

Classes II and Modules

Every slide from the lecture, with what was said over it. The slides are rendered natively — the text is selectable, searchable, and prints cleanly.



Introduction to Computation (CS1602) · Lecture 10

Inheritance, Protocols and Modules

Instructor: Tao Huang

Part 4: Abstraction and Organization · Fall 2026

1

Slide 1 · Inheritance, Protocols and Modules

L09已经解释单个类的状态、接收者、共享与公开接口。本讲研究相关类怎样复用实现，以及怎样把定义组织到文件中。

分两个45分钟，参考区保留较长机制与旧例，不把全部96页都计入课堂必讲。

I Objects, Receivers and Contracts 对象与接口



- **Receiver 接收者**: each call selects an object
- **State and result 状态与结果**: observe both
- **Contract 约定**: supported operations and failures

2

Slide 2 · Objects, Receivers and Contracts 对象与接口

复习必要前提：方法选接收者，返回值和状态变化分开，数据与操作需要约定。继承不会免除这些规则。

不重复上一讲完整定义或全量练习，下面的问题由已有工具自然引出。



I Reusable Reports 可复用的说明

Records 记录	Common 共用	Different 差异
Ordinary members	Name and report interface	Role-specific description
Special roles	The same supported operation	Additional state or behavior

先不看代码。几种记录都要生成说明，共同部分可以复用，不同角色仍有不同信息。

现实分类不是正确继承设计的证明；程序中还要检查公开操作的行为兼容。

I Quiz 1 · A Saved Method



扫码作答 · 1 题

1 保存方法引用后的状态观察

Submit this round once. 按已教规则推理。

<https://taohuang.info/cs1602/zh/quiz/10/>

4

Slide 4 · Quiz 1 · A Saved Method

开场只考已经教过的绑定方法、属性更新和两个实例。先预测保存的方法读哪一份状态，不使用继承或super。

暂不揭晓，第11页回顾原分布，不重复提交。

I Lecture Structure



- | | | |
|---|----------------|---|
| 1 | Reuse 复用 | Inheritance and behavioral interfaces |
| 2 | Protocols 协议 | Representations, operators and containers |
| 3 | Modules 模块 | Files, bindings and entry points |
| 4 | Environment 环境 | Lookup, installation and execution |

5

Slide 5 · Lecture Structure

上半节从继承到操作协议，下半节从模块到环境和执行。中间课间是自然分界。

没有重复的独立AI政策页；实验保留实际使用声明。

I · Inheritance and Behavioral Interfaces

基类与子类建立复用关系，但调用者关心的是受支持的行为。先看一个短继承例子，再检查初始化与方法内调用。

继承不是复制父实例，也不会自动检查业务规则。



I A Base Class and Its Subclass 基类与子类

Excerpt · complete definition in notes

```
class Animal:
    def __init__(self, name: str) -> None:
        self.name = name

class Dog(Animal):
    def speak(self) -> str:
        return "woof"
```

- **Base class 基类:** Animal
- **Subclass 子类:** Dog
- Inherited initialization uses this instance

Complete runnable example 完整程序

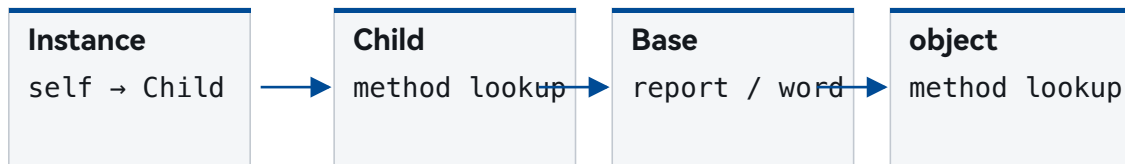
7

Slide 7 · A Base Class and Its Subclass 基类与子类

Animal建立name, Dog没有重写init, 因此使用继承的初始化。Dog重写自己的行为。

本页是结构片段, 完整独立Animal/Dog/Cat在讲义第一节, 不借用其他页的执行状态。

I Inherited Initialization 继承的初始化



Ordinary method lookup 方法查找; self stays the actual instance.

父初始化建立的仍是当前子类实例的字段。类关系查找最终到object，但object不自动支持所有操作。

图是逻辑方法查找与接收者关系，不是物理内存复制；完整C3在参考区。

I Method Overriding 方法重写



Which implementation does the receiver select? 先预测。

```
1 class Base:
2     def word(self):
3         return "base"
4     def report(self):
5         return self.word()
6
7 class Child(Base):
8     def word(self):
9         return "child"
10
11 print(Child().report())
```

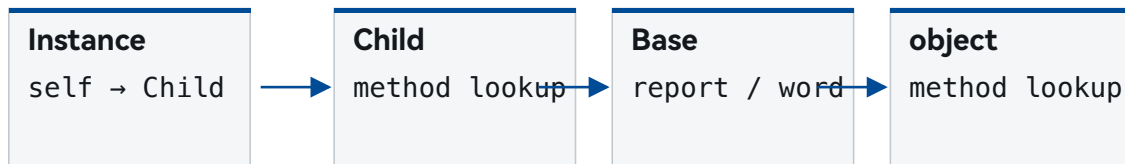
9

Slide 9 · Method Overriding 方法重写

先预测Child的调用最终选哪个word。report虽写在Base中，self仍是当前对象。

本页不显示结果，下一页解释查找，第12页进一步区分父函数来源和实际接收者。

| Method Lookup 方法查找



Ordinary method lookup 方法查找; self stays the actual instance.

10

Slide 10 · Method Lookup 方法查找

实际word结果为child。子类重写的方法通过实际类型查找，不因为report在基类就固定所有后续调用。

普通单继承简图不能推广成所有属性优先级，描述符等例外在参考说明。



| The Receiver in Reused Code 复用代码的接收者

code

```
1 class Point:
2     def __init__(self, x):
3         self.x = x
4     def read(self):
5         return self.x
6
7 a = Point(3)
8 b = Point(4)
9 f = a.read
10 a.x = 8
11 print(f(), b.read())
```

output

8 4

11

Slide 11 · The Receiver in Reused Code 复用代码的接收者

回收开场题，实际8 4。f绑定a，没有保存字段旧值；a更新后仍被同一接收者观察，b保持4。

查看原分布不再次提交。继承方法同样不会把self换成一个新基类对象。

| Calls Inside an Inherited Method 继承方法内调用



Does Base.report force Base.word?

```
1 class Base:
2     def word(self):
3         return "base"
4     def report(self):
5         return self.word()
6
7 class Child(Base):
8     def word(self):
9         return "child"
10
11 print(Child().report())
```

12

Slide 12 · Calls Inside an Inherited Method 继承方法内调用

父函数report内通过self.word访问，先口头解释接收者和查找。不能说进入父函数就将所有调用固定父类。

第15页的实际复用总结继续回顾child结果。

| An Overridden Initializer 重写的初始化



Which required field is missing? 缺哪个字段?

```
1 class Base:
2     def __init__(self, value):
3         self.value = value
4
5 class Child(Base):
6     def __init__(self, value):
7         self.extra = value
8
9 print(Child(3).value)
```

13

Slide 13 · An Overridden Initializer 重写的初始化

子类只建立extra。预测value是否已经由继承自动建立，以及错误发生在构造还是访问。

下一页用实际缺字段原因说明初始化协调，不提前给答案。



I super() and Required State 必需状态

Excerpt · complete definition in notes

```
class Student(Person):  
    def __init__(self, name, age, school):  
        super().__init__(name, age)  
        self.school = school
```

Actual complete-program output 实跑输出

Alex, 18, SJTU

- Establish required name / age
- Then establish school
- In this case, super finds Person

Complete runnable example 完整程序

14

Slide 14 · super() and Required State 必需状态

上一例访问value触发AttributeError，重写init不会自动调父init。这里明确调用super建立name/age，再增加school。

单继承中找到Person；一般规则沿实际类型MRO在当前类之后继续，不机械要求永远第一行。

I Reuse and Override 复用与重写



Excerpt · complete definition in notes

```
def give_raise(self, fraction, bonus=0.1):  
    super().give_raise(fraction + bonus)
```

Actual complete-program output 实跑输出

Chen 115.00

- Reuse the existing pay update
- The actual receiver is retained
- Base.report selected child in the previous example

Complete runnable example 完整程序

15

Slide 15 · Reuse and Override 复用与重写

保留往年Person/Manager薪资例子。Manager增加bonus，复用已有更新；真实示例输出姓氏Chen、115.00。

第12页实际child说明父函数仍用当前self。改接口或不变量需要回归，不保证改父类后所有子类自动正确。

| Polymorphism and Duck Typing 多态与鸭子类型



code

```
1 class Circle:
2     def area(self) -> float:
3         return 3.141592653589793
4 class Square:
5     def area(self) -> float:
6         return 4.0
7
8 total = 0.0
9 for shape in [Circle(), Square()]:
10     total += shape.area()
11 print(f"{total:.4f}")
```

output

7.1416

16

Slide 16 · Polymorphism and Duck Typing 多态与鸭子类型

同一area调用可以作用在不同类型，只要接口参数与行为一致，共同基类并非调用必需。

方法名存在不保证正确语义；普通Shape抛NotImplementedError只在调用时失败，不自动成为抽象类。

I Quiz 2 · Dispatch and Initialization



扫码作答 · 2 题

1 父方法内的self调用 2 重写初始化与缺字段

Submit this round once. 按已教规则推理。

<https://taohuang.info/cs1602/zh/quiz/10/>

17

Slide 17 · Quiz 2 · Dispatch and Initialization

两题只用已解释的继承、接收者和初始化。分别推理函数实现来源与实例状态，不根据类名猜答案。

下页给实际结果和原分布，不再发问题。

I Quiz Review · Reuse and State



Question	Actual result	Reason
Q2	child	Inherited code retains the Child receiver
Q3	AttributeError	No assignment established value

方法查找和状态建立分开核对。

18

Slide 18 · Quiz Review · Reuse and State

Q2实际child，Q3访问缺字段AttributeError。继承report并未改变接收者，继承关系也没自动执行被重写的初始化。

两个问题不同，不能用一句super能复用就代替状态分析。

II · Built-in Operation Protocols

从普通方法进入Python文档定义的操作协议。不是任意自造双端下划线名字都有解释器行为。

先解释表示，再学运算支持范围、反射与容器协议。

| String Representations 字符串表示



Which representation does each operation use?

```
1 class Point:
2     def __init__(self, x: float, y: float) -> None:
3         self.x, self.y = x, y
4     def __str__(self) -> str:
5         return f"({self.x}, {self.y})"
6     def __repr__(self) -> str:
7         return f"Point({self.x}, {self.y})"
8
9 p = Point(3, 4)
10 print(p)
11 print(repr(p))
12 print([p])
```

20

Slide 20 · String Representations 字符串表示

先说明str/repr都返回字符串，再预测print、repr和列表内显示分别使用哪个。

本页不显示结果，下一页静态答案来自实际程序。



I `__str__` and `__repr__`

code

```
1 class Point:
2     def __init__(self, x: float, y: float) -> None:
3         self.x, self.y = x, y
4     def __str__(self) -> str:
5         return f"({self.x}, {self.y})"
6     def __repr__(self) -> str:
7         return f"Point({self.x}, {self.y})"
8
9 p = Point(3, 4)
10 print(p)
11 print(repr(p))
12 print([p])
```

output

```
(3, 4)
Point(3, 4)
[Point(3, 4)]
```

21

Slide 21 · `__str__` and `__repr__`

真实三行分别为坐标字符串、Point构造样式和列表内Point表示。默认str可回退repr，反过来不成立。

可重建是设计目标非安全eval承诺。默认地址不作固定答案。

| Supported and Unsupported Operands 操作数范围



Return/action	Meaning 含义
Result	This implementation supports the operands
NotImplemented	Let operator dispatch try other applicable paths
raise NotImplementedError	An exception, not operator fallback

| 区分返回哨兵与抛出异常。

22

Slide 22 · Supported and Unsupported Operands 操作数范围

先解释NotImplemented返回哨兵，再在Vector的方法使用。它不是False，也不同于异常NotImplementedError。

不做哨兵真假测试，最终不支持的运算通常TypeError。

| Addition and New Results 加法与新结果



Excerpt · complete definition in notes

```
def __add__(
    self,
    other: "Vector",
) -> "Vector":
    if not isinstance(other, Vector):
        return NotImplemented
    x = self.x + other.x
    y = self.y + other.y
    return Vector(x, y)
```

Actual complete-program output 实跑输出

Vector(4, 6)

- This + creates a new Vector
- The operands remain unchanged
- Check support before reading fields

Complete runnable example 完整程序

23

Slide 23 · Addition and New Results 加法与新结果

本例两个向量相加产生新对象，不改输入。短片段链接含完整定义与调用的讲义，不伪装独立运行。

接口由实现决定，不概括所有加法都复制。

I Reflected Operations 反射运算



Excerpt · complete definition in notes

```
def __rmul__(  
    self,  
    scalar: float,  
) -> "Vector":  
    return self * scalar
```

Actual complete-program output 实跑输出

```
Vector(3, 6)  
Vector(3, 6)
```

- Vector × scalar: `__mul__`
- Scalar × Vector: `__rmul__`
- Reflected dispatch has type-dependent rules

Complete runnable example 完整程序

24

Slide 24 · Reflected Operations 反射运算

用数学语义自然的标量乘向量解释左右方向，实际两种乘法结果一致。

保留旧radd的分派知识，完整子类型优先与同类型限制放参考，不说永远先左再右。

I Augmented Assignment 增强赋值



Does the alias observe the changed object?

```
1 class Value:
2     def __init__(self, n: int) -> None:
3         self.n = n
4     def __add__(self, other: "Value") -> "Value":
5         return Value(self.n + other.n)
6     def __iadd__(self, other: "Value") -> "Value":
7         self.n += other.n
8         return self
9
10 a = Value(1)
11 b = a
12 a += Value(2)
13 print(a.n, b.n, a is b)
14 a = a + Value(3)
15 print(a.n, b.n, a is b)
```

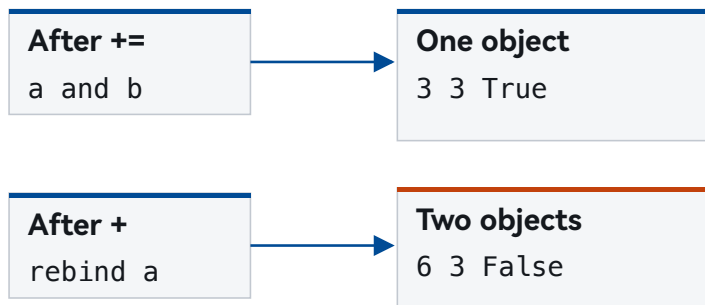
25

Slide 25 · Augmented Assignment 增强赋值

已经说明iadd返回本例self，再预测别名会观察什么。+=是否原地取决于方法和回退，不按语法表面判断。

下一页图中观察值来自此完整程序。

| Object Mutation and Rebinding 修改与重新绑定



This implementation only 按本例实现观察; += need not preserve identity.

26

Slide 26 · Object Mutation and Rebinding 修改与重新绑定

实际先3 3 True，再6 3 False。本例iadd修改共享对象，普通加法新建后只重新绑定a。

图不作为+=普遍定义，缺方法或哨兵回退也可使其产生新对象。

| Length, Indexing and Membership 长度与下标



Excerpt · complete definition in notes

```
def __getitem__(self, index: int) -> str:
    return self._songs[index]

def __setitem__(
    self,
    index: int,
    value: str,
) -> None:
    self._songs[index] = value
```

Actual complete-
program output 实跑
输出

2 changed True
first
changed

- len → `__len__`
- Membership → `__contains__`
- Negative indices and `IndexError` follow this list contract

Complete runnable example 完整程序

27

Slide 27 · Length, Indexing and Membership 长度与下标

保留Playlist的独立集合、读写下标、长度与成员。完整程序链接有全部定义。

len需非负整数，赋值和查询不同；不是所有对象都有这些协议。

| Sequence-Style Iteration 序列式迭代



What index is requested after the last item?

```
1 class Values:
2     def __init__(self):
3         self.items = ["A", "B"]
4     def __getitem__(self, index):
5         print("index", index)
6         return self.items[index]
7
8 print(list(Values()))
```

28

Slide 28 · Sequence-Style Iteration 序列式迭代

没有iter时可以从getitem请求0、1继续。先预测最后还请求哪个下标以及如何停止。

下页回顾实际0/1/2请求，不把None当通用结束。

I Equality and Identity 相等与身份



code

```
1 class Item:
2     def __init__(self, value):
3         self.value = value
4     def __eq__(self, other):
5         if not isinstance(other, Item):
6             return NotImplemented
7         return self.value == other.value
8
9 a = Item(2)
10 b = Item(2)
11 c = a
12 print(a == b, a is b, a is c)
```

output

True False True

29

Slide 29 · Equality and Identity 相等与身份

上一页实际请求0、1、2，IndexError结束后得到两个值。然后区分自定义eq与对象身份，实际 True False True。

is不受eq重定义影响，也不只允许None；默认普通对象相等通常按身份。

I Identity and Type Checks 身份与类型



Question 问题	Operation 操作
Same object?	<code>a is b</code>
Equal by the contract?	<code>a == b</code>
The None singleton?	<code>x is None</code>
Accept subclasses?	<code>isinstance(x, Base)</code>
Exact runtime type?	<code>type(x) is Base</code>

30

Slide 30 · Identity and Type Checks 身份与类型

用途决定选择：is检查身份，eq走相等协议，isinstance考虑子类。精确type也有明确使用场景。

数值和字符串比较不依赖驻留，完整内省工具在参考。

Quiz 3 · Equality and Identity



扫码作答 · 1 题

1 相等字段、不同实例与别名

Submit this round once. 按已教规则推理。

<https://taohuang.info/cs1602/zh/quiz/10/>

31

Slide 31 · Quiz 3 · Equality and Identity

题目显式定义eq，先查实际支持类型再比较字段。不要靠小整数或字符串缓存猜身份。

这里只提交一次，下一页回顾结果。

I Quiz Review · Values and Aliases



code

```
1 class Item:
2     def __init__(self, value):
3         self.value = value
4     def __eq__(self, other):
5         if not isinstance(other, Item):
6             return NotImplemented
7         return self.value == other.value
8
9 a = Item(2)
10 b = Item(2)
11 c = a
12 print(a == b, a is b, a is c)
```

output

True False True

32

Slide 32 · Quiz Review · Values and Aliases

两个实例字段相等，因此eq为True，身份不同为False，c绑定a为True。

运算协议需自然含义、支持类型和副作用边界，不要求背全部特殊方法名单。上半节在这里结束。

HALFWAY

Break

两个45分钟的自然分界。上半节讲类之间复用和Python操作接口。

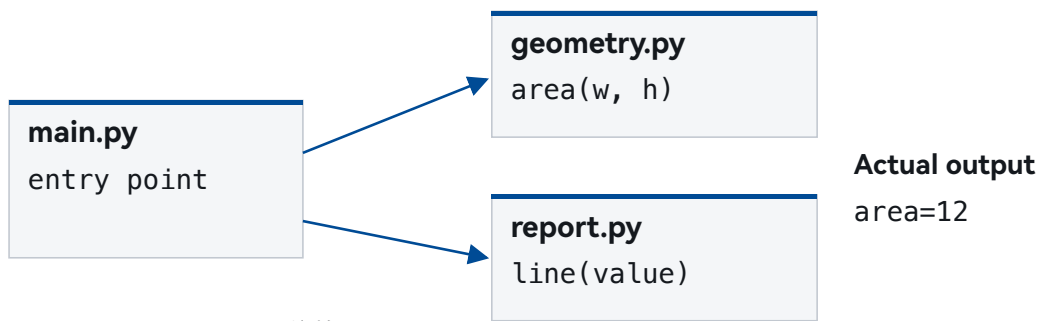
下半节讲真实文件、导入引用和运行环境，继续使用已学的绑定与证据，不重新介绍全部对象语法。

III · Modules and Name Bindings

模块为代码与数据提供命名空间，不是所有架构唯一最高层次。

以真实多文件项目开始，不把单浏览器代码框当作完整工程。

I A Small Program Across Files 多文件程序



Dependency edges 导入依赖; not threads or simultaneous calls.

35

Slide 35 · A Small Program Across Files 多文件程序

geometry定义操作，report负责显示，main组织入口。依赖箭头不表示三个线程或函数同时执行。

实际三个文件执行得到area=12，讲义给全部文件和本机运行条件。

| Modules as Objects 模块对象



code

```
1 import math
2 print(math.sqrt(16))
3 print(type(math).__name__)
```

output

```
4.0
module
```

36

Slide 36 · Modules as Objects 模块对象

math是已有模块对象，成员通过属性访问。本例结果4.0与module。

本讲自写模块常为py文件，但还有扩展、内置与冻结模块，不能用文件扩展名定义所有模块。

| Explicit Imports 显式导入



Form 写法

Access 访问

import math	math.sqrt(16)
import math as m	m.sqrt(16)
from math import sqrt	sqrt(16)
from math import sqrt as root	root(16)

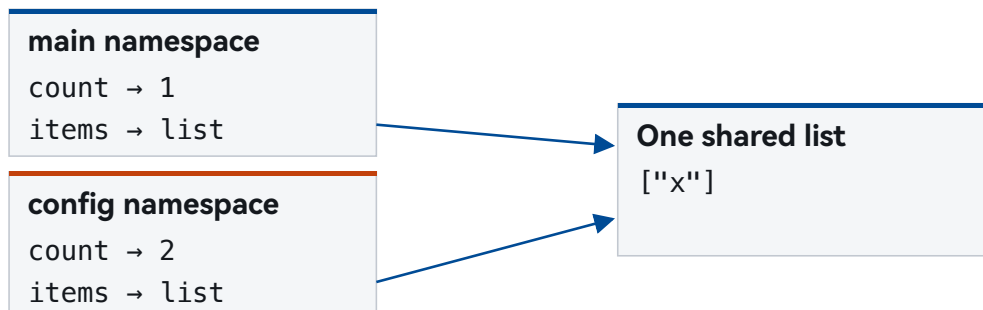
37

Slide 37 · Explicit Imports 显式导入

四种绑定方式已有真实讲义例子，别名只改变导入者访问名，不改模块原名。

形式不同不代表对象复制，下一页分析引用。

Module and Member Bindings 模块与成员绑定



Rebinding vs mutation 重新绑定与修改对象; observed: 1 2 ['x']

38

Slide 38 · Module and Member Bindings 模块与成员绑定

`import`绑定模块对象，`from`绑定选中的对象引用。模块字段重新绑定不会自动改本地名字；可
变列表修改仍可见。

图先给规则，后面的新预测用另一轮相同机制，不借用旧网页状态。



I Import Style and Exports 导入风格

- **Explicit names 显式名称**: keep origins readable
- **Separate groups 导入分组**: standard, third-party, local
- **Avoid wildcard imports** in ordinary course code

来源与导出范围须明确，不把风格建议当语法限制。

39

Slide 39 · Import Style and Exports 导入风格

保持来源明确，星号可能覆盖已有名字，普通课程代码避免使用。

星号是合法语法并受all等规则控制，不说无条件导入所有名称或永远禁止。



I Module Initialization 模块初始化

```
helper.py
print("module body")
def action():
    print("function body")
```

```
main.py
import sys
import helper
import helper
same = sys.modules["helper"]
print("cached", same is helper)
helper.action()
```

Multi-file example 多文件示例 · save and run locally.

40

Slide 40 · Module Initialization 模块初始化

先预测导入执行哪些顶层语句，def只建立函数，调用后才运行函数体。

本页只显示真实文件源码，不提前给答案，下一页给实际结果。

| Repeated Imports and sys.modules 导入缓存



helper.py

```
print("module body")
def action():
    print("function body")
```

main.py

```
import sys
import helper
import helper
same = sys.modules["helper"]
print("cached", same is helper)
helper.action()
```

Actual Python 3.13 output · real files

```
module body
cached True
function body
```

| Multi-file example 多文件示例 · save and run locally.

41

Slide 41 · Repeated Imports and sys.modules 导入缓存

实际顶层只打印一次module body，重复普通导入命中缓存，显式调用action才打印function body。

限定同进程同名普通导入，显式reload或新进程另分析。

| Mutation and Module Rebinding 模块更新



config.py

```
count = 1  
items = []
```

main.py

```
import config  
from config import count, items  
config.count = 2  
config.items.append("x")  
print(count, config.count, items)
```

| Multi-file example 多文件示例 · save and run locally.

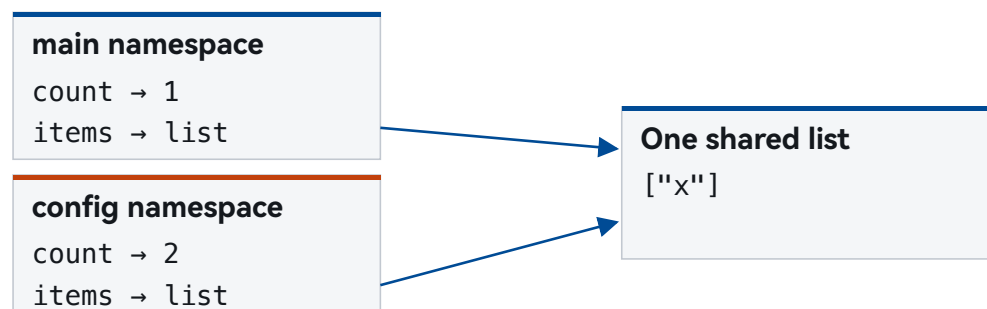
42

Slide 42 · Mutation and Module Rebinding 模块更新

两个文件都实际存在。先预测模块count换值和列表追加对from名字的影响。

这页不显示结果，下一页关系图来自实际进程。

| Imported References 导入引用



Rebinding vs mutation 重新绑定与修改对象; observed: 1 2 ['x']

43

Slide 43 · Imported References 导入引用

实际1 2与含x列表。数字字段改绑不改本地count，列表仍是同一个对象。

不称from复制对象或实时链接，也不在单片上伪造项目运行。

| Module Names and Entry Points 模块名与入口



tools.py

```
print("load")
if __name__ == "__main__":
    print("demo")
```

main.py

```
import tools
import tools
print("main")
```

| Multi-file example 多文件示例 · save and run locally.

44

Slide 44 · Module Names and Entry Points 模块名与入口

先根据name预测guard是否执行。tools的顶层load不在guard内，仍会运行。

两文件定义与入口清楚，下一页分别给直接执行和导入结果。

Direct Execution and Import 直接与导入



Entry 入口

Actual output 实际输出

Run tools.py

load
demo

Run main.py

load
main

| main guard只控制该块，其他顶层副作用仍会发生。

45

Slide 45 · Direct Execution and Import 直接与导入

直接tools打印load与demo，main导入打印load与main。guard不执行测试不表示模块完全无副作用。

完整本机文件可复现，不要求先在网页运行上一例。

I Quiz 4 · Imported State and Entry Guards



扫码作答 · 2 题

1 模块数字改绑与共享列表 2 普通导入缓存与入口guard

Submit this round once. 按已教规则推理。

<https://taohuang.info/cs1602/zh/quiz/10/>

46

Slide 46 · Quiz 4 · Imported State and Entry Guards

手机标明两个文件和全新普通脚本条件，不用网页共享上题命名空间。所有绑定和入口机制已教。

各题提交一次，下一页给完整实际输出。

I Quiz Review · Imports and State



Question	Actual output
Q5	1 2 ['x']
Q6	load main

47

Slide 47 · Quiz Review · Imports and State

Q5仍为1 2含x，引用和重新绑定不同；Q6仅load和main，普通缓存与guard分别解释。

查看原分布，不再提交或凭模块加载总数猜答案。

I Regular Packages 普通包



mypackage/geometry.py

```
def area(w, h):  
    return w*h
```

main.py

```
from mypackage import geometry  
print(geometry.area(3, 4))
```

Actual Python 3.13 output · real files

12

Multi-file example 多文件示例 · save and run locally.

48

Slide 48 · Regular Packages 普通包

普通包包含init和子模块，这个实际项目得到12。namespace包是另一机制，不把缺init当通用写法。

这里只投影两个主要文件，完整目录和init说明在讲义，网页不提供不完整工程按钮。

IV · Environment and Execution

从导入行为进入环境。检查缓存、模块来源与解释器，不要求背所有查找器实现。

下面用独立临时项目，不在真实学生目录留下遮蔽文件。

I Module Search 模块搜索



- 1 Cache 缓存** Check an existing module entry.
- 2 Finders 查找器** Built-in, frozen and other import finders.
- 3 Paths 路径** Top-level path-based lookup uses sys.path.

启动方式与缓存影响选择，当前目录不永远第一。

普通脚本常将脚本目录加入路径，但并非所有环境当前目录第一；缓存、内置和冻结查找器也会影响。

path项目不全是普通目录，包内路径与顶层路径也不同，参考保留细节。

| A Same-Named Local File 同名本地文件



random.py

```
print("local random")
```

main.py

```
import random  
print(random.randint(1, 3))
```

| Multi-file example 多文件示例 · save and run locally.

51

Slide 51 · A Same-Named Local File 同名本地文件

新普通脚本进程在受控目录中导入random，先预测randint可否使用。

不在真实项目创建同名文件，不提前显示错误，下一页给诊断。

I Identifying the Actual Environment 确认环境



Previous prediction · actual result

local random

```
AttributeError: module 'random' has no attribute 'randint' (consider renaming '/privaterandom.py' since it has the same name as the standard library module named 'random' and prevents importing that standard library module)
```

- **sys.executable**: selected interpreter
- **module.__file__**, when present: source
- **python -m pip --version**: matching installer

52

Slide 52 · Identifying the Actual Environment 确认环境

上一页实际加载本地random并发生AttributeError。检查解释器、模块路径和pip对应关系，不把所有属性错误归因遮蔽。

文件位置随环境改变，本例不显示用户私有目录或固定地址。

I Virtual Environments 虚拟环境



- `python -m venv .venv`
- **Use that environment's interpreter and pip.**
- **Dependency isolation**, not a security sandbox

确认解释器与安装器对应，再处理依赖。

53

Slide 53 · Virtual Environments 虚拟环境

venv组织第三方依赖，不是运行未知学生代码的安全沙箱。两平台激活命令在讲义说明，讲堂不要求安装新依赖。

推荐对应解释器的pip，避免已安装却导入不到。

I Installing Compatible Distributions 安装兼容包



Task	Command pattern
Install	<code>python -m pip install PACKAGE</code>
Inspect	<code>python -m pip list</code>
Upgrade	<code>python -m pip install --upgrade PACKAGE</code>
Remove	<code>python -m pip uninstall PACKAGE</code>
Local wheel	<code>python -m pip install compatible.whl</code>

54

Slide 54 · Installing Compatible Distributions 安装兼容包

这是占位命令表，不实际卸载他人依赖。wheel和pin需匹配Python与平台，旧NumPy1.26不能当3.13正确示例。

发行工具知识保留，不强制学生安装全部生态库。

| The Scientific Python Ecosystem 科学计算生态



Library	Typical use
NumPy	Arrays and numerical operations
Matplotlib	Visualization
SciPy	Scientific algorithms
pandas	Tabular data
pytest	Behavioral tests

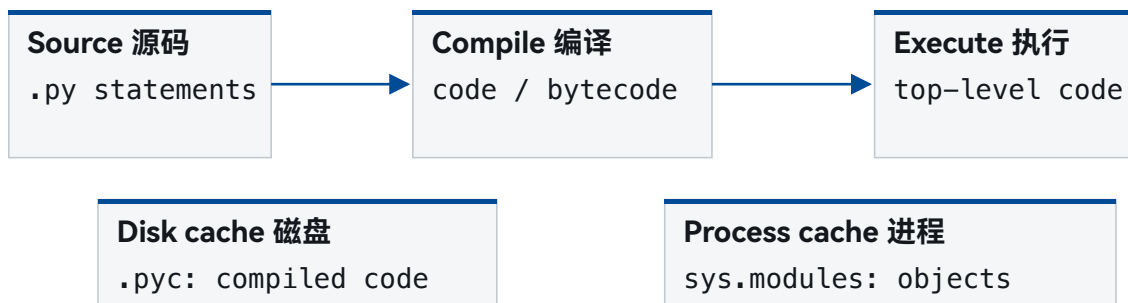
55

Slide 55 · The Scientific Python Ecosystem 科学计算生态

概览当前任务相关库，完整网络/游戏与其他功能保留参考，不删除旧名称。

官网用途不是本讲全部安装要求，团队简化数组项目仍按课程安排。

CPython · Source and Execution 源码与执行



CPython 3.13 model 缓存层不同; new processes still initialize modules.

56

Slide 56 · CPython · Source and Execution 源码与执行

CPython结合编译与执行，源码成为代码对象和字节码，再由运行时处理。不是所有Python实现都等同CPython。

编译目标可为中间表示或原生码，不沿用必转汇编与逐条转机器码的硬分类。

| Bytecode Operations 字节码操作



Actual output 实跑输出

3	RESUME	0
4	LOAD_FAST_LOAD_FAST BINARY_OP RETURN_VALUE	1 (a, b) 0 (+)

- Load operands 取值
- Add 运算
- Return 返回
- Version-specific instructions

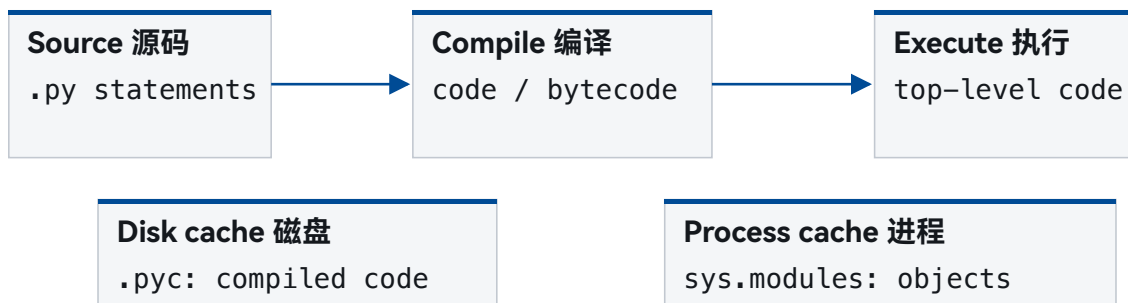
57

Slide 57 · Bytecode Operations 字节码操作

真实3.13反汇编只识别取操作数、运算、返回三个动作，不背所有指令和偏移。

这是执行实现观察，不是程序性能证明，调用帧也不是字节码操作数栈同一层表示。

I Two Kinds of Caching 两层缓存



CPython 3.13 model 缓存层不同; new processes still initialize modules.

58

Slide 58 · Two Kinds of Caching 两层缓存

sys.modules存本进程已初始化对象，pyc存磁盘编译缓存。新进程使用有效缓存也会执行顶层初始化。

缓存有效性、权限和启动参数影响写入，不简单说每次都免编译或免执行。

| Cached Code and Runtime Work 缓存与运行



Will another fresh process print the top-level message?

```
1 # helper.py
2 print("helper")
3 value = 3
4
5 # main.py
6 import helper
7 print(helper.value)
```

59

Slide 59 · Cached Code and Runtime Work 缓存与运行

这是明确两个文件的静态表示，不加运行按钮。先问已有pyc时新进程会不会打印helper。

下一页实际启动两个新进程给结果，不删除用户缓存来测速度。

| Implementation and Cost 实现与代价



Process	Actual output
1	helper 3
2	helper 3

- New processes still initialize modules.
- A cache does not hide source or prove faster code.

60

Slide 60 · Implementation and Cost 实现与代价

两个进程实际都打印helper与3。首次导入成本还包括依赖、加载和初始化，不能全部归编译。

pyc不是跨版本通用保密格式，真正工作量和证据到L11进一步分析。



I Solo Assignment 2 · Counting Primes

- **Five methods:** brute force, $\sqrt{n}$, $6k \pm 1$, sieve, Miller–Rabin
- **First verify**, then measure a feasible workload
- **Three solo assignments: 15% total; this one: 5%**

Due: 1 Dec, 23:59 GMT+8. 本周任务是素数，大整数第12周启动。

按现行配置第10周启动素数五法，截止由solo2统一校历读取，三份各5%，不用旧大整数7.5%安排。

五种完整算法不在此页公开，L11继续讲方法和规模。慢算法允许测较小输入，不能换算法冒充。

I Lab 10 · Interfaces and Module Organization



Part	Work	Raw points
A	Shapes / accounts / Money / Deck	16 auto + 8 staff
B	Real module project + evidence	8 auto + 4 staff
C	Environment and debugging	0-point record

62

Slide 62 · Lab 10 · Interfaces and Module Organization

保留五项原练习，四类题加真实工程适配，三张记录卡分开。接口在实验明确，不能自定牌面又要求统一自动判题。

HW4和solo2入口分开，课程权重不变；本地保存与验收不等于线上提交。

| Reuse, Protocols and Evidence



- Inherited state and behavior
- Supported operands and failures
- Imported bindings and side effects
- Interpreter and cache conditions

63

Slide 63 · Reuse, Protocols and Evidence

复用、操作语法和模块结构仍需说明行为，重点核对状态、输入类型、异常与环境条件。

组织方式不会代替验证；先找实际证据，再作正确性或性能结论。

| Complexity, Exceptions and Testing



- How much work does the implementation perform?
- How should failures be handled?
- What evidence supports correctness?

64

Slide 64 · Complexity, Exceptions and Testing

下一讲正式分析复杂度、异常和测试。本讲用少量预期异常表达接口失败，没有提前塞入完整异常处理语法。

主线到此结束，后面完整机制与旧例供查阅，不计入90分钟必讲。

Reference 参考

参考保留多继承、完整协议、模块细节、工具与旧应用。长程序链接完整讲义，片段不设置独立运行按钮。

旧知识不删，当前项目与历史大整数安排分清。

I Person and Manager · Complete Example



Excerpt · complete definition in notes

```
class Manager(Person):
    def __init__(self, name, pay):
        super().__init__(name, "manager", pay)
    def give_raise(
        self,
        fraction,
        bonus=0.1,
    ):
        super().give_raise(fraction + bonus)
```

Actual complete-program output 实跑输出

Chen 115.00

- name / job / pay
- get_last_name / give_raise
- Same-name def rebinding is not argument overloading

Complete runnable example 完整程序

66

Slide 66 · Person and Manager · Complete Example

保留完整Person/Manager及表示、姓氏和更新接口。片段依赖Person，讲义独立入口含全部定义。

显式基类调用有特定含义，但不作为全面取代super的推荐。

I Required State and Compatible Interfaces



- Inherited names do not create instance fields.
- Override signatures and outcomes need a contract.
- `NotImplementedError` is raised when called.
- Initialization order depends on state requirements.

字段存在、行为兼容与是否抽象是不同问题。

67

Slide 67 · Required State and Compatible Interfaces

进一步检查子类能否替代基类，不只看继承语法。改变错误、返回或有效域可能破坏使用者预期。

普通基类可创建而area调用报错，不暗中引入abc机制。

Cooperative Multiple Inheritance 合作式多继承



Actual output 实跑输出

```
['Combined', 'Left', 'Right', 'Root', 'object']  
['Combined', 'Left', 'Right', 'Root']
```

- **C3 MRO**, not BFS
- **super continues after the current class**
- Cooperative methods must call the next path

68

Slide 68 · Cooperative Multiple Inheritance 合作式多继承

实际Combined/Left/Right/Root/object顺序和一次调用链。Left后可为Right，Root只一次。

MRO不自动执行所有父方法；合作式代码需要自身正确使用super和接口。



I MRO Constraints 方法顺序约束

code

```
1 class A: pass
2 class B(A): pass
3 class C: pass
4 class D(B, C): pass
5 print([c.__name__ for c in D.__mro__])
```

output

```
['D', 'B', 'A', 'C', 'object']
```

69

Slide 69 · MRO Constraints 方法顺序约束

D/B/A/C/object明确不是BFS。相互矛盾的基类先后顺序会在创建类时报TypeError，完整反例在讲义。

不按旧关系图先入为主，实际mro和约束分别解释。

| Inheritance and Composition 继承与组合



Choice	Behavioral requirement
Subclass	Preserve what clients depend on
Delegate / compose	Reuse another object without claiming its full interface
Mutable Square/Rectangle	Independent width/height updates may conflict with equal sides

70

Slide 70 · Inheritance and Composition 继承与组合

现实Square是Rectangle并不保证可独立更新尺寸的程序接口适合继承。组合和委托也能复用。
保留旧谨慎继承的设计动机，不用一条现实分类作正确性证明。



| Name Mangling Across Subclasses

Excerpt · complete definition in notes

```
def __init__(self, values):  
    self.items = []  
    self.__update(values)  
  
__update = update
```

Actual complete-program output 实跑输出

```
[1, 2, ('a', 3)]
```

- `__update` → `_Mapping__update`
- Subclass update may have another signature
- Name mangling is not access control

Complete runnable example 完整程序

71

Slide 71 · Name Mangling Across Subclasses

接续L09：基类保存自己的update，子类不同签名不破坏初始化。完整Mapping/Child实际结果为两整数及一对键值。

改写仍可以从外部访问，不当作权限边界。

Introspection Tools 内省工具



Actual output 实跑输出

```
True False
True
{'name': 'Rex'}
['Dog', 'Animal', 'object']
['name']
```

- Type vs isinstance
- Class MRO and ordinary instance dict
- dir is a diagnostic aid

72

Slide 72 · Introspection Tools 内省工具

type、isinstance、issubclass和类mro有不同用途。普通实例dict记录字段，dir是诊断工具。

不是每个对象有dict，也不是所有描述符按简单实例优先查找；完整接口以文档为准。

| Functions and Code Objects 函数与代码对象



code

```
1 def add(a, b):  
2     return a + b  
3  
4 print(add.__name__)  
5 print(add.__code__.co_varnames)  
6 print(add.__code__.co_names)  
7 print(add.__code__.co_consts)
```

output

```
add  
( 'a', 'b' )  
( )  
(None, )
```

73

Slide 73 · Functions and Code Objects 函数与代码对象

先解释局部名、引用名与常量，再访问code属性。函数仍可赋名或作参数，保留旧函数对象例子。

这些是静态元数据，不代表当前所有运行状态，不提前lambda或闭包。

I Rich Comparisons and Sorting 富比较



Symbol	Method
< / <=	__lt__ / __le__
== / !=	__eq__ / __ne__
> / >=	__gt__ / __ge__
Unsupported equality	Try applicable paths, then identity
Equality and hash	Equal keys must have equal hashes

74

Slide 74 · Rich Comparisons and Sorting 富比较

六关系保留，但没有req式名字；不支持比较如何分派需依协议。仅定义eq通常使类不可哈希。若设计hash须与相等一致，不能把可变业务对象随意作字典键。

I Readable and Informative Representations



Excerpt · complete definition in notes

```
def __repr__(self) -> str:  
    return f"Point({self.x}, {self.y})"
```

Actual complete-program output 实跑输出

```
(3, 4)  
Point(3, 4)  
[Point(3, 4)]
```

- str may fall back to repr
- Containers display repr
- Both must return str

Complete runnable example 完整程序

75

Slide 75 · Readable and Informative Representations

Point完整演示保留默认/自定义表示与容器行为。可重建和调试可读是设计目标，不保证eval安全。

实际结果来自完整定义，不打印不稳定地址作固定答案。

| Container Operations 容器操作



Excerpt · complete definition in notes

```
def __setitem__(  
    self,  
    index: int,  
    value: str,  
) -> None:  
    self._songs[index] = value
```

Actual complete-program output 实跑输出

```
2 changed True  
first  
changed
```

- Negative indices
- Length and membership
- IndexError ends sequence fallback

Complete runnable example 完整程序

76

Slide 76 · Container Operations 容器操作

保留全部add、读写、长度和成员，完整接口在讲义。explicit iter None会阻止回退，L13再展开迭代器。

切片是否支持需另写契约，不按getitem名字猜任意索引都可用。



| Binary and Reflected Dispatch

Excerpt · complete definition in notes

```
def __mul__(
    self,
    scalar: float,
) -> "Vector":
    if not isinstance(scalar, (int, float)):
        return NotImplemented
    return Vector(self.x * scalar, self.y * scalar)

def __rmul__(
    self,
    scalar: float,
) -> "Vector":
    return self * scalar
```

Actual complete-program output 实验输出

```
Vector(3, 6)
Vector(3, 6)
```

- A right subtype can have priority
- Same-type reflection is not always retried
- Implicit methods are usually type-level

Complete runnable example 完整程序

77

Slide 77 · Binary and Reflected Dispatch

保留add/radd、标量左右方向和哨兵。不支持操作不会因为有个名字就正确运行。

实例临时加len不能等同隐式len支持，讲义有真实TypeError例子。



| In-Place Operations and Aliases

Actual output 实验输出

```
3 3 True
6 3 False
```

- This iadd returns self
- Another iadd may return a new object
- Missing/unsupported iadd can fall back

78

Slide 78 · In-Place Operations and Aliases

再次把协议条件说完整，保留列表与元组的原地/新对象知识，但不套通用等式。

实际两行观察用于解释别名，不复制旧地址。



| Triangle and Vector Protocol Examples

Excerpt · complete definition in notes

```
def __lt__(
    self,
    other: "Triangle",
) -> bool:
    if not isinstance(other, Triangle):
        return NotImplemented
    return self.area() < other.area()
```

Actual complete-program output 实跑输出

```
True 3
4
```

- This Triangle orders by area
- Indexing selects edges
- Invalid updates preserve old edges

Complete runnable example 完整程序

79

Slide 79 · Triangle and Vector Protocol Examples

旧Triangle比较与下标更新完整保留，几何含义在本例明确定义。L09的n维Vector仍通过链接承接。

不能说任意Triangle相加都是自然几何运算。



I Point, Complex and Mapping Examples

code

```
1 def isomorphic(s: str, t: str) -> bool:
2     if len(s) != len(t):
3         return False
4     forward, reverse = {}, {}
5     for a, b in zip(s, t):
6         if a in forward and forward[a] != b:
7             return False
8         if b in reverse and reverse[b] != a:
9             return False
10        forward[a], reverse[b] = b, a
11    return True
12
13 print(isomorphic("egg", "add"))
14 print(isomorphic("foo", "bar"))
```

output

True
False

80

Slide 80 · Point, Complex and Mapping Examples

同构字符串用两映射确保一一对应，实际egg/add为True、foo/bar为False。Point2D/nD维度和Complex加法完整讲义保留。

HappyNumber与L09有完整链接，不只列名称当作保全。



I A Complete Multi-File Program

geometry.py

```
"""Geometry operations."""  
def area(w, h):  
    return w*h
```

main.py

```
import geometry  
import report  
value = geometry.area(3, 4)  
print(report.line(value))
```

Actual Python 3.13 output · real files

```
area=12
```

| Multi-file example 多文件示例 · save and run locally.

81

Slide 81 · A Complete Multi-File Program

保留main/geometry/report的实际依赖和输出，所有三文件在讲义完整列出，源码保存与运行有明确路径。

本页选两个文件，其他文件不藏在运行按钮，网页没有伪项目执行。



Initialization Order and Circular Imports

```
a.py
import b
value = 1
```

```
b.py
from a import value
```

Actual Python 3.13 output · real files

```
ImportError: cannot import name 'value' from 'a' (consider renaming
'/privatea.py' if it has the same name as a library you intended to
import)
```

Multi-file example 多文件示例 · save and run locally.

82

Slide 82 · Initialization Order and Circular Imports

循环访问未建立的名称产生实际ImportError。初始化次序、def绑定和调用不能混在一起。
调整职责或延后访问，不将换IDE当语义修复。



importlib.reload

```
settings.py
print("load")
value = 1
```

```
main.py
import importlib
import settings
from settings import value
settings.value = 9
print(value, settings.value)
importlib.reload(settings)
print(value, settings.value)
```

Actual Python 3.13 output · real files

```
load
1 9
load
1 1
```

Multi-file example 多文件示例 · save and run locally.

83

Slide 83 · importlib.reload

重载实际再次打印load，将模块value重建为1，但之前from的名字不是自动实时链接。
用importlib代替旧imp；不是并发安全生产热更新保证。

| Aliases and Future Statements 别名与future



code

```
1 from __future__ import annotations
2
3 class Node:
4     def copy(self) -> Node:
5         return Node()
6
7 print(Node().copy().__class__.__name__)
```

output

Node

84

Slide 84 · Aliases and Future Statements 别名与future

as改变本地访问名，不修改模块本身。future是编译指令，允许合法文档/注释前置，不要求物理第一行。

annotations从3.7可选，在3.13须显式启用，不是类型校验。



| Search Paths and Environment Details

Mechanism	Condition
sys.modules	Already initialized process objects
Import finders	Built-in / frozen / custom and path-based
Path entries	Script context / PYTHONPATH / site / zip
Launch options	Normal script, -m and -I differ

85

Slide 85 · Search Paths and Environment Details

保留pth、site-packages和路径条件；项不全为普通目录，包搜索也有自己的上下文。
不把当前工作目录优先说成无条件规则，也不随意修改全局环境。



| Packages and Exports 包与导出

Item	Meaning
Regular package	<code>__init__.py</code> establishes initialization
Namespace package	A distinct package mechanism
<code>__all__</code>	Controls intended wildcard exports
Relative imports	Need a suitable package context

86

Slide 86 · Packages and Exports 包与导出

普通包与namespace分开；main的直接路径与-m包启动不是同一上下文。
保留旧导出、规范和相对路径知识，完整实际项目在讲义。

I pip, Wheels and Compatible Versions



Concern	Check
Interpreter	python -m pip targets that environment
Distribution	Python / ABI / platform compatibility
Sources	PyPI or a known compatible local wheel
Operations	Install / list / upgrade / uninstall

87

Slide 87 · pip, Wheels and Compatible Versions

保留旧pip与wheel知识，不执行示范卸载命令。固定版本应与本课解释器兼容，不能照抄1.26pin。

命令占位与实际执行结果明确区分。



| Library Reference 库参考

Libraries	Scope
NumPy / Matplotlib	Arrays / visualization
SciPy / pandas	Scientific algorithms / tables
Scrapy / BeautifulSoup	Web processing
Pygame / pytest	Games / tests

88

Slide 88 · Library Reference 库参考

旧科学、网络、游戏名称与用途完整保留，不因主线只介绍五库就删掉。

概览不要求本周一次装全部依赖，也不从旧生态图推断当前版本。



| Implementations and Toolchains 实现与工具链

Examples	Category
CPython / PyPy / Jython	Python implementations
Cython / Nuitka	Compilation tools
Native / bytecode / JIT	Compilation and execution can be combined

89

Slide 89 · Implementations and Toolchains 实现与工具链

编译和解释可结合，不能把所有工具都称解释器。CPython的C实现与标准库不同组成保留。

旧Coco定位不明，只记录待考证不编造介绍；源码阅读使用当前仓库而非旧行号。

| Explicit Compilation 显式编译



code

```
1 source = "answer = 2 + 3"
2 code = compile(source, "trusted.py", "exec")
3 namespace = {}
4 exec(code, namespace)
5 print(namespace["answer"])
```

output

5

90

Slide 90 · Explicit Compilation 显式编译

可信源码compile得到code object，exec在指定命名空间执行得到5。不是直接将pyc二进制交exec。

py_compile/compileall命令只选择自己的教学目录，不扫描整台机器或执行未知字符串。



I dis and Function Metadata 字节码与元数据

code

```
1 def add(a, b):  
2     return a + b  
3  
4 print(add.__name__)  
5 print(add.__code__.co_varnames)  
6 print(add.__code__.co_names)  
7 print(add.__code__.co_consts)
```

output

```
add  
( 'a', 'b' )  
( )  
(None, )
```

91

Slide 91 · dis and Function Metadata 字节码与元数据

co_consts、co_varnames、co_names的含义先解释再用，dis只识别动作。

指令版本、偏移与源码文件行号都不是跨版本事实，不要求死记。



I .pyc Cache Conditions 缓存条件

- **Validity and version tags** matter
- **Permissions and -B** can change cache writing
- **Imported modules and direct scripts** differ
- **Compiled code is not source confidentiality**

| 缓存存在不意味着新进程跳过模块初始化。

92

Slide 92 · .pyc Cache Conditions 缓存条件

缓存有效性和权限决定能否写/复用；通常直接运行脚本不生成自己的pyc，导入模块不同。

字节码不跨全部实现/版本通用，也不提供可靠源码保密。

I BigInt · A Later Assignment 大整数预告



Representation

Supported interface

Decimal input string

Sign and grouped digits

Display / compare

str, repr, eq, lt

Arithmetic

add, sub, mul without whole-operand int
conversion

当前个人作业2是素数计数，大整数后续启动。

93

Slide 93 · BigInt · A Later Assignment 大整数预告

保留旧BigInt表示与全部基本运算，正式启动在第12周，不作为本周大整数7.5%项目发布。

这里不给完整作业算法答案，历史分项和当前课程权重区分。

I BigInt · Verification and Report



- **Zero, leading zeros and signs**
- **Unequal lengths, negative differences, carries and borrows**
- **Use int as a test oracle**, not the implementation
- **One-page report**: representation, signs, evidence

94

Slide 94 · BigInt · Verification and Report

零与前导零、符号、不等长、进位借位和竖式乘法保留。int可以对拍，不能替代实现。

旧20/20/20/10/15/5/10只作历史记录，不按旧项目评分重新发布。

Counting Primes · Contracts and Evidence



Contract	Evidence
$0 \leq p < n; n \leq 100000$	Small values and cross-method counts
Five genuine methods	Inspect source and report, not wrappers
MR bases 2 and 3	Deterministic within the stated bound
Timing after correctness	Feasible sizes, repeated measures, environment

95

Slide 95 · Counting Primes · Contracts and Evidence

统一候选区间与小值，MR确定性只在本题域内，不推任意大整数结论。

助教核对不同方法，报告记录可完成规模和三次公平测量。L11继续具体算法，不要求未教先考完整五法。

I Lab 10 · Fixed Interfaces 固定接口



Entry	Required boundary
Shapes	Finite positive dimensions; ValueError
Accounts	Integer cents and explicit overdraft rules
Money	Decimal display → rounded cents; sentinel operands
Deck	Fixed cards; invalid deal preserves state
Project	Five real files; silent imports and guarded self-tests

96

Slide 96 · Lab 10 · Fixed Interfaces 固定接口

五项题面、骨架、测试和记录用同一约定，HW4与solo2分开。原始分不改学期权重或日历。
完成本地不等于提交/部署，AI声明与操作证据真实，未做部分明确说明。