

AI 辅助编程（一）：需求、审查与验证

课件的每一页，配上讲这一页时说的话。幻灯片是网页原生渲染的，文字可以选中、可以搜索，也可以直接打印。



Introduction to Computation (CS1602) · Lecture 8

AI-Assisted Programming I

Instructor: Tao Huang

Part 3: Problem Solving with AI · Fall 2026

1

第七讲用任务、基准情形、终止条件和返回值分析了递归。本讲把这些检查用于同学、网络资料或 AI 提供的实现。

一份程序通过样例，还要核对输入范围、状态变化、边界和数据规模。课堂候选均为教学构造，不是真实模型回答记录。

Code Review and Verification 代码审查



L07 → L08 · the same checks, now applied to another source.

- **Input assumptions 输入前提**: which inputs are allowed?
- **Termination 终止**: base cases and progress, as in L07
- **Results 结果**: returned values and state changes

候选代码、行为约定、检查证据要相互对应。

2

第 2 页 · Code Review and Verification 代码审查

上一讲检查了递归的输入前提、基准和进展，也区分了打印指令与实际状态改变。别人提供的代码同样需要这些检查。

本讲先明确需求，再阅读实现、设计检查和定位失败。AI 可以参与这些工作，提交者仍要能够解释采用的代码。

I A Candidate Implementation 候选实现



The sample passes. The code is short.

Is that enough to accept it? 样例通过后，还要检查什么？

3

第 3 页 · A Candidate Implementation 候选实现

设想同学给了一段实现，样例运行正常，代码也很短。先不要评价来源，而要问：哪些行为已经确认，哪些还不知道？

例如，输入是否被改变，重复项是否保留，空输入是否处理，数据变多是否仍能使用。今天按这些具体问题审查候选。

Quiz 1 · Lists and Side Effects



扫码作答 · 一题

1 别名、remove 与输入变化

Predict both printed lists. Submit once.

<https://taohuang.info/cs1602/zh/quiz/8/>

开场题只考第五讲已介绍的列表别名和 `remove`。函数返回以后，原列表是什么？先独立预测两次打印。

本题只收一次答案。讲到输入不变的约定时再公布结果，不再扫码重做。

I Lecture Structure



- | | |
|----------------------|---|
| 1 Generate 生成 | Keep a candidate; do not confuse an explanation with execution. |
| 2 Specify 约定 | Define inputs, results, boundaries and side effects. |
| 3 Review 审查 | Read the implementation; check APIs and readability. |
| 4 Verify 验证 | Test failures and scale; debug and recheck corrections. |
| 5 Accept 采纳 | Record the source, changes and actual evidence. |

5

第 5 页 · Lecture Structure

先区分候选代码和实际执行记录，再把函数任务写成明确约定。随后分别检查可读性、功能、边界和效率。

末尾把原候选、反例、修正与复查结果整理成记录。较长的规范表和往年练习放在参考区，不要求课堂全部逐行阅读。

I · Generated Code and Evidence

样例只提供一组输入的证据。接下来区分文本生成、工具执行和按需求进行的验证，避免把解释当成运行记录。

I Language Models 大语言模型



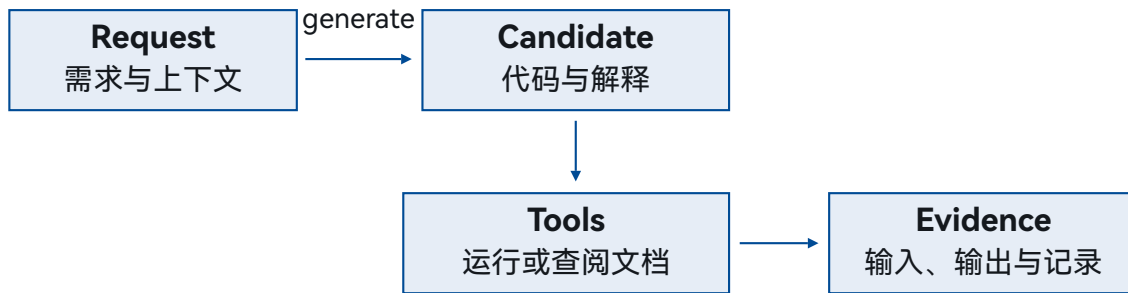
- **Token 文本单位**: a word, part of a word or punctuation
- **Generation 生成**: context → subsequent tokens → code and explanation
- **Evidence 证据**: recorded checks of that implementation

解释不能当作运行记录；工具实际执行的检查要单独核对。

这里的 token 是模型处理文本的单位，可能是词、词的一部分或标点，不固定等于一个字符。大语言模型按上下文生成后续 token，程序和解释本身不证明代码已经运行。

这里不展开模型结构，也不把所有模型概括成只会模仿。课堂关心的是采纳实现需要什么检查。

Generation and Tool Execution



Acceptance: compare the contract, code and evidence.

上方是根据需求生成候选，下方是工具运行或查阅文档后产生的证据。带工具的助手可以真的执行这些步骤，所以不能一概说 AI 从未运行代码。

要查看实际的输入、输出和测试记录。即使某次运行成功，也还要对照需求与代码判断其覆盖范围。

| API Availability 接口是否存在



code

```
1 s = "hello"
2 print(s.reverse())
```

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    print(s.reverse())
          ^^^^^^^^^
AttributeError: 'str' object has no attribute 'reverse'
```

9

第 9 页 · API Availability 接口是否存在

字符串没有 reverse 方法，实际运行触发 AttributeError。列表上的同名方法不能推导出字符串也有这个方法。

接口检查要核对对象类型、方法名称、参数、返回值和副作用。这是教学构造例子，不断言某个模型实际给过这段代码。

| Documentation Checks 文档核对



code

```
1 s = "hello"
2 print(s[::-1])
3 print(s)
```

output

```
olleh
hello
```

10

第 10 页 · Documentation Checks 文档核对

切片生成字符串 `olleh`，原变量 `s` 仍绑定 `hello`。反转方法的选择来自字符串接口与切片语义，而不是从列表方法名称类推。

字符串不可变也不是它必然没有 `reverse` 方法的逻辑证明；不可变对象可以提供返回新对象的方法。这里按实际文档核对。



I Empty Inputs 空输入

code

```
1 def average(nums: list[float]) -> float:
2     return sum(nums) / len(nums)
3
4 print(average([1, 2, 3]))
5 print(average([]))
```

traceback

```
Traceback (most recent call last):
  File "example.py", line 5, in <module>
    print(average([]))
    ~~~~~^~~~~~
  File "example.py", line 2, in average
    return sum(nums) / len(nums)
    ~~~~~^~~~~~
ZeroDivisionError: division by zero
```

11

第 11 页 · Empty Inputs 空输入

第一组得到 2.0，第二组的长度为零，计算时触发 ZeroDivisionError。普通样例成功没有决定空输入的行为。

若约定只接受非空列表，空输入不在范围内；若任务要求接受空列表，就要先约定结果或错误处理方式。不能根据候选当前行为倒推需求。



I Failure Categories 错误类别

Check 检查	Concrete case 具体例子	Question 审查问题
API availability	<code>"hello".reverse()</code>	Does that method exist on a string?
Boundary behavior	<code>average([])</code>	Is empty input allowed, and with what behavior?
Runtime version	<code>print "hello"</code>	Is the syntax valid in Python 3.13?
Performance	A search through a growing list	Is the repeated work suitable for the size?
Readability	<code>f(l, D)</code>	Can the purpose and state be understood?

不报错，不等于已满足功能、规模和可读性要求。

12

第 12 页 · Failure Categories 错误类别

先检查接口、行为边界和运行环境，再检查效率与可读性。它们不是只能归到某一类的互斥缺陷。

课堂四份候选分别强调接口、边界、重复查找和命名，但不预设每份代码恰好只有一个问题。

I Runtime Compatibility 运行环境



Python 2 syntax · excerpt

```
print "hello"
```

Python 3 syntax

```
print("hello")
```

左侧是 Python 2 写法，在本课程 Python 3.13 中会触发 `SyntaxError`；右侧是 Python 3 的形式。

库接口也可能随版本变化。这个例子说明兼容性检查，不说明当前模型普遍生成 Python 2。错误片段的运行结果已独立核对。

I Review Evidence 审查依据



Which evidence is missing when only an explanation is supplied?

- **Requirement 需求**: input domain and promised behavior
- **Implementation 实现**: bounds, state updates and side effects
- **Documentation 文档**: actual APIs and runtime version
- **Observed checks 检查记录**: test results and required scale

14

第 14 页 · Review Evidence 审查依据

只有解释时，还缺需求对应关系、接口核对、实际测试和规模证据。解释中的推理也可以审查，但不能默认它等同于执行。

下一节先把需求写完整，后面的测试与修改才有依据。

II · Function Specifications 函数规格

空输入和接口错误常暴露未写清的前提。先定义输入、结果和允许的状态变化，再评判一份实现。

I Ambiguous Requirements 不完整的需求



“Merge two lists, sorted.”

- Are both inputs **already sorted**?
- Must **duplicates** be kept?
- A **new list**, or may inputs be changed?
- Which **input sizes** must be handled?

先把未决定的行为写清楚，再评价实现。

16

第 16 页 · Ambiguous Requirements 不完整的需求

这句话没有明确两个输入是否已经有序，也没有说明重复元素和输入修改。不同实现都可能符合某一种理解。

请学生先补充这些约定，再看函数签名。需求写得更多不等于写得准确，关键是决定可观察的行为。

I Signatures and Behavior 签名与行为



Code / excerpt 代码或片段

```
def merge_sorted(  
    a: list[int], b: list[int]  
    ) -> list[int]:  
    ...
```

- **Signature 签名:** parameter and result types
- **Not specified:** ordering, duplicates or mutation
- **Not enforced:** runtime checks from annotations alone

| ... 表示尚未实现；类型提示不自动完成行为验证。

17

第 17 页 · Signatures and Behavior 签名与行为

签名描述参数和返回类型。排序、保留重复和输入不变没有由 `list[int]` 自动保证；Python 默认也不按类型注解强制检查实参。

编辑器或类型检查器可以使用这些信息。行为约定和实际实现仍需另行核对。省略号表示未实现，不是排序操作。



I A Complete Contract 完整约定

Code / excerpt 代码或片段

```
def merge_sorted(  
    a: list[int], b: list[int]  
) -> list[int]:  
    """Inputs are ascending.  
    Return a new sorted list.  
    Keep every duplicate.  
    Leave inputs unchanged.  
    """  
    ...
```

- **Empty:** $([], []) \rightarrow []$
- **Merge:** $([1, 3], [2]) \rightarrow [1, 2, 3]$
- **Duplicates:** $([1, 1], [1]) \rightarrow [1, 1, 1]$
- **Mutation:** check each input separately

接口、文字约定与例子共同说明任务，不是已完成的实现。

18

签名、文档字符串与例子共同说明任务。两个空列表、交错数值和重复项分别检查不同约定，还需要单独检查输入不变。

这份规格没有提供完整实现。自然语言不是签名的竞争对象，而是表达行为所必需的补充。

I Side Effects 输入变化



code

```
1 def without_two(items):
2     result = items
3     result.remove(2)
4     return result
5
6 values = [1, 2, 3, 2]
7 print(without_two(values))
8 print(values)
```

output

```
[1, 3, 2]
[1, 3, 2]
```

19

第 19 页 · Side Effects 输入变化

现在回到开场题。`result` 和 `items` 指向同一列表，`remove` 删除首个 2；返回值和原 `values` 都是 `[1, 3, 2]`。公布开场分布，不重新作答。

若需求是返回新列表、删除全部匹配项，这段候选同时违反输入不变与删除范围两项约定。

I A New Result 新结果



code

```
1 def remove_all(  
2     items: list[int], value: int  
3 ) -> list[int]:  
4     return [x for x in items if x != value]  
5  
6 values = [1, 2, 3, 2]  
7 print(remove_all(values, 2))  
8 print(values)
```

output

```
[1, 3]  
[1, 2, 3, 2]
```

20

第 20 页 · A New Result 新结果

列表推导逐项保留不等于 value 的元素，产生新列表。实际返回 [1, 3]，原输入仍是 [1, 2, 3, 2]。

这里的实现与删除全部且输入不变的约定一致。不能只看到返回值对了，就忽略输入是否被改变。

I Checking Input Preservation 输入保全



code

```
1 def remove_all(  
2     items: list[int], value: int  
3 ) -> list[int]:  
4     return [x for x in items if x != value]  
5  
6 values = [1, 2, 3, 2]  
7 before = values.copy()  
8 result = remove_all(values, 2)  
9 print(result == [1, 3])  
10 print(values == before)
```

output

```
True  
True
```

21

第 21 页 · Checking Input Preservation 输入保全

调用前保存输入副本，调用后分别检查结果与原输入。两次比较均得到 True。

本例元素是整数，浅拷贝已经够用。若输入是嵌套可变结构，需要根据允许变化的层级选择检查方式，不能把浅拷贝当成全部深层状态的独立快照。



I Prompt Components 提问内容

count_words(text: str) → dict[str, int]

- **Normalize**: split on whitespace and ignore case
- **Punctuation**: strip only . , ! ? ; : from the two ends
- **Empty tokens**: do not count words that become empty

Example 规格例子

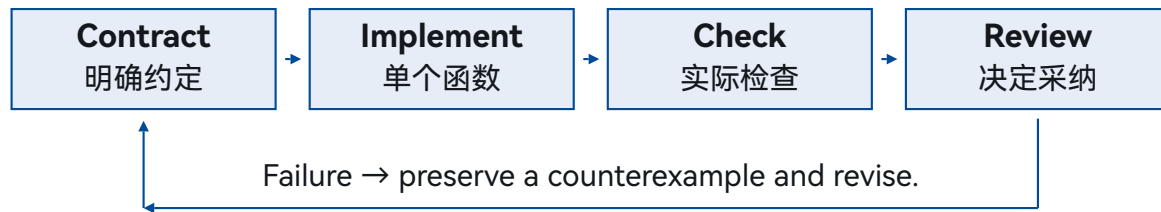
Expected 预期

"The cat. THE dog"	{"the": 2, "cat": 1, "dog": 1}
"" or "... !"	{}

Python 3.13; 只用 L01–L07 已教语法，仍需检查生成的实现。

除了函数名，词频规格还要决定拆分、大小写、标点范围和清理后空单词的处理。这里明确只去掉两端这六种标点，不能自动推广到全部语言文字。

提示中加入边界，是明确需求的方法，不保证模型必然遵守。取得实现后仍要检查。



Keep the original candidate and recheck earlier cases.

先明确一个函数的任务，再实现、检查与决定采纳。失败时保留输入和预期，修正候选并重新检查此前用例。

这与上一讲的问题分解衔接。AI 可以参与讨论和实现，但任务规模要让提交者能够阅读和解释，不把一大段无法理解的代码直接交上去。



I Failure Reports 失败报告

“It is wrong. Try again.” 缺少可复现信息

Record 记录项

A useful report 具体反馈

Input

[7]

Expected

0 · specified for fewer than two values

Actual

ValueError from max([])

Runtime

Python 3.13

Relevant code

The construction of gaps and final max call

24

第 24 页 · Failure Reports 失败报告

反馈要指出具体输入、预期、实际结果、环境和相关代码。完整反例使修改可以核对，而不是反复猜测需求。

工具给出的失败解释同样需要审查。没有运行就不写成已复测，也不要为显得完整编造结果。

III · Readable Code 可读代码

规格决定任务，接下来检查实现能否清楚表达任务。先读出行为，再改写名字和结构，不把格式代替逻辑。

I A Readability Review 可读性审查



Which keys are retained? 请先说明函数的行为。

```
1 def f(l, D):
2     R = []
3     for i in range(0, len(l)):
4         if l[i] in D:
5             if D[l[i]] > 0:
6                 R.append(l[i])
7     return R
```

26

第 26 页 · A Readability Review 可读性审查

函数按输入顺序遍历各键，只保留字典中存在且计数大于零的键。函数名 `f` 和参数 `l`、`D` 没有说明这些含义。

重复键也会重复保留。先把这个行为说清楚，再整理遍历、条件和名字，不先以行数判断质量。

I Names and Documentation 命名与说明



filter_positive(keys, counts) · a name that describes the task

- **Selection 选择**: present keys with positive counts
- **Order 顺序**: keep the input order
- **Duplicates 重复项**: keep repeated keys
- **Naming 命名**: snake_case for variables/functions; UPPER_CASE for constants

| Avoid l, 0, I and unwanted built-in shadowing; annotations are not a PEP 8 mandate.

27

第 27 页 · Names and Documentation 命名与说明

名字可以说明输入和任务，文档应补充顺序、重复项和缺失键的约定。避开难分辨的 l、O、I，也不要覆盖稍后需要调用的内置名字。

类型提示帮助表达接口，但没有类型提示不直接意味着 PEP 8 违规。

| Behavior-Preserving Refactoring 等价改写



code

```
1 def filter_positive(  
2     keys: list[str], counts: dict[str, int]  
3 ) -> list[str]:  
4     """Keep present keys with positive counts, in input order."""  
5     return [key for key in keys  
6             if counts.get(key, 0) > 0]  
7  
8 keys = ["a", "b", "c"]  
9 counts = {"a": 1, "b": 0, "c": 5}  
10 print(filter_positive(keys, counts))
```

output

```
['a', 'c']
```

28

第 28 页 · Behavior-Preserving Refactoring 等价改写

`counts.get` 对缺失键返回默认的 0，条件把缺失和非正计数一起排除。这个列表推导保留输入顺序和重复键，实际输出 a 与 c。

重构需要核对改写前后行为，不以缩短代码作为判断依据。完整多输入比较在验证脚本中执行。

| Whitespace and Layout 空格与布局



Recommended 推荐布局

```
spam(ham[1], {eggs: 2})
```

```
if x == 4:
    print(x, y)
```

```
i = i + 1
```

Style issue · not necessarily syntax

```
spam( ham[ 1 ], { eggs: 2 } )
```

```
if x == 4 :
    print(x , y)
```

```
i=i+1
```

这些片段展示括号内、标点前和赋值运算两侧的空格。它们依赖示例名字，不作为独立运行入口。

右侧在合适上下文中可以是合法 Python，但不符合推荐排版。不能把风格反例说成语法必然错误。

| Continuation Lines 续行



Align or hang 参数续行

```
result = some_function(first,
                        second)

result = some_function(
    first,
    second,
)
```

Operators and line length

```
income = (gross_wages
          + interest
          - deduction)

# PEP 8 code limit: 79
# Project/tool limits may differ.
```

30

第 30 页 · Continuation Lines 续行

括号内续行可以对齐到开括号，也可以采用悬挂缩进。运算符之前或之后换行都允许，局部保持一致；PEP 8 建议新代码在运算符之前换行。

通常代码行长为 79，注释和文档字符串为 72，团队可以约定其他长度。工具默认长度不等于唯一合法长度。



I Formatters and Linters 格式与静态检查

Work 分工	Command 工具	What it does 作用
Format 格式化	<code>ruff format file.py</code>	Adjust layout and whitespace
Lint 静态检查	<code>ruff check file.py</code>	Report enabled rules
Test 行为检查	Inputs + expected results	Check a specified requirement

- **Black** is an alternative formatter, not an extra requirement.
- **Ruff configuration** selects the rules; defaults are not all of PEP 8.

排版整齐或没有诊断，都不能代替功能验证。

格式化调整排版，lint 按启用规则检查一些代码问题。两者都不决定空列表等输入的任务含义。

Ruff 的规则配置需要记录，不把默认设置说成全部 PEP 8。Black 是另一种格式化选择，不要求同时安装。

| Rule Diagnostics 规则诊断



Checked file 检查文件

```
import os
print(total)
```

Actual Ruff report 实际报告

ruff 0.16.10

Rules: F401, F821

F401 · line 1

`os` imported but unused

F821 · line 2

Undefined name `total`

32

第 32 页 · Rule Diagnostics 规则诊断

本页实际使用标明版本的 Ruff，并显式选择 F401 与 F821。报告来自这两行代码，分别指出未使用导入 os 和未定义 total。

这是静态检查，不执行程序。规则提示需要理解后处理；没有报告也不能据此确认功能。

I Formatting and Behavior 格式与行为



Will formatting add empty-input behavior? 格式化能补上这个约定吗?

```
1 def average(nums: list[float]) -> float:
2     return sum(nums) / len(nums)
3
4 print(average([]))
```

33

第 33 页 · Formatting and Behavior 格式与行为

先口头预测：代码按格式排得整齐，空列表仍会除以零。这个行为由计算和输入约定决定，不由空格决定。

本页不再扫码。可直接对照前面的实际错误例子，区分格式工具和行为检查。

I Review Checklist 审查检查表



- Interface and contract 接口与约定
- Names and readable structure 命名与结构
- State updates, bounds and returns 状态与结果
- Unknown APIs and side effects 陌生接口与副作用
- Finite tests are evidence, not a universal proof.

34

第 34 页 · Review Checklist 审查检查表

阅读时把疑问具体化：哪条约定由哪些条件保证，状态怎样更新，边界是否访问合法，输入是否改变。

测试可以获得证据，但有限用例通常不能证明所有输入正确。下一节用反例定位候选的失败。

IV · Tests and Debugging 测试与调试

审查提出具体疑问，测试根据规格确认哪些输入失败。调试随后观察中间状态，解释为什么失败。

I Tests from the Contract 由需求设计测试



max_gap · sort for comparison, but **leave the input unchanged**

Input 输入	Expected 预期	Purpose 检查内容
[3, 9, 1, 15]	6	Ordinary unsorted values
[]	0	Empty input
[7]	0	One element
[2, 2, 2]	0	Repeated values
[-3, -1, -10]	7	Negative values

36

第 36 页 · Tests from the Contract 由需求设计测试

先规定函数对空列表、单元素和重复项的结果，再列出预期。负数同样属于整数输入，差值在排序后计算。

不修改输入要单独检查。测试来自需求，不从候选写了哪些分支决定题目要求。



I A Minimal Counterexample 最小反例

code

```
1 def max_gap(nums: list[int]) -> int:
2     ordered = sorted(nums)
3     gaps = [ordered[i + 1] - ordered[i]
4             for i in range(len(ordered) - 1)]
5     return max(gaps)
6
7 print(max_gap([7]))
```

Error line

```
ValueError: max()
iterable argument
is empty
```

37

第 37 页 · A Minimal Counterexample 最小反例

一个元素无法形成相邻差，gaps 为空，max 在空可迭代对象上触发 ValueError。右栏只摘录实际异常的最后五行，完整回溯见讲义。

这里已经约定长度不足 2 返回 0，因此候选缺少相应行为。不是所有 max 函数都必须自动返回零，修改依据来自本题规格。

I A Guard and Regression Checks 边界与回归



code

```
1 def max_gap(nums: list[int]) -> int:
2     if len(nums) < 2:
3         return 0
4     ordered = sorted(nums)
5     gaps = [ordered[i + 1] - ordered[i]
6             for i in range(len(ordered) - 1)]
7     return max(gaps)
8
9 print(max_gap([3, 9, 1, 15]))
10 print(max_gap([7]))
11 print(max_gap([]))
```

output

```
6
0
0
```

38

第 38 页 · A Guard and Regression Checks 边界与回归

短输入先返回 0，其余输入排序、构造相邻差并求最大值。实际输出分别是 6、0、0。

修正失败输入后，也要复查此前正常的输入。本页使用列表推导，不提前使用生成器表达式；完整讲义另核对输入不变。

I Coverage and Limitations 覆盖与限制



- **Boundaries 边界**: empty, one, duplicates, zero and negatives
- **State 状态**: output order and input preservation
- **Scale 规模**: the actual data size and performance requirement
- **Domain 输入范围**: wrong-type handling is contract-dependent

测试能暴露反例；有限测试通常不能证明所有输入正确。

检查维度要与输入范围相符。只接收整数列表的函数，不未经要求就扩展成接受任意对象。

重复项、顺序、副作用和规模可能是任务的重要约定。有限测试通过后，逻辑审查和运行环境检查仍然必要。



I Reading a Traceback 阅读回溯

code

```
1 def read_number(text):
2     return int(text)
3
4 read_number("abc")
```

traceback

```
Traceback (most recent call last):
  File "example.py", line 4, in <module>
    read_number("abc")
    ~~~~~^~~~~~
  File "example.py", line 2, in read_number
    return int(text)
ValueError: invalid literal for int() with base 10: 'abc'
```

40

第 40 页 · Reading a Traceback 阅读回溯

这份回溯由 Python 3.13 实际生成。调用行显示 `read_number` 的输入，函数中的 `int` 语句是失败位置，最后一行给出 `ValueError` 及信息。

源码和波浪线可能占多行，不能固定说倒数第二行就是错误行。先找异常类型，再沿调用位置核对失败语句。异常处理语法留到第十一讲。

I A State-Update Failure 状态更新错误



Input: [1, 1, 0, 1, 1, 1]. What will be returned?

```
1 def longest_run(bits: list[int]) -> int:
2     current = 0
3     best = 0
4     for bit in bits:
5         if bit == 1:
6             current += 1
7         else:
8             best = max(best, current)
9             current = 0
10    return best
```

41

第 41 页 · A State-Update Failure 状态更新错误

候选只在遇到 0 时把 current 计入 best。输入 [1, 1, 0] 会得到正确的 2，但末尾更长的连续段没有遇到 0。

先预测最后 best 和 current 的值，再用下一页的实际状态说明。这个错误是本讲构造的候选，不把它归到原旧课件正确实现。

I Breakpoints 断点



Code · selected frame

```
else:  
    best = max(best, current)  
    current = 0  
return best
```

Variables · longest_run

```
best = 2  
current = 3
```

Paused before return; the final run has not updated best.
调试界面示意 · verified state, not a screenshot

42

第 42 页 · Breakpoints 断点

图明确标为调试界面示意，不是真实 VS Code 截图。状态由 Python 3.13 在返回语句的 line 事件核对：best 为 2，current 为 3。

断点让执行在语句之前暂停。选中当前调用帧后观察它的变量，就能发现最后连续段尚未计入最大值。



I Stepping and Watches 单步与观察

Action 动作	Common key 按键	Execution 执行效果
Step Over	F10	Execute this line without entering the call
Step Into	F11	Enter the function called on this line
Step Out	Shift+F11	Finish this function and return to its caller
Continue	F5	Resume until the next breakpoint or completion

Variables / Watch use the **selected frame** while paused. Use toolbar actions if keys differ.

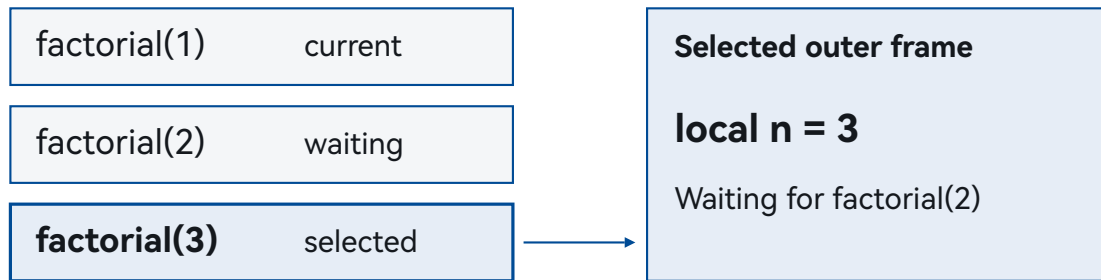
Step Over 执行当前行而不进入被调用函数，Step Into 进入调用，Step Out 执行完当前函数回到调用者，Continue 运行到下一断点或结束。

系统和键盘可能改变按键，所以也说明动作名称。Watch 在暂停时按选中上下文求值，不是程序运行时的持续监视器。

I Recursive Call Frames 递归调用帧



Call stack · $n = 1$ is current



局部变量属于各自调用 · logical frames, not memory addresses

44

第 44 页 · Recursive Call Frames 递归调用帧

factorial(3) 进入 n 为 1 的调用时，栈中有 n 为 1、2、3 的三个独立局部环境。图中选择外层 factorial(3)，其局部 n 仍为 3。

调用栈面板可以切换帧，变量面板随选中帧变化。图表示逻辑调用关系，不是物理内存地址，也不是真实界面截图。

I Fix and Recheck 修正与复查



code

```
1 def longest_run(bits: list[int]) -> int:
2     current = 0
3     best = 0
4     for bit in bits:
5         if bit == 1:
6             current += 1
7         else:
8             best = max(best, current)
9             current = 0
10    return max(best, current)
11
12 print(longest_run([]))
13 print(longest_run([1]))
14 print(longest_run([1, 1, 0, 1, 1, 1]))
```

output

```
0
1
3
```

45

第 45 页 · Fix and Recheck 修正与复查

按任务约定，返回时比较已记录的 `best` 与未结束的 `current`。空输入为 0，单个 1 为 1，末尾三连 1 的例子为 3。

`print` 或调试器都可以帮助定位这两个状态。修正后再测全 0、全 1、正常中间段和末尾段；不是换一个工具就自动验证完成。

V · Performance Checks 效率检查

返回值正确以后，还要检查实际规模。这里使用已经学过的列表和集合，不提前展开第十一讲复杂度的正式定义与证明。

I Repeated List Searches 反复列表查找



code

```
1 def dedup(items: list[int]) -> list[int]:
2     result = []
3     for x in items:
4         if x not in result:
5             result.append(x)
6     return result
7
8 print(dedup([3, 1, 3, 5, 1]))
```

output

[3, 1, 5]

47

第 47 页 · Repeated List Searches 反复列表查找

程序实际输出 [3, 1, 5]，保留首次出现顺序。每次 `x not in result` 都在逐渐变长的列表中搜索。

阅读代码已经可以提出效率疑问。接着用显式比较计数核对，不能说性能问题读代码发现不了。

I Operation Counts 操作计数



Distinct n	List comparisons	Set queries
10	100	20
100	10,000	200
1,000	1,000,000	2,000

Two ordered passes over n distinct integers. Different units.

48

第 48 页 · Operation Counts 操作计数

对 n 个不同整数按相同顺序出现两轮，显式线性查找检查了 100、10000、1000000 次。集合版本的外层成员查询数分别是 20、200、2000。

两列单位不同：左边是显式比较，右边是成员查询调用数。不是测 CPU 指令，也不声称哈希查询固定一步。特定输入的比较数为 n^2 ，不等于所有输入都恰好如此。

I A Set and an Ordered Result 集合与顺序



Membership and output order need different structures.

```
1 def dedup(items: list[int]) -> list[int]:
2     seen = set()
3     result = []
4     for x in items:
5         if x not in seen:
6             seen.add(x)
7             result.append(x)
8     return result
```

49

第 49 页 · A Set and an Ordered Result 集合与顺序

seen 集合只记录出现过哪些整数，result 列表按第一次出现的顺序保存结果。不能直接把集合当成有序输出。

输入限定为可哈希整数。集合查找有较低平均代价，但不是无条件固定一步；若元素是列表，接口约定和实现还要另行处理。



I Equivalent Results 结果等价核对

Input 输入	List search 列表版	Set-assisted 集合版
[]	[]	[]
[3, 1, 3]	[3, 1]	[3, 1]
[-1, 0, -1]	[-1, 0]	[-1, 0]

- **Results match** on these recorded cases.
- **Inputs unchanged**: check separately, not from the result alone.

提速不能悄悄改变顺序、输入范围或重复项约定。

表中两个输出列来自完整函数的独立运行，三个用例分别得到空列表、[3,1]、[-1,0]。输入不变另行检查。

先确认行为一致再讨论效率，不能因输出相同就忽略顺序、输入域和副作用。

I Scale and Measurement 规模与测量



Measure 测量	Meaning 含义	Limit 限制
Operation count	Work performed on a chosen case	Identify the unit and input distribution
Elapsed seconds	Observed time in this environment	Hardware/runtime/background work affect it

- **Same task, data and environment** before comparing
- **Equivalent results and repeated measurements** before drawing conclusions

一次计时倍率不是通用规律；L11 再系统分析时间与空间。

51

第 51 页 · Scale and Measurement 规模与测量

确定的操作计数适合比较工作怎样随输入变化。秒数还受硬件、解释器、浏览器和其他任务影响，不能把一次测量倍率当作通用规律。

先确认任务和结果一致，再比较。第十一讲进一步分析时间与空间，不在这里为所有嵌套循环套统一结论。

VI · Review and Acceptance 审查与采纳

前面分别检查了规格、可读性、功能和规模。现在把这些方法用于 Lab08 的完整审查活动，并整理来源与验证记录。

I An IPv4 Specification IPv4 规格



is_valid_ipv4(s: str) → bool

- **Structure 结构**: exactly four parts separated by .
- **Characters 字符**: ASCII digits only, each value **0–255**
- **Leading zeros 前导零**: only the single digit 0 is allowed
- **Reject 排除**: signs, empty parts and any whitespace

输入为字符串；先按这份规格写测试，不给出实验完整实现。

合法地址必须按本题严格点分十进制定义解释。四段、数值范围之外，还要写清字符、前导零、符号和空白。

只考虑字符串输入。这里不提供实验的完整正确实现；学生先按规格自己写测试，随后检查候选。



I Independent Test Cases 独立用例

Three valid cases: 0.0.0.0 · 192.168.1.1 · 255.255.255.255

Invalid input 非法输入

Contract boundary 对应约定

""

Empty input

"1.2.3"

Part count

"256.1.1.1"

Value range

"01.2.3.4"

Leading zero

" 1.2.3.4"

ASCII only · full-width digit

"1.2.3.4 "

No whitespace · trailing space

| isdigit() 和 int() 的接受范围不自动等于本题规格。

54

第 54 页 · Independent Test Cases 独立用例

用例的预期来自规格，不来自候选的判断顺序。至少三个合法和五个非法，覆盖段数、范围、前导零、字符范围和空白。

isdigit 接受部分非 ASCII 数字，int 也可能接受前导符号或空白；它们的接受范围不自动等于本题规格。阅读后还可以补针对性测试，有限用例不保证发现所有遗漏。

I An AI Usage Declaration 使用声明



- | | | |
|---|-------------------|--|
| 1 | Source 来源 | Actual tool/model or instructor candidate; unknown model → say unknown |
| 2 | Scope 范围 | Name the functions or blocks generated/adopted. |
| 3 | Changes 修改 | Record manual corrections; say if none were made. |
| 4 | Checks 验证 | Actual inputs, expected results, observations and regressions |

未使用 AI 就据实说明; “AI 保证正确”不是测试记录。

声明与实际工作相符：用了什么、采用了哪些代码、做了哪些修改和真实检查。型号不清楚就注明，教师候选也写明来源。

完全未使用 AI 可以据实填写。不能伪造聊天记录，不能把 AI 说正确或尚未执行的测试写成全部通过。

I Responsibility and Safe Use 责任与安全



- **Understand 理解**: explain each adopted block
- **Preserve 保留**: original code, counterexamples and corrections
- **Inspect 审查**: unfamiliar file, install and network operations
- **Protect 保护**: remove private information, passwords, keys and tokens

沿用现有课程政策，本页不另订处罚。

对提交内容负责，需要能说明实现、修改和检查依据。格式或注释多，不代表已经理解。

外部工具的输入不能包含密码、密钥或会话令牌。未经理解不运行陌生文件操作、安装或网络请求。本页不新增处罚标准，只按现有课程政策说明责任。

I Quiz 2 · Review and Verification



扫码作答 · 五题

- | | |
|--------|--------|
| 1 函数签名 | 2 最小反例 |
| 3 操作计数 | 4 调用帧 |
| 5 使用声明 | |

All topics have been covered. Submit once.

<https://taohuang.info/cs1602/zh/quiz/8/>

57

第 57 页 · Quiz 2 · Review and Verification

第二轮五题分别检查签名与行为、最小反例、重复查找计数、各调用的局部参数和真实声明。全部内容已讲过，不重复开场列表题。

先按规格和代码推理，再对照执行证据。不要把工具名称或格式检查当成验证完成。

I Quiz Review · Contracts and Evidence



- | | | |
|---|---------------|---|
| 1 | Q2 · contract | Annotations do not enforce sorting or input preservation. |
| 2 | Q3 · failure | [1]: candidate 0 , required 1 ; the final run was omitted. |
| 3 | Q4 · counts | This input: n^2 comparisons vs $2n$ member queries ; different units. |

源代码检查次数不能当作固定 CPU 指令数。

第二题：类型提示不自动保证排序和输入不变。第三题：单个 1 使 current 增加而 best 不变，候选返回 0，预期为 1。

第四题的 n^2 是这组输入的显式比较数，不推广成每种数据分布的精确公式；成员查询与 CPU 指令也不是相同单位。

I Quiz Review · Frames and Declaration



Q5 · select factorial(3): local n is still 3

- **Current inner call:** factorial(1) has its own $n = 1$.
- **Selected outer call:** factorial(3) retains $n = 3$.
- **Q6 · declaration:** actual source, adopted scope and real verification
- **Never invent:** tool/model names, conversations or test results

59

第 59 页 · Quiz Review · Frames and Declaration

第五题选择外层 factorial(3)，它自己的 n 仍为 3。内层 n 为 1 不覆盖外层参数。

第六题采用教师候选而没有使用在线 AI，也能据实写来源与检查；不知道型号就注明，不编造使用和结果。



I Lab 8 · A, B and C 实验三段

Part 实验段	Work 学习任务	Marks 计分
A · 40 min	Five functions + four review records	Raw 24 · weekly 60%
B · 35 min	IPv4: specification → tests → review → correction	Raw 12 · weekly 40%
C · 15 min	Ruff / recursive frames / command-line recap	Completion only

6 code cards + 3 text cards, prepared locally; server synchronization pending.

A 段有五个函数：第二大的不同值、标题大小写、重复检测、字符计数和最大相邻差，另交四份审查记录。B 段把 IPv4 分成规格与测试、候选检查、修正与回归、说明与声明四步。

C 段记录 Ruff、课堂阶乘调用帧和命令复习，不增加分数。六道代码题与三张文字卡已经在本地准备，尚未同步服务器。正式发布后使用 HW3 第八周入口；本地保存不能称为在线提交。

I Solo Assignment 1 · Independent Work



Task 任务	AI rule 要求	Submission 提交
Solo assignment 1	Level 0 · independent Hanoi implementation	5% of course grade · due 10 Nov, 23:59 GMT+8
Lab08	Level 1 · assistance with a truthful declaration	Use the week-eight HW3 entries after publication

跨周作业顶栏角标不能替代题目要求；不要让 AI 生成或改写个人作业 1。

个人作业一仍独立实现普通和相邻杆汉诺塔，占总成绩 5%，截止由统一校历读取。不能因为第八周进入 Level 1，就让 AI 生成或改写该作业代码。

HW3 顶栏按最早周显示 Level 0，具体要求仍按每周实验。这个页面说明跨周任务，不再重复整套政策。

I Data and Operations 数据与操作



- **Before acceptance:** contract, code, documentation and environment
- **Recorded evidence:** boundaries, mutation, scale and corrections

L09 · Data and operations 数据与操作

How can similar records and their operations be organized together?

62

第 62 页 · Data and Operations 数据与操作

回答开场问题：样例通过以后，还要把规格、实现、接口、边界、副作用与规模证据一起核对，再记录采用的理由。

下一讲进入类，把同类记录的数据和操作放在一起组织。下面参考区保留长表和往年练习，不计入本讲必讲时间。

AFTER CLASS

Reference 参考

参考区按需查阅常见错误、完整规范与往年练习。主线已经结束，不要求把下面内容在课堂逐页讲完。

片段明确标注依赖，讲义提供完整运行入口。往年截图不原样搬运，所有新输出来自实际程序。



| Common Exceptions 常见异常

重点区分：类型 / 值

- **TypeError**
- **ValueError**

Meaning 原因

Wrong **type** for an operation
Accepted type, invalid **value**

Example 例子

```
"3" + 1  
int("abc")
```

其他常见异常 · 逐项查阅

Meaning 原因

- **SyntaxError**
- **IndentationError**
- **NameError**
- **IndexError**
- **KeyError**
- **ZeroDivisionError**
- **AttributeError**
- **RecursionError**

Invalid **syntax**
Invalid **indentation**
Name not defined
Index out of range
Dictionary **key** not found
Division by **zero**
Attribute not available
Recursion **depth limit** reached

Example 例子

```
if True : missing :  
if body not indented  
print(total) · undefined  
[1, 2][2]  
{ }["id"]  
1 / 0  
"abc".reverse()  
f() keeps calling f()
```

阅读顺序：异常名 → 原因 → 例子；回溯中的**失败语句**另行核对。

64

先对照 `TypeError` 与 `ValueError`。字符串与整数相加是类型不匹配；`int` 可以接收字符串，但 `abc` 不能按整数文字转换，这是值的问题。两段例子均实跑确认。

下面按异常名、原因、例子逐项查阅，分别指出语法、缩进、名字、下标、键、除数、属性和递归深度。并非只凭异常名就确定根因，还要核对回溯中的失败语句。`None` 是值；合法但过深递归也可能触发 `RecursionError`。

I Naming and Documentation 命名与文档



Object 对象	Convention 约定	Example 例子
Variables / functions	snake_case	student_count, filter_positive
Constants	UPPER_CASE	MAX_RETRY
Classes	CapWords	BankAccount · introduced later
Modules / packages	Short lowercase names	utils, mypackage
Internal names	A leading underscore	_helper · convention, not access enforcement

文档补充任务、参数与结果；避免混淆名字与内置名覆盖，后续规范见 PEP 8。

命名与文档帮助阅读，不是正确性证明。前导下划线表达约定，不等于强制访问控制。

旧目录中类、继承、异常名和公共接口等条目保留在完整指南链接，相关语法后续讲次展开。

当前不为了列举名字就提前使用 class 代码。



I Layout Reference 布局规范

Code / excerpt 代码或片段

```
def f(x, y=2):  
    ...  
  
def g(x: int, y: int = 2) -> int:  
    ...  
  
# Avoid crowded statements.
```

- **Indentation:** four spaces per level
- **Continuation:** align or use a hanging indent
- **Separation:** two blank lines at top level
- **Consistency:** operator breaks; no trailing whitespace

| Annotated defaults: `y: int = 2`; unannotated defaults: `y=2`. Avoid crowded statements.

本页保留缩进、续行、顶层空行、末尾空白、注解和默认参数的排版约定。带注解默认值的等号有空格，未注解默认值和关键字实参通常没有。

单元素元组需要逗号，多行结构尾逗号、长表达式空格和行长遵循项目一致性。完整规则查 PEP 8，不把每个合法片段都当必须照抄的风格。

I Imports and Comments 导入与注释



Code / excerpt 代码或片段

```
import os
import sys

import numpy as np

from mypackage import helper
```

- **Group imports:** standard → third-party → project
- **Blank lines** separate the groups.
- **Avoid wildcard imports** and unknown shadowing.
- **Docstrings** describe interfaces; keep comments current.

Python 3 source defaults to UTF-8. Third-party/project imports here are formatting excerpts.

67

第 67 页 · Imports and Comments 导入与注释

三组导入分开，通常放在模块文档字符串之后。这里的 numpy 和 mypackage 只是格式示例，不假设已安装，也不设运行按钮。

注释解释原因，文档字符串说明用途、参数和结果。源码默认 UTF-8；引号保持一致，不把单引号或双引号中的一种当唯一正确形式。



I Formatting Tools 格式工具

Command 命令

Effect 用途

`python3 -m pip install ruff`

Install into that interpreter's environment

`ruff check file.py`

Lint under the selected rules

`ruff format --check file.py`

Check formatting without rewriting

- **Editor extension** is another route; Black is an alternative formatter.

- **Record version and rules**; do not force administrator pip.

| 没有 lint 报告，仍不能证明行为正确。

安装到正确解释器，或使用编辑器扩展。系统管理环境报错时，不改用管理员权限强制 pip；虚拟环境在第十讲系统介绍。

检查模式不修改文件，普通格式化命令会改写排版。先理解报告和操作，再决定是否使用自动修复。

I Debugging Reference 调试速查



1	Pause 暂停	Breakpoint: stop before a statement executes.
2	Inspect 查看	Variables / Watch: the selected-frame context while paused
3	Navigate 切换	Call stack: choose a call; Over / Into / Out / Continue
4	Record 记录	Minimal input, expected/actual, source and runtime; print remains useful.

69

第 69 页 · Debugging Reference 调试速查

断点、变量、Watch 和调用栈解决暂停和上下文观察问题；print 仍适合少量值或批量记录。提问时保留最小输入、源码、运行环境和预期与实际。截图应显示必要上下文，不包含私人信息。

| The Zen of Python Python 之禅



- **Explicit** is better than implicit.
- **Readability** counts.
- In the face of **ambiguity**, refuse the temptation to guess.
- If the implementation is **hard to explain**, it's a bad idea.

Tim Peters · design principles for explicitness, readability and explanation; not a correctness certificate.

四条句子摘自实际运行 `import this` 的完整输出，作者为 Tim Peters。先读加粗的明确性、可读性、歧义与解释难度，再联系本讲的规格和审查。

这些句子引导设计取舍，不是逐行判分规则，也不是正确性证书。完整输出仍在讲义保留。

I Leap-Year Tests 闰年测试



Code / excerpt 代码或片段

```
def is_leap_year(  
    year: int  
) -> bool:  
    return (  
        (year % 4 == 0  
         and year % 100 != 0)  
        or year % 400 == 0  
    )
```

Actual output 实跑输出

```
1900 False  
2000 True  
2024 True  
2100 False
```

- **Century boundary:** 1900 / 2100 are not leap years.
- **400-year rule:** 2000 is a leap year.
- **Positive Gregorian years;** all three versions remain in the notes.

71

第 71 页 · Leap-Year Tests 闰年测试

公历正整数年份按能被 4 整除但不被 100 整除，或能被 400 整除判断。实际结果：1900 否、2000 是、2024 是、2100 否。

往年三个等价版本在讲义完整保留并相互核对。世纪边界比只测一个普通闰年更有检查价值。

| Prime-Number Boundaries 素数边界



Code / excerpt 代码或片段

```
def is_prime(n: int) -> bool:
    if n < 2:
        return False
    divisor = 2
    while divisor * divisor <= n:
        if n % divisor == 0:
            return False
        divisor += 1
    return True
```

Actual output 实跑输出

```
-1 False
0 False
1 False
2 True
4 False
9 False
97 True
```

- **$n < 2 \rightarrow \text{False}$** : include negative values, 0 and 1.
- **$\text{divisor}^2 \leq n$** : keep the equality; both versions are in the notes.

Complete runnable example 完整程序

72

第 72 页 · Prime-Number Boundaries 素数边界

两个旧版本都缺少 n 小于 2 的判断，负数、零和一会因空循环误报 True。补上输入边界再比较试除。

若有因数大于平方根，其配对因数小于平方根，所以可在 divisor^2 不超过 n 的范围内检查。等号不能遗漏。使用整数乘法，不用浮点 sqrt 决定边界。

I Digit-Square Sums 数位平方和



Code / excerpt 代码或片段

```
def sum_digit_squares(  
    n: int  
) -> int:  
    if n < 10:  
        return n * n  
    smaller = sum_digit_squares(  
        n // 10)  
    return (n % 10) ** 2 + smaller
```

Actual output 实跑输出

```
0 0  
2 4  
9 81  
19 82  
29 85
```

- **Old base case** returned n , not $n*n$.
- **Input 2** exposes it; input 19 happens to pass.
- **Loop / recursive / string** versions are retained.

73

第 73 页 · Digit-Square Sums 数位平方和

旧递归基准对单个数字直接返回 n ，本应返回平方。输入 2 得到 2 而不是 4；19 的首位为 1，样例恰好没有发现这项错误。

循环、递归、字符串版本在讲义保留并检查一致。输入为非负整数，递归仍受深度限制，不使用未教生成器。

I Happy Numbers 快乐数



Code / excerpt 代码或片段

```
def is_happy(n: int) -> bool:
    seen = set()
    while n != 1:
        if n in seen:
            return False
        seen.add(n)
        n = sum_digit_squares(n)
    return True
```

Actual output 实跑输出

True
False

- **Requires sum_digit_squares** from the complete program.
- **Positive integers only**; 0 is outside this contract.
- **Seen set** detects repeated states, not their counts.

Complete runnable example 完整程序

74

第 74 页 · Happy Numbers 快乐数

先将正整数替换为数位平方和。到 1 为成功，出现已访问状态则以后重复同一过程。seen 集合只需记录是否出现，不需次数。

完整程序含辅助函数，本页静态片段不设置独立运行按钮。19 实际为 True，2 为 False；零不属于正整数定义。



| Consecutive Ones 连续的 1

Code / excerpt 代码或片段

```
if bits[i] == 1:
    j = i
    while (j < len(bits)
           and bits[j] == 1):
        j += 1
    best = max(best, j - i)
    i = j
else:
    i += 1
```

Actual output 实跑输出

```
0
0
1
3
```

- **Outer i moves to j**, skipping the scanned run.
- **Nesting alone** does not imply quadratic work.
- **Empty / zero / one / final run** are separately checked.

Complete runnable example 完整程序

75

第 75 页 · Consecutive Ones 连续的 1

往年双 while 版本在发现连续段后把 i 移到 j，跳过内层已经扫描的段。空、全零、单一和末尾长段分别得到 0、0、1、3。

虽然嵌套循环，不能只看外观就说平方工作量。它与主线漏末尾段的教学候选不同，不能把旧实现误报为同一错误。



I Continuous Increasing Runs 连续递增段

Code / excerpt 代码或片段

```
def longest_increasing_run(
    nums: list[int]
) -> int:
    if not nums:
        return 0
    current = 1
    best = 1
    for i in range(1, len(nums)):
        if nums[i] > nums[i - 1]:
            current += 1
        else:
            current = 1
        best = max(best, current)
    return best
```

Complete runnable example 完整程序

Actual output 实跑输出

```
0
3
1
1
```

- **Continuous:** do not skip positions.
- **Strictly increasing:** an equal value resets the run.
- **Empty input → 0;** ordinary and repeated values are checked.

76

第 76 页 · Continuous Increasing Runs 连续递增段

连续严格递增要求相邻比较且后项大于前项。空列表为 0，相等值结束当前段。四例结果为 0、3、1、1。

不能跳过 4 拼出 [1,3,5,7] 当连续段。往年练习的这种区别在讲义与参考页保留。



I Anagrams and Multiplicity 字符次数

Code / excerpt 代码或片段

```
def is_anagram(  
    s: str, t: str  
) -> bool:  
    left = char_counts(s)  
    right = char_counts(t)  
    return left == right
```

Actual output 实跑输出

```
True  
False  
True
```

- **Requires char_counts(s)** from the complete program.
- **Lowercase ASCII**; empty strings match.
- **set("aab") == set("abb")** loses the unequal character counts.

Complete runnable example 完整程序

77

第 77 页 · Anagrams and Multiplicity 字符次数

字母重排要求每个字符的次数相同。字典计数与 sorted 比较都能保留次数；集合会丢失重数，aab 与 abb 是反例。

三例实际为 True、False、True。输入限定小写 ASCII 字符串，空串与空串互为重排。完整依赖在讲义，不把较长片段缩字塞进运行区。



I Boolean Conditions 布尔条件

Value 对象	bool(value) 真假值	value == True 比较
False	False	False
True	True	True
2	True	False
[1]	True	False

实际运行结果：真假值不等于和 True 比较；+= / + 也不能机械互换。

表中结果来自本页完整示例实跑，False、True、2 和非空列表分别比较 bool(value) 与 value==True。2 和 [1] 都为真，但与 True 比较为假。

只有明确的布尔参数才能不加条件套用简化。列表 += 与 + 也可能改变共享状态，不能只为简短互换。

I Development and Review 开发与审查



- | | | |
|---|------------|---|
| 1 | Plan 设计 | Define meaningful state and interfaces . |
| 2 | Build 实现 | Implement and test small units , not a large opaque block. |
| 3 | Review 核对 | Reuse results and check unfamiliar operations. |
| 4 | Improve 修正 | Preserve counterexamples, changes and regressions . |

函数长度、嵌套层数或代码简短，都不是正确性证明。

往年可借鉴的是小任务、接口、逐步测试、资料核对和中间状态维护。它们是可执行的检查习惯，不依赖没有出处的脑力范围数字。

函数之间可以通过参数中的可变对象相互影响，不能称天然无耦合。固定函数行数、禁用 append 或 AC 都不能替代任务与实现审查。

I Review Records and Submission 审查记录



- | | | |
|---|---------------|--|
| 1 | Contract 规格 | The task and the permitted inputs/results |
| 2 | Source 来源 | Original candidate, tool/model or instructor source |
| 3 | Evidence 证据 | Actual input / expected / observed records |
| 4 | Correction 修正 | Changes, regressions and a truthful declaration |

9 week-eight cards are ready locally; not deployed. Six lecture exercises remain; Solo1 stays Level 0.

80

第 80 页 · Review Records and Submission 审查记录

完整记录说明采纳依据，也使失败可以重新检查。六项理论练习和原有实验内容保留，实验已按 A/B/C 组织并补齐本地代码题与文字卡。

本轮没有部署、导入小测或改发布周次。发布前保存本地工作，发布后逐项使用 HW3 第八周入口；个人作业一仍独立完成。