程序设计与计算思维

第 27 讲：课程复习

韩文弢

清华大学

2026 年 6 月

课程回顾

课程设计思路

本课程参考 MIT 的 Introduction to Computational Thinking，以**案例驱动**的方式讲解计算思维，将编程教学融入案例。

模块	讲次	核心案例
概论	lec0	
图像处理	lec1–8	数组、变换、卷积、牛顿法
科学数据处理	lec9–16	随机变量、蒙特卡罗法、随机游走、线性模型
气候模式	lec17–22	ODE → 雪球地球 → 平流扩散 → PDE 模板
工程实践与 C++ 基础	lec23–26	终端、模块化、Git、AI 编程、C++ 基础

考试范围

期末测验只考以下内容，不考 C++：

1. Python 语言基础（数据类型、控制流、函数、类）
2. NumPy、matplotlib、Pandas 的基本使用
3. 计算思维的基本概念和运用

Python 语言基础

数据类型

类型	可变	示例	备注
int	否	42, 0b1010, 2**1000	任意精度
float	否	3.14, 1e-3	IEEE 754 双精度
bool	否	True, False	True == 1
str	否	"hello"	不可变序列
list	是	[1, 2, 3]	有序、可变
tuple	否	(1, 2, 3)	有序、不可变
dict	是	{"a": 1}	键值对
set	是	{1, 2, 3}	无序、不重复
NoneType	否	None	空值

运算符

类别	运算符
算术	`+` `-` `*` `/` `//` `%` `**`（注意 `//` 整除、`**` 幂运算）
比较	`<` `<=` `==` `!=` `>` `>=`（支持链式：`0 <= x < 10`）
逻辑	`and` `or` `not`（短路求值）
成员	`in`、`not in`（列表/字典/集合）
身份	`is`、`is not`（主要用于 `None` 检查）
赋值	`=`、`+=`、`-=`、`*=`、`//=` 等
三元	`x if cond else y`（条件表达式）
矩阵	`A @ B`（NumPy 矩阵乘法）

字符串操作

```
s = 'Hello'
len(s)                  # 5
s[0], s[-1]             # 'H', 'o'
s[1:3], s[::-1]         # 'el', 'olleH'
s + ' World'            # 拼接
'ha' * 3                # 'hahaha'
text.split()            # 按空白分割
','.join(words)         # 用逗号连接
line.strip()            # 去除首尾空白

# f-string 格式化
f'{name}: {score:.1f}'      # 指定精度
f'{x:10.3f}'                # 宽度 + 精度
f'Left {{ Right }}'         # 转义花括号
```

列表与元组

```python
a = [1, 2, 3]
a.append(4)              # [1, 2, 3, 4]
a.pop()                  # 移除末尾，返回 4
a.sort()                 # 原地排序
a.sort(key=lambda x: -x)  # 自定义排序
a.reverse()              # 原地反转
[0] * 5                  # [0, 0, 0, 0, 0]
list('abc')              # ['a', 'b', 'c']

# 切片（左闭右开）
a[1:3]                   # 索引 1, 2
a[:2]                    # 从开头
a[::2]                   # 步长 2
a[::-1]                  # 反转
```

列表与元组

```
# 元组（不可变）
t = (1, 2, 3)
x, y, z = t                    # 解包
```

字典与集合

```python
d = {'Alice': 95, 'Bob': 87}
d['Alice']              # 95
d.get('X', 0)           # 不存在返回 0
d['Charlie'] = 78       # 添加/修改
del d['Bob']            # 删除
d.pop('Charlie')        # 删除并返回
d.keys(), d.values(), d.items()   # 视图
for k, v in d.items():            # 遍历

# 集合
s = {1, 2, 3}
s.add(4)
s.remove(2)
a & b, a | b, a - b, a ^ b     # 交集、并集、差集、对称差集
```

控制流

条件语句

```
if x > 0:
    print('positive')
elif x == 0:
    print('zero')
else:
    print('negative')
```

while 循环

```
while condition:
    # ...
    if done:
        break
    if skip:
        continue
```

for 循环

```
for i in range(5):
    print(i)

for x in my_list:
    print(x)

for i, x in enumerate(lst):
    print(i, x)
```

for…else

```
for item in items:
    if match(item):
        break
else:
    # 循环正常结束（无 break）
    print('not found')
```

函数定义

```
def sigmoid(x, alpha=1.0):
    """Docstring."""
    return 1.0 / (1.0 + np.exp(-alpha * x))

# 调用方式
sigmoid(0.5)                    # 位置参数
sigmoid(0.5, alpha=2.0)         # 关键字参数

# 函数作为对象
def apply(f, x):
    return f(x)
apply(np.sqrt, 4.0)  # 2.0
```

lambda 与高阶函数

```python
# lambda 表达式
square = lambda x: x ** 2
sorted(pairs, key=lambda p: p[1])

# map, filter, sum
list(map(str, [1, 2, 3]))
sum(x**2 for x in range(10))  # 生成器表达式
min(candidates, key=lambda c: c[1])

# 高阶函数：返回函数的函数
scale = lambda a: lambda x: a * x
double = scale(2)
double(5)  # 10
```

LEGB 名字查找规则

Python 在查找变量名时，按以下**由内到外**的顺序搜索：

层级	名称	说明
L	Local	当前函数内部的局部变量
E	Enclosing	外层嵌套函数的变量（闭包）
G	Global	模块级别的全局变量
B	Built-in	Python 内建名称（`len`、`print` 等）

LEGB 名字查找规则

```
x = 'global'

def outer():
    x = 'enclosing'
    def inner():
        x = 'local'
        print(x)      # L → 'local'
    inner()

outer()                  # 'local'
print(x)                 # G → 'global'
```

若 inner 中没有 x = 'local'，则向上查找 E 层得到 'enclosing'；
若 outer 中也没有，则查 G 层得到 'global'。

推导式

```python
# 列表推导式
squares = [x**2 for x in range(10)]
evens = [x for x in range(20) if x % 2 == 0]

# 字典推导式
{name: score for name, score in data if score >= 60}
{v: k for k, v in original.items()}  # 翻转键值

# 生成器表达式（惰性求值）
sum(x for x in range(100))
min(abs(x) for x in values)
```

多重赋值与解包

```python
a, b = 0, 1                  # 多重赋值
x, y, z = (1, 2, 3)          # 元组解包
head, *_ = my_list            # 只要第一个
a, *rest, b = [1,2,3,4]      # 中间收集

# 函数调用中解包
args = [3, 6]
list(range(*args))            # 等价于 range(3, 6)

# 链式比较
0 <= x < 10                   # 等价于 0 <= x and x < 10
```

错误处理

```
try:
    result = risky_operation()
except ValueError as e:
    print(f'值错误: {e}')
except Exception:
    print('其他错误')
finally:  # 无论是否发生异常都会执行
    cleanup()

# 抛出异常
raise ValueError('参数不合法')
raise NotImplementedError('子类必须实现')
```

文件与上下文管理器

```
with open('data.txt', 'w') as f:
    f.write('Hello\n')

with open('data.txt') as f:
    content = f.read()
    for i, line in enumerate(f, 1):
        print(i, line.strip())
```

常用内建函数

函数	说明	示例
range(start, stop, step)	整数序列	range(5)、range(10, 0, -1)
enumerate(iterable, start)	带索引遍历	for i, x in enumerate(lst):
zip(a, b)	并行遍历	for x, y in zip(xs, ys):
sorted(iterable, key, reverse)	排序（返回新对象）	sorted(d.items(), key=lambda x: x[1])
map(func, iterable)	映射	list(map(str, [1, 2, 3]))
sum(iterable, start)	求和	sum(x**2 for x in range(10))

常用内建函数

函数	说明	示例
`isinstance(obj, cls)`	类型检查	`isinstance(dog, Animal)`
`type(obj)`	获取类型	`type(42) == int`
`len(obj)`	长度	`len([1, 2, 3]) == 3`

常用标准库

模块	常用功能	课程中的使用
collections	defaultdict、Counter	PageRank 计数、词频统计
pathlib	Path、文件路径操作	文件读写、目录遍历
math	sin、cos、sqrt、pi	数学函数
time / timeit	计时、性能测量	算法性能对比
urllib.request	urlretrieve	下载图像数据

面向对象编程

类的定义与特殊方法

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self):
        return f'Vector({self.x}, {self.y})'

    def __add__(self, other):
        return Vector(self.x + other.x,
                      self.y + other.y)

    def __mul__(self, scalar):
        return Vector(self.x * scalar,
```

类的定义与特殊方法

```
                self.y * scalar)

    def magnitude(self):
        return (self.x**2 + self.y**2)**0.5
```

常用特殊方法：__init__（构造）、__repr__（字符串表示）、__add__/__mul__（运算符重载）、__len__、__getitem__、__iter__/__next__（迭代器）。

继承与抽象基类

```python
from abc import ABC, abstractmethod

class RandomVariable(ABC):
    @abstractmethod
    def sample(self):
        pass

    def mean(self):
        raise NotImplementedError

class Gaussian(RandomVariable):
    def __init__(self, mu, sigma2):
        self.mu = mu
        self.sigma2 = sigma2
```

继承与抽象基类

```
    def sample(self):
        return self.mu + (self.sigma2**0.5
                          ) * np.random.randn()

class Bernoulli(RandomVariable):
    def __init__(self, p):
        self.p = p

    def sample(self):
        return int(np.random.rand() < self.p)
```

继承与抽象基类

核心概念：`class Child(Parent)`、`super().__init__()`、`ABC` + `@abstractmethod`、`raise NotImplementedError`、`isinstance()`、类属性 vs 实例属性。

生成器与 yield

```
def count_up(n):
    i = 0
    while i < n:
        yield i
        i += 1

list(count_up(5))  # [0, 1, 2, 3, 4]

# 实际案例：SIR 模型模拟
def discrete_SIR(s0, i0, r0, T=1000):
    s, i, r = s0, i0, r0
    for _ in range(T):
        ds, di, dr = ...  # 状态转移
        s, i, r = s+ds, i+di, r+dr
```

生成器与 yield

```
        yield (s, i, r)

results = list(discrete_SIR(0.99, 0.01, 0))
```

- yield 使函数变为生成器，**惰性**产出值
- 用 list() 消费生成器，或用 for 循环遍历
- 适合逐步产生大量数据的场景（ODE 模拟、随机游走）

NumPy、matplotlib、pandas

数组操作

```python
import numpy as np

a = np.array([1, 2, 3, 4, 5])
a.shape          # (5,)
a.dtype          # int64
a[1:3]           # array([2, 3])
a[a > 2]         # array([3, 4, 5])
a * 2            # array([2, 4, 6, 8, 10])
a.sum()          # 15
a.mean()         # 3.0

M = np.zeros((3, 4))
M[1, 2] = 7.0
M[:, 0]          # first column
```

核心概念

1. **向量化**：用数组运算替代循环，性能接近 C 语言
2. **广播**：不同形状的数组自动扩展进行运算
3. **索引**：`a[i]`、`a[i:j]`、`a[condition]`、`a[mask]`
4. **形状**：`.shape`、`.reshape()`、`.T`（转置）
5. **常用函数**：`np.zeros`、`np.ones`、`np.linspace`、`np.arange`、`np.meshgrid`

绘图基础

```
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.figure(figsize=(8, 4))
plt.plot(x, y, label='sin(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Sine Wave')
plt.legend()
plt.grid(True)
plt.savefig('sin.png', dpi=150)
```

绘图基础

常用图表类型：plot（折线）、scatter（散点）、imshow（图像/热力图）、hist（直方图）、bar（柱状图）、contourf（等高线填充）。

```python
import pandas as pd

df = pd.read_csv('data.csv')
df.head()
df.describe()

# 筛选
subset = df[df['age'] > 20]

# 分组统计
grouped = df.groupby('city')['salary'].mean()

# 排序
df.sort_values('age', ascending=False)
```

数据分析

核心概念：DataFrame、Series、读取数据（read_csv）、筛选（布尔索引）、分组（groupby）、排序（sort_values）。

计算思维

四大核心概念

概念	含义	课程案例
抽象	从具体问题中提取本质特征	网页→图、图像→数组、随机变量→抽象基类
分解	将复杂问题拆分为子问题	PageRank 迭代、PDE 模板分解、OOP 继承层次
建模	用计算模型描述现实问题	ODE 建模雪球地球、扩散方程建模海洋温度
计算实验	通过计算验证和理解模型	蒙特卡罗仿真、参数敏感性分析、可视化

科学研究范式

范式	代表	特征
第一	理论科学	用数学公式描述规律（牛顿力学）
第二	实验科学	通过实验验证假设（伽利略）
第三	计算科学	用计算机模拟复杂系统（天气预报）
第四	数据驱动	从海量数据中发现规律（机器学习）

计算科学工作流程

数据 → 输入 → 处理 → 模型 → 可视化 → 输出

- **数据**：图像像素、传感器读数、统计数据
- **处理**：数组运算、滤波、标准化
- **模型**：ODE、PDE、统计模型、PageRank
- **可视化**：折线图、热力图、动画、交互控件

课程中的重要案例

案例	计算方法	核心知识点
图像处理	数组索引、卷积	NumPy 数组操作
PageRank	迭代法、矩阵乘法	图建模、线性代数
牛顿法	迭代求根	数值方法、SymPy
ODE 求解	Euler / RK 方法	SciPy solve_ivp
雪球地球	ODE 耦合系统	相图分析、平衡态
蒙特卡罗	随机采样	概率统计、NumPy random
随机变量库	抽象基类 + 继承	OOP、运算符重载
平流扩散	有限差分、模板	PDE、边界条件、numba

备考建议

1. **理解概念**：计算思维的四大核心概念及其在案例中的应用
2. **读懂代码**：能理解 Python 代码的执行结果（侧重阅读理解）
3. **掌握基础**：数据类型、控制流、函数、类
4. **了解库的用法**：NumPy、matplotlib、pandas 的常用函数
5. **不考 C++**：lec24–26 的内容不在考试范围内

样题

单项选择题

1. 以下代码的输出是：

```
x = [1, 2, 3, 4, 5]
print(x[1:4])
```

A. `[1, 2, 3, 4]`　　B. `[2, 3, 4]`　　C. `[2, 3]`　　D. `[1, 2, 3]`

单项选择题

1. 以下代码的输出是：

```
x = [1, 2, 3, 4, 5]
print(x[1:4])
```

A. `[1, 2, 3, 4]`　　B. `[2, 3, 4]`　　C. `[2, 3]`　　D. `[1, 2, 3]`

答案：B。切片 `x[1:4]` 取索引 1、2、3，即 `[2, 3, 4]`。

单项选择题

2. 以下代码的输出是：

```
d = {'a': 1, 'b': 2, 'c': 3}
print(d.get('d', 0))
```

A. `1`　　B. `3`　　C. `0`　　D. 抛出 KeyError

2. 以下代码的输出是：

```
d = {'a': 1, 'b': 2, 'c': 3}
print(d.get('d', 0))
```

A. `1`　　B. `3`　　C. `0`　　D. 抛出 KeyError

答案：C。`d.get('d', 0)` 在键 `'d'` 不存在时返回默认值 `0`。

3. 以下关于 NumPy 的说法，**错误**的是：

A. `np.array([1, 2, 3])` 创建一维数组

B. 数组的 `shape` 属性返回其各维度大小

C. NumPy 数组的元素可以是不同类型

D. `a[a > 0]` 返回满足条件的元素

3. 以下关于 NumPy 的说法，**错误**的是：

A. `np.array([1, 2, 3])` 创建一维数组
B. 数组的 `shape` 属性返回其各维度大小
C. NumPy 数组的元素可以是不同类型
D. `a[a > 0]` 返回满足条件的元素

答案：C。NumPy 数组的所有元素必须是**同类型**的（由 `dtype` 决定）。

4. 以下代码的输出是：

```
class A:
    def greet(self):
        return 'A'

class B(A):
    def greet(self):
        return 'B'

obj = B()
print(obj.greet())
```

A. A　　B. B　　C. 报错　　D. None

4. 以下代码的输出是：

```
class A:
    def greet(self):
        return 'A'

class B(A):
    def greet(self):
        return 'B'

obj = B()
print(obj.greet())
```

A. A　　B. B　　C. 报错　　D. None

答案：B。B 继承 A 并重写了 greet()，obj.greet() 调用的是子类 B 的方法（多态）。

5. 蒙特卡罗方法估算 π 的基本原理是：

A. 用随机数生成器直接计算 π 的精确值

B. 在单位正方形内随机撒点，统计落入四分之一圆内的比例

C. 用迭代法逐步逼近 π

D. 求解与 π 相关的微分方程

5. 蒙特卡罗方法估算 π 的基本原理是：

A. 用随机数生成器直接计算 π 的精确值

B. 在单位正方形内随机撒点，统计落入四分之一圆内的比例

C. 用迭代法逐步逼近 π

D. 求解与 π 相关的微分方程

答案：B。在 1×1 的正方形内随机撒点，落入四分之一圆的比例约为 $\pi/4$，乘以 4 即得 π。

6. 以下代码的输出是 _____。

```
x = [i**2 for i in range(5) if i % 2 == 0]
print(x)
```

6. 以下代码的输出是 _____。

```
x = [i**2 for i in range(5) if i % 2 == 0]
print(x)
```

答案：`[0, 4, 16]`。`i` 取 0、2、4 时满足 `i % 2 == 0`，平方后为 0、4、16。

填空题

7. 以下代码的输出是 _____。

```
import numpy as np
a = np.array([[1, 2], [3, 4]])
print(a[1, 0])
```

7. 以下代码的输出是 _____。

```
import numpy as np
a = np.array([[1, 2], [3, 4]])
print(a[1, 0])
```

答案：3。a[1, 0] 是第 2 行第 1 列（索引从 0 开始）。

8. 以下代码的输出是 _____。

```
def f(x, y=10):
    return x + y

print(f(3))
print(f(3, y=20))
```

8. 以下代码的输出是 _____。

```
def f(x, y=10):
    return x + y

print(f(3))
print(f(3, y=20))
```

答案：13 和 23。`f(3)` 使用默认 `y=10` 得 13；`f(3, y=20)` 用关键字参数覆盖得 23。

填空题

9. 以下代码的输出是 _____。

```
s = {1, 2, 3, 2, 1}
print(len(s))
```

9. 以下代码的输出是 _____。

```
s = {1, 2, 3, 2, 1}
print(len(s))
```

答案：3。集合 s 中的元素不重复，最终包含 1、2、3 三个元素。

10. 在计算思维中，将复杂问题拆分为若干子问题的过程称为_____。

10. 在计算思维中，将复杂问题拆分为若干子问题的过程称为_____。

答案：分解。

11. 阅读以下代码，回答问题：

```
import numpy as np

def simulate(N=10000):
    rng = np.random.default_rng(42)
    x = rng.uniform(0, 1, N)
    y = rng.uniform(0, 1, N)
    inside = np.sum(x**2 + y**2 < 1)
    return 4 * inside / N

result = simulate(1000000)
print(f'pi ≈ {result:.4f}')
```

1. 解释第 5–7 行的数学原理：为什么 `4 * inside / N` 可以估算 π？
2. 如果将 `N` 从 10^6 增大到 10^8，估计值会怎样变化？为什么？

答案：

1. 在单位正方形 $[0,1)\times[0,1)$ 内随机撒点，点 (x,y) 落入四分之一圆 $x^2+y^2<1$ 的概率为 $\pi/4$。因此 `inside / N` 近似 $\pi/4$，乘以 4 得到 π 的估计。
2. `N` 增大后，估计值会**更接近** π 的真实值。因为样本量越大，由大数定律，频率越接近概率。

12. 阅读以下代码，回答问题：

```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, r):
        self.r = r
    def area(self):
        return 3.14159 * self.r ** 2
```

解答题

```
class Rectangle(Shape):
    def __init__(self, w, h):
        self.w = w
        self.h = h
    def area(self):
        return self.w * self.h

def total_area(shapes):
    return sum(s.area() for s in shapes)

shapes = [Circle(1.0), Rectangle(2, 3)]
print(total_area(shapes))
```

解答题

1. `Shape` 是什么？为什么不能直接 `Shape()` 创建对象？
2. `total_area` 函数中，`s.area()` 分别调用了哪个类的方法？这体现了 OOP 的什么特性？
3. 如果新增 `Triangle` 类需要继承哪个类？至少实现什么方法？

解答题

1. Shape 是什么？为什么不能直接 Shape() 创建对象？
2. total_area 函数中，s.area() 分别调用了哪个类的方法？这体现了 OOP 的什么特性？
3. 如果新增 Triangle 类需要继承哪个类？至少实现什么方法？

答案：

1. Shape 是抽象基类（ABC），含有 @abstractmethod 装饰的 area() 方法，不能直接实例化。
2. s.area() 对 Circle 对象调用 Circle.area()，对 Rectangle 对象调用 Rectangle.area()。这体现了**多态**——同一接口，不同实现。
3. 继承 Shape，至少实现 area() 方法。

期末安排

- 6 月 10 日（周三）：计算技术展望，课程答疑
- 6 月 12 日（周五）：期末测试，**闭卷**，10:00-11:00，建华楼 LG1-21
- 需要补交作业的同学请在 6 月 28 日前提交，根据补交政策给分