. 先确定绑定位置。

```
1 class Record:
2     label = "class"
3
4 a = Record()
5 b = Record()
6 a.label = "only A"
7 print(a.label, b.label, Record.label)
```

| Attribute Binding and Mutation 绑定与修改

Previous prediction · actual result

only A class class

Operation 操作

Effect 影响

a.label = ...

Establish an ordinary instance binding

Record.label = ...

Change the class binding

a.items.append(...)

Mutate the referenced list

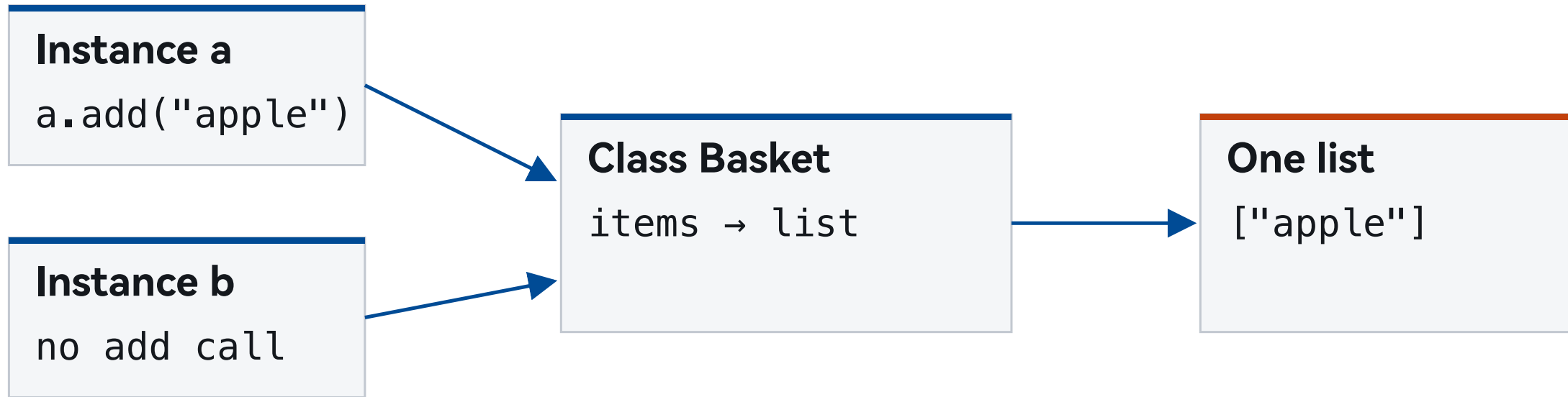
| 区分重新绑定与修改对象。

| A Shared Basket List 共享列表

Which list does add() modify? 修改的是哪份列表?

```
1  class Basket:
2      items: list[str] = []
3      def add(self, item: str) -> None:
4          self.items.append(item)
5
6  a = Basket()
7  b = Basket()
8  a.add("apple")
9  print(b.items)
10 print(a.items is b.items)
```

Shared-List References 共享列表引用



Bind a name vs mutate its object 绑定位置和被修改的对象须分别核对。

| Per-Instance Collections 实例集合

code

```
1 class Basket:
2     def __init__(self) -> None:
3         self.items: list[str] = []
4     def add(self, item: str) -> None:
5         self.items.append(item)
6
7 a = Basket()
8 b = Basket()
9 a.add("apple")
10 print(a.items, b.items)
11 print(a.items is b.items)
```

output

```
['apple'] []
False
```

I Shared Input References 输入引用

Does initialization copy an input list? 初始化会自动复制吗?

```
1 class Bucket:
2     def __init__(self, items: list[str]) -> None:
3         self.items = items
4
5 shared = []
6 a = Bucket(shared)
7 b = Bucket(shared)
8 a.items.append("apple")
9 print(b.items)
10 print(a.items is b.items)
```

| An Instance-Creation Counter 创建计数

code

```
1  class Student:
2      count = 0
3      def __init__(self, name: str) -> None:
4          self.name = name
5          Student.count += 1
6
7
8  Student("A")
9  Student("B")
10 print(Student.count)
```

output

2

| Class Methods and cls 类方法

- **@classmethod** marks a method receiving the class
- **cls** is the conventional first parameter
- **No instance required** for access through the class

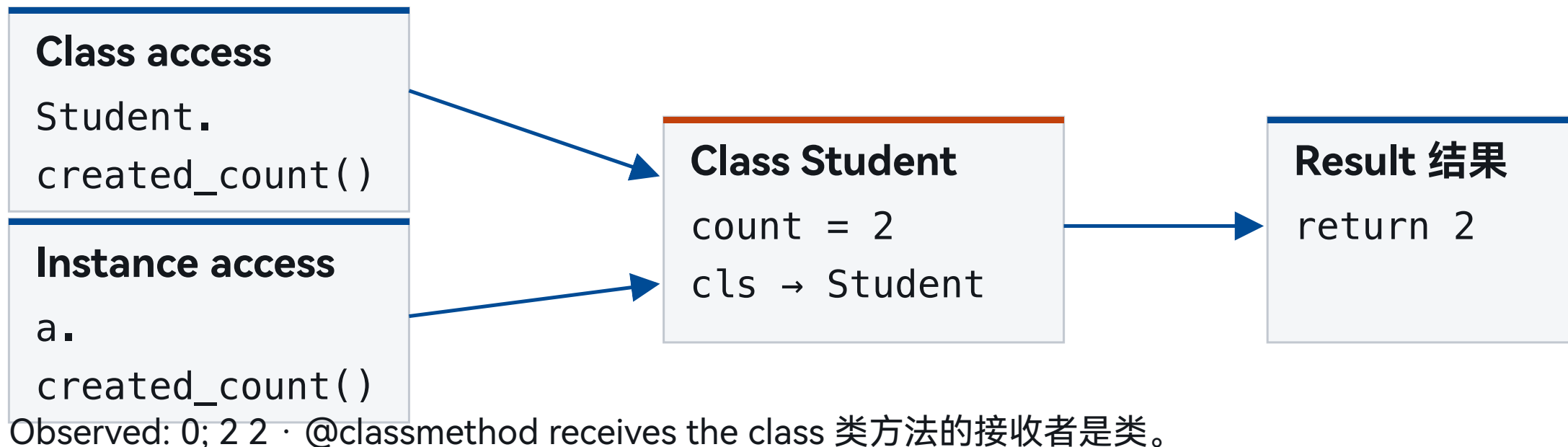
| 确定自动参数传的是实例还是类。

Calling a Class Method 类方法调用

Which receiver and results? 接收者与结果分别是什么?

```
1  class Student:
2      count = 0
3      def __init__(self, name: str) -> None:
4          self.name = name
5          Student.count += 1
6
7      @classmethod
8      def created_count(cls) -> int:
9          return cls.count
10
11  print(Student.created_count())
12  a = Student("A")
13  b = Student("B")
14  print(Student.created_count(), a.created_count())
```

| Method Categories 方法类别



Kind 类别	Implicit receiver 自动参数	Typical purpose
Instance method	Instance (self)	Per-instance operations
@classmethod	Class (cls)	Class-level operations
@staticmethod	None	Related operation; no receiver state

Quiz 4 · Sharing and Class Methods



扫码作答 · 2 题

1 列表修改与实例重新绑定 2 类方法的自动接收者

Submit this round once. 按已学规则分析。

<https://taohuang.info/cs1602/zh/quiz/9/>

Quiz Review · Sharing and Binding

Q5 · actual list lengths

0 1 1

Q6 · actual class count

2

- **Q5**: mutate the class list, then bind a new instance list
- **Q6**: both accesses supply Sensor as cls

创建实例不自动调用任意方法；给实例新绑定也不替换类上的列表。

| Design Choices 类的设计选择

Consider 考虑因素

Reasoning 判断依据

State lifetime 状态持续

Must data be retained across calls?

Related operations 相关操作

Does one interface make them clearer?

Rules and failures 规则与失败

Which updates and invariants need a contract?

Alternatives 其他组织

A function, dictionary or record may suffice

| 按状态与接口选择组织方式。



| Lab 9 · Design and Verification

Part 实验段	Tasks 任务	Raw points
A	Rectangle / Stack / WordCounter / Temperature	16 auto + 8 staff = 24
B	Gradebook refactoring and regression	8 auto + 4 staff = 12
C	Class/function choices; receiver debugging	0-point learning record

| HW3 due: **17 Nov, 23:59 GMT+8**. 接口和入口以实验页为准。



| Inheritance, Protocols and Modules

- A class organizes one kind of state and behavior.
- How can related classes reuse implementation?
- How can classes and functions be organized into files?

| 参考区供课后查阅。

Reference 参考

Complete programs, checks and legacy examples.

After class; not an additional lecture.

Built-in Constructors 内置构造器

code

```
1 classes = [int, float, complex, str,  
2           list, tuple, dict, set]  
3 for cls in classes:  
4     value = cls()  
5     print(cls.__name__, repr(value))
```

output

```
int 0  
float 0.0  
complex 0j  
str ''  
list []  
tuple ()  
dict {}  
set set()
```

| Class Execution and Object Identity

Actual output 实际输出

True

False True

- **Class statements execute when reached.**
- **Instance creation differs from alias assignment.**
- **Identity 身份** is separate from coordinate values.

| Point · Complete Program

Excerpt · complete definition in notes

```
def distance_to(  
    self,  
    other: "Point",  
) -> float:  
    dx = self.x - other.x  
    dy = self.y - other.y  
    return (dx * dx + dy * dy) ** 0.5
```

Actual complete-program output 实跑输出

```
5.0  
1 1  
3.605551275463989
```

- **Constructor defaults** and required x/y
- **show / move / distance_to**
- **Check state and result separately**

[Complete runnable example 完整程序](#)

| Three Sources of Shared Collections

Source 来源	What is shared? 共享内容	Check 核对
Class list	One class-level object	Class access and instance shadows
Mutable default	One default object created at definition	Calls omitting that argument
Repeated input	The same caller-owned object	Aliased constructor arguments

| 构造表达式与传入引用须分别核对。



| Triangle · Interface and Validation

Interface 接口

Contract 约定

Defaults

Edges (1, 1, 1), not (0, 0, 0)

is_valid()

Finite positive edges; strict inequalities

set_edges(a, b, c)

False on invalid update; retain old edges

area / print / print_area

Check current data; use self.area()

| Triangle · Heron's Formula

Excerpt · complete definition in notes

```
def area(self) -> float:
    if not self.is_valid():
        raise ValueError("invalid current edges")
    s = (self.a + self.b + self.c) / 2
    product = s * (s - self.a)
    product *= s - self.b
    product *= s - self.c
    return product ** 0.5
```

Actual complete-
program output 实跑
输出

Area: 6.000

- $s = (a+b+c)/2$
- Reject invalid current edges
- 3,4,5 → area 6

[Complete runnable example](#) 完整程序

| Triangle · Independent Instances

Actual output 实跑输出

```
6.0  
3.897114317029974  
False 6.0
```

- **Separate edge data** 独立边数据
- **Rejected updates** keep all old values
- **Same method implementation** selects its receiver

| Dog · Records and Updates

Excerpt · complete definition in notes

```
def print_dog(self) -> None:
    print(
        self.name, self.color,
        self.age, self.weight,
    )
```

Actual complete-program
output 实跑输出

```
Fox Red 3 4
Tiger Red 4 2.0
```

- **name / color / age / weight**
- **External functions** can also operate on objects
- **Setter names do not imply validation**

[Complete runnable example 完整程序](#)

I Happy Numbers · Steps and Cycles

Actual output 实跑输出

19 82 True

2 4 False

1234567890987654321 570 False

- **Positive integer input 正整数**
- **Digit-square sum** defines the next value
- **Stop at 1 or a repeated value**
- **Ordered trace**, not set display order

I Happy Numbers · Query Side Effects

Version 版本	Actual result 实际结果	Retained input?
Legacy candidate	True 100	No: number is changed
Local working state	True 19	Yes: number stays 19

- **current** holds this query's working value
- **seen** holds processed values
- **A query name does not guarantee no mutation**

| Vector · Storage and Length

Excerpt · complete definition in notes

```
def length(self) -> float:
    total = 0.0
    for value in self.values:
        total += value * value
    return math.sqrt(total)
```

Actual complete-program output 实跑输出

Length: 5.000

- **Scalar list**, copied at construction
- **length / print_length**
- **Zero vector** has length 0

[Complete runnable example](#) 完整程序

| Vector · Dot Product and Scaling

Excerpt · complete definition in notes

```
def dot(self, other: "Vector") -> float:
    if len(self.values) != len(other.values):
        raise ValueError("different dimensions")
    total = 0.0
    for i in range(len(self.values)):
        total += self.values[i] * other.values[i]
    return total
```

Actual complete-program output 实跑输出

```
11.0
None
10.0 [3.0, 4.0]
```

- **Same dimension** required for dot
- **scale mutates** and returns None
- **Caller's outer list** is retained

[Complete runnable example 完整程序](#)

| Vector · Direction Angles 方向角

Excerpt · complete definition in notes

```
def angle(self) -> list[float]:  
    length = self.length()  
    if length == 0:  
        raise ValueError("zero vector")  
    angles = []  
    for value in self.values:  
        direction = value / length  
        cosine = max(-1.0, min(1.0, direction))  
        angles.append(math.acos(cosine))  
    return angles
```

Actual complete-
program output 实跑输出

Angle: 0.9273 rad
Angle: 0.6435 rad

- **Angles to coordinate axes**, in radians
- **acos domain**: -1 to 1
- **Zero vector** has no direction

[Complete runnable example 完整程序](#)

| Namespaces and Function Objects 名称与函数对象

Excerpt · complete definition in notes

```
def area(self) -> float:
    return area(self.a, self.b, self.c)

f = print
f("function object")
```

Actual complete-program
output 实跑输出

```
6.0 6.0
function object
```

- **External area** and **instance area** have explicit paths
- **Functions are objects** that names can reference
- **Ordinary functions already isolate locals**

[Complete runnable example 完整程序](#)

| Bound-Method Metadata 绑定方法元数据

Excerpt · complete definition in notes

```
print(a.show.__self__ is a)
print(a.show.__func__ is b.show.__func__)
f = a.show
a.x = 8
f()
```

Actual complete-program output 实跑输出

```
True
True
8 0
```

- **`__self__`**: bound receiver
- **`__func__`**: underlying Python function
- **Saved method** observes later instance state

[Complete runnable example 完整程序](#)

| Static and Unbound Calls 静态与显式调用

code

```
1 class Test:
2     @staticmethod
3     def add(u: int, v: int) -> int:
4         return u + v
5
6 print(Test.add(1, 1))
7 print(Test().add(3, 4))
```

output

```
2
7
```

| Annotations and Initialization Failures

code

```
1 class Wrong:
2     def __init__(self):
3         return 1
4
5 Wrong()
```

Actual exception · final line 实际异常

```
TypeError:
__init__() should return None, not 'int'
```



| Object-Oriented Organization and Alternatives

- **Simula (1960s)** and **Smalltalk (1970s)**: early influential mechanisms
- **Python supports objects and procedural organization.**
- **C++, Java and Python** are familiar related languages
- **Functions, dictionaries and records** remain valid alternatives

| 按状态、接口与生命周期选择，不用范式名称代替设计依据。

I Independent Verification 独立核验

Dimension 检查维度

Evidence 证据

Construction 构造

Defaults, attributes, valid and invalid inputs

Sharing 共享

Two instances, aliases and shared arguments

Effects 副作用

Results, output and retained state

Failure 拒绝

All old fields unchanged, not a partial update

| Lab 9 · Interfaces and Edge Cases

Class 类	Fixed contract 固定约定
Rectangle	Finite positive dimensions; in-place scale → None
Stack	Empty pop and peek → IndexError
WordCounter	Exact keys; count descending, word ascending
Temperature	Invalid start → ValueError; rejected update → False
Gradebook	Empty top → None; missing average → KeyError; fixed ties

| 不要把另一页示例的方法名或返回行为直接当作本题接口。



| Scope of L09 and L10 本讲与下讲范围

Lecture 范围

Topics 知识

L09 · this lecture

Objects, initialization, method binding, interfaces, sharing, method kinds

L10 · next lecture

Inheritance, protocols, polymorphism, multiple inheritance, modules and execution