

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Introduction to Computation (CS1602) · Lecture 6

Tuples, Dicts, Strings and Sets

Instructor: Tao Huang

Part II: Built-in Data Structures · Fall 2026

Where Lecture 5 Ended

- A list stores references; `b = a` does not copy it
- Lists are mutable; integers and strings are immutable
- `copy()` copies one level; nested objects may still be shared
- Comprehensions build a new list; do not change length while iterating

Campus Movie Night 校园电影夜

One event, several kinds of data.



A ticket identifies **one seat**: row and number.

A snack order contains a list that **can grow**.

随堂小测 · 订单修改

扫码作答

1 执行报错后，零食清单有没有改变?

https://taohuang.info/cs1602/zh/quiz/6/

Containers and Their Tasks 容器与任务

Learning Goals 学习目标



list	tuple	dict	set	str
Keep order	Fix a record	Find by key	Test membership	Process text
顺序与增删	固定记录与引用	键值设计与查找	成员判断与名单	拆分与格式化

PART 1 OF 6

Tuples

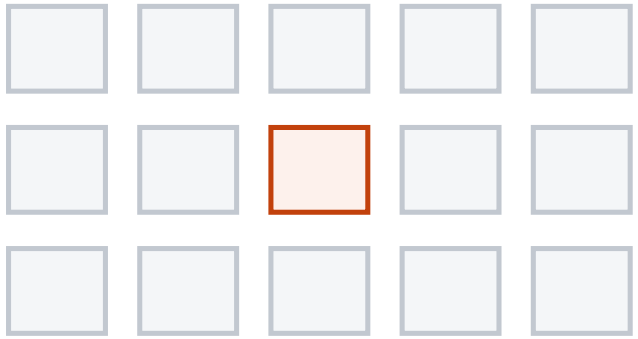
A list can grow and change.

A coordinate needs fixed positions: x and y.

A Fixed Sequence（元组）

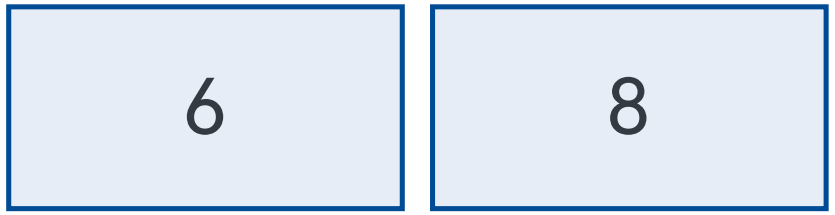
A seat record keeps **row and number together**.

Cinema seat



Row 6, seat 8

seat = (6, 8)



row, number = seat

list 也能保存坐标；tuple 用来限制这条记录的原地修改。

Item Assignment Is Not Supported

A tuple rejects **replacement of an item** at runtime.

code

```
1  seat = (6, 8)
2  seat[0] = 9
```

Actual exception 实际异常

```
TypeError: 'tuple' object does not support item assignment
```

> Construct a new tuple when different contents are needed.

The Comma Makes the Tuple

The **comma** creates a tuple; parentheses can group an expression.

code

```
1  a = (1)
2  b = (1,)
3  c = 1, 2, 3
4  print(type(a).__name__)
5  print(type(b).__name__)
6  print(c)
7  print(tuple())
```

Actual output 实跑输出

```
int
tuple
(1, 2, 3)
()
```

单元素元组需要逗号：(1,)。(1) 仍是整数。

Packing and Unpacking（打包与解包）

Packing groups values; unpacking assigns them to **matching names**.

code

```
1  point = 3, 4
2  x, y = point
3  print(x, y)
4  (a, b), c = ((1, 2), 3)
5  print(a, b, c)
6  first, *rest = (1, 2, 3, 4)
7  print(first, rest)
```

Actual output 实跑输出

```
3 4
1 2 3
1 [2, 3, 4]
```

Nested structures must match. A starred target collects a list of remaining items.

Unpacking Needs Matching Lengths

The number of values must match the receiving names.

code

```
1  a, b = (1, 2, 3)
```

Actual exception 实际异常

```
ValueError: too many values to unpack (expected 2)
```

不匹配会报 ValueError。带星号的名字收集剩余值，得到 list。

Returning a Record

A function can return **one tuple** containing several related results.

code

```python
def bounds(nums: list[int]) -> tuple[int, int]:
    return min(nums), max(nums)

result = bounds([3, 9, 5])
print(result)
low, high = result
print(low, high)
```

Actual output 实跑输出

```
(3, 9)
3 9
```

Keep the returned tuple as one record or unpack it into two names.

Fixed Slots, Mutable Contents

The tuple keeps the same references; its **snack list can still change**.

code

```
1  order = ((6, 8), ["cola"])
2  order[1].append("popcorn")
3  print(order)
```

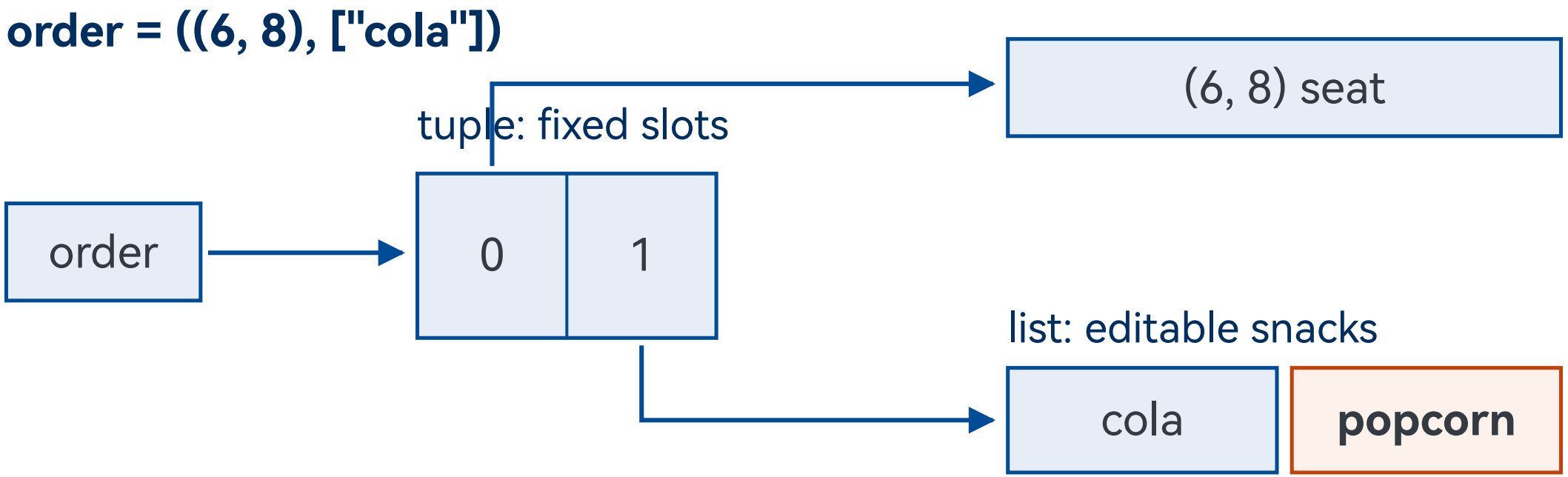
Actual output 实跑输出

```
((6, 8), ['cola', 'popcorn'])
```

append 修改零食列表，不会替换 order 中指向列表的引用。

Two Levels of References

Follow the reference to identify **which object changes**.



order[1].append("popcorn")：列表改变，元组仍指向同一个列表。

List + and +=

+: new list

```
a = ["cola"]
b = a
a = a + ["popcorn"]
print(a)
print(b)
print(a is b)
```

output

```
['cola', 'popcorn']
['cola']
False
```

+=: same list

```
a = ["cola"]
b = a
a += ["popcorn"]
print(a)
print(b)
print(a is b)
```

output

```
['cola', 'popcorn']
['cola', 'popcorn']
True
```

列表的 + 生成新列表；+= 扩展原列表，别名也会看到变化。

Opening Quiz Review 开场题回顾

Q1 · TypeError; the snack list has changed.

Opening Q1 开场原题

```
order = ((6, 8), ["cola"])
order[1] += ["popcorn"]
```

Actual exception 实际异常

```
TypeError: 'tuple' object does not support item assignment
```

After exception 异常后状态

```
((6, 8), ['cola', 'popcorn'])
```

1 · Read 取得列表

Get the list referenced by `order[1]`.

2 · Extend 原地扩展

List += adds `"popcorn"` to that list.

3 · Assign 尝试写回

Tuple slot assignment raises `TypeError`.

报错不会自动撤销之前已经完成的修改。元组的两个引用仍然没有被替换。

A Separate Snack Snapshot

Choose a **separate immutable record** when later edits must not affect it.

code

```
1  snacks = ["cola"]
2  order = ((6, 8), tuple(snacks))
3  snacks.append("popcorn")
4  print(order)
5  print(snacks)
```

Actual output 实跑输出

```
((6, 8), ('cola',))
['cola', 'popcorn']
```

本例元素是字符串。tuple() 只收集这一层的引用，不会递归冻结任意嵌套对象。

PART 2 OF 6

Dictionaries

A tuple fixes the positions in one record.

For many records, look up by a meaningful key, not a row number.

Finding an Order 按票号查订单

The question is **which order matches this ticket?**



A list search checks records **one by one**.

A dictionary maps **ticket ID → order**.

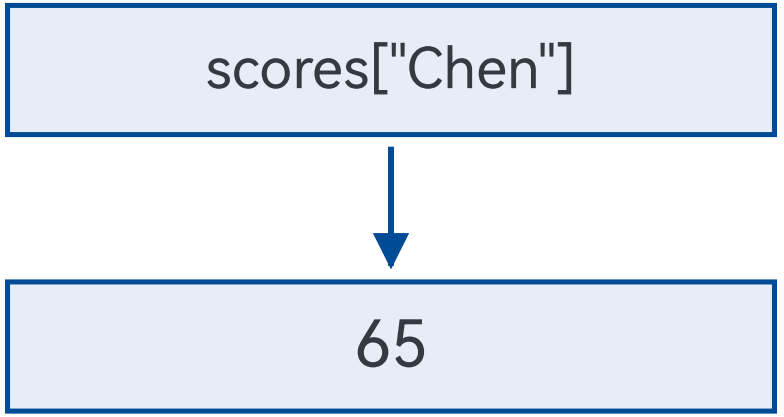
By Position or by Key

LIST: compare one by one



目标越靠后，需要比较的记录越多。

DICT: look up a key



平均查找很快，但可能发生哈希碰撞。

这里只画用法，不画实际内存布局。

Scanning Order Records

A list stores order; finding a key may require **scanning every record**.

code

```
orders = [(17, "cola"), (10, "tea"),
          (25, "popcorn")]
checks = 0
for ticket, snack in orders:
    checks += 1
    if ticket == 25:
        print(snack, checks)
        break
```

Actual output 实跑输出

```
popcorn 3
```

Dictionary Lookup 字典查找机制

key → hash → candidate slot → **key check** → value

- **Hash 哈希值**: compute an integer from the key.
- **Index 索引**: use it to choose a candidate slot.
- **Compare 比较**: verify the stored key before returning its value.
- **Probe 探查**: try another slot if different keys collide.

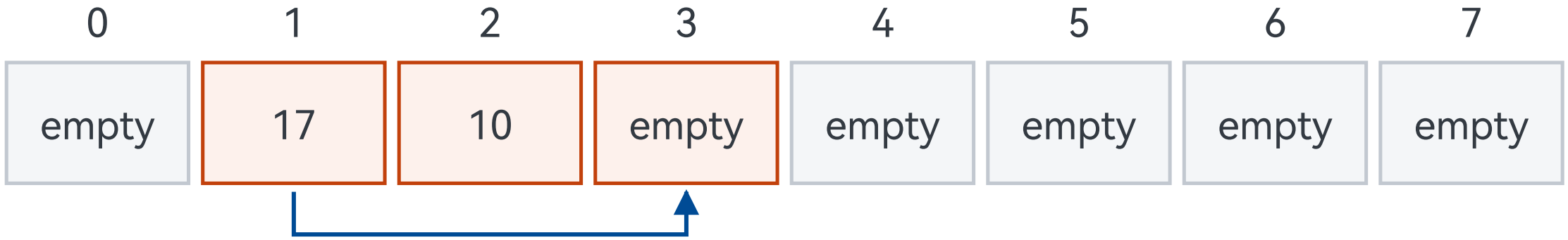
> 相同的键必须能再次沿查找规则找到；碰撞时不能覆盖另一个键。

A Collision 碰撞与插入

Different keys can choose **the same initial slot**.

key 25 → home = 25 % 8 = 1

Teaching model 教学模型：modulo + linear probing



17 ≠ 25 → 10 ≠ 25 → empty: insert 25 here

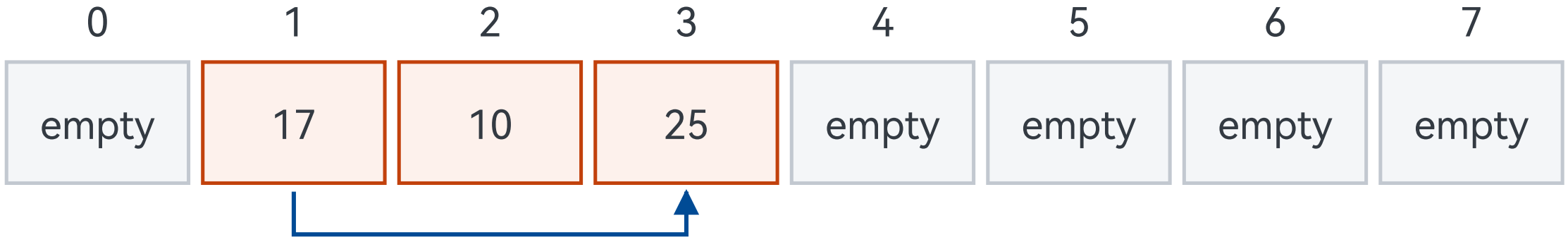
模型仅用来解释机制；CPython 的哈希与探查规则更复杂。

Lookup after a Collision 碰撞后的查询

A lookup follows the **same probe rule** used for insertion.

key 25 → home = 25 % 8 = 1

Teaching model 教学模型：modulo + linear probing



17 ≠ 25 → 10 ≠ 25 → 25: found

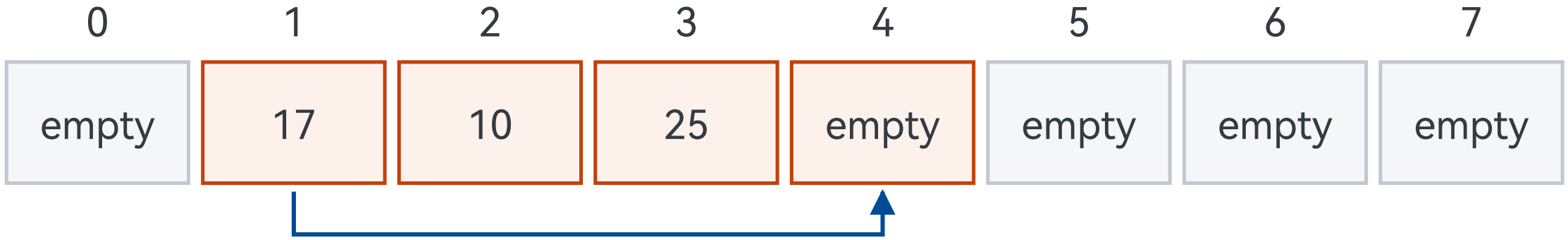
模型仅用来解释机制；CPython 的哈希与探查规则更复杂。

An Unsuccessful Lookup 查找失败

In this model, a **never-used empty slot** ends an unsuccessful search.

key 33 → home = 33 % 8 = 1

Teaching model 教学模型：modulo + linear probing



17 ≠ 33 → 10 ≠ 33 → 25 ≠ 33 → empty: absent

模型仅用来解释机制；CPython 的哈希与探查规则更复杂。

Growth and Reindexing 扩容与重建索引

More empty slots reduce crowding; growing the table requires **rebuilding its index**.

8 slots; key 25 → home 1



rebuild index 重新分配位置

16 slots; key 25 → home 9

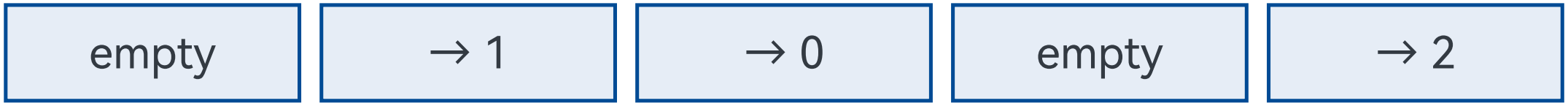


平均查找通常高效；碰撞和扩容都需要额外工作。

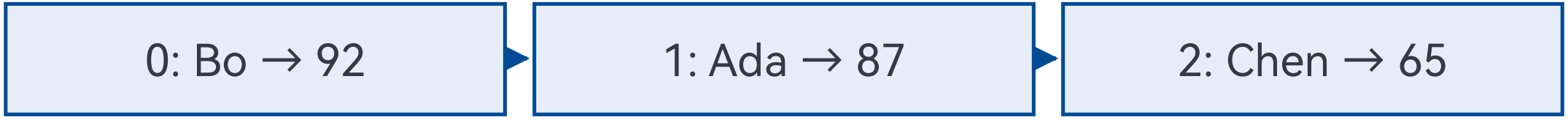
Lookup and Insertion Order 查找与插入次序

Fast lookup and ordered iteration can use **different parts of the representation**.

Lookup index 查找索引



Entries in insertion order 插入次序



查找依据键；遍历依据插入次序。两种职责可以分开实现。

概念图：不表示所有 Python 实现或所有字典的精确内存布局。

Keys and Values（键与值）

A dictionary maps a **key** to a value; `in` checks keys.

code

```
1  scores = {"Ada": 87, "Bo": 92}
2  print(scores["Ada"])
3  print(len(scores))
4  print("Ada" in scores)
5  print(87 in scores)
```

Actual output 实跑输出

```
87
2
True
False
```

> Searching values requires a separate traversal or a reverse mapping.

Nested Dictionaries（嵌套字典）

Each lookup selects one level: **student, then subject**.

code

```
results = {"Ada": {"Math": 87, "Physics": 92}}
print(results["Ada"])
print(results["Ada"]["Math"])
results["Ada"]["Math"] = 90
print(results["Ada"]["Math"])
```

Actual output 实跑输出

```
{'Math': 87, 'Physics': 92}
87
90
```

Two brackets mean two lookups. Updating one subject preserves the other subjects.

KeyError

A missing key raises `KeyError`; a missing record may indicate a data error.

code

```
1  scores = {"Ada": 87}
2  print(scores["Chen"])
```

Actual exception 实际异常

```
KeyError: 'Chen'
```

> Choose whether absence means invalid data or an expected case.

get() Does Not Insert

`get()` returns a fallback for a missing key **without inserting it**.

code

```
1  scores = {"Ada": 87}
2  print(scores.get("Chen"))
3  print(scores.get("Chen", 0))
4  print(scores)
```

Actual output 实跑输出

```
None
0
{'Ada': 87}
```

查询成绩时，“缺记录”与“考了零分”需要区分；计数从 0 开始则很合理。

setdefault() Can Insert

`setdefault()` inserts only when the key is missing, then returns the value.

code

```
1  scores = {"Ada": 87}
2  print(scores.setdefault("Chen", 0))
3  print(scores)
4  print(scores.setdefault("Ada", 100))
5  print(scores["Ada"])
```

Actual output 实跑输出

```
0
{'Ada': 87, 'Chen': 0}
87
87
```

> If the key exists, its old value is returned and left unchanged.

Adding, Updating and Removing

Assignment inserts or updates; `pop()` removes **and returns** a value.

code

```
1  scores = {"Ada": 87, "Bo": 92}
2  scores["Chen"] = 65
3  scores["Ada"] = 90
4  del scores["Bo"]
5  print(scores)
6  print(scores.pop("Chen"))
7  print(scores)
```

Actual output 实跑输出

```
{'Ada': 90, 'Chen': 65}
65
{'Ada': 90}
```

A new dictionary key inserts an entry; an out-of-range list index raises IndexError.

Insertion Order（插入顺序）

Iteration follows **insertion order**, not sorted key order.

code

```
1  d = {"Bo": 92, "Ada": 87}
2  d["Bo"] = 95
3  d["Chen"] = 65
4  print(list(d))
5  print(d.popitem())
6  print(list(d))
```

Actual output 实跑输出

```
['Bo', 'Ada', 'Chen']
('Chen', 65)
['Bo', 'Ada']
```

更新已有键不移动位置；删除后重新插入，会排到末尾。

Keys, Values and Items

`items()` supplies `(key, value)` pairs that can be unpacked in a loop.

code

```
1  scores = {"Ada": 87, "Bo": 92}
2  print(list(scores.keys()))
3  print(list(scores.values()))
4  print(list(scores.items()))
5  for name, mark in scores.items():
6      print(name, mark)
```

Actual output 实跑输出

```
['Ada', 'Bo']
[87, 92]
[('Ada', 87), ('Bo', 92)]
Ada 87
Bo 92
```

> These methods return views. `list()` is used here only to display their contents.

Changing Size During Iteration

Adding or deleting keys changes the structure being traversed.

code

```
scores = {"Ada": 87, "Bo": 92}
for name in scores:
    if scores[name] < 90:
        del scores[name]
```

Actual exception 实际异常

```
RuntimeError: dictionary changed size during iteration
```

遍历时不要增删键。更新已有键的值与改变键集合是两种操作。

Keep the Iteration Separate

Collect first, delete second: traverse a separate list of keys.

code

```
1  scores = {"Ada": 87, "Bo": 92, "Chen": 65}
2  remove = [k for k, v in scores.items() if v < 90]
3  for name in remove:
4      del scores[name]
5  print(scores)
```

Actual output 实跑输出

```
{'Bo': 92}
```

The second loop visits the separate removal list while changing the dictionary.

Hashable Keys（可哈希的键）

Lookup needs a **stable hash**; equal keys must have equal hashes.

code

```
1  d = {"Ada": 87, (3, 4): "point", 42: "answer"}
2  print(d[(3, 4)])
3  print(hash((3, 4)) == hash((3, 4)))
```

Actual output 实跑输出

```
point
True
```

数、字符串与各项均可哈希的元组可作键；列表、字典、集合不可。

An Unhashable Tuple

A tuple is hashable **only if every element is hashable**.

code

```
1  key = ([1, 2], 3)
2  d = {key: "value"}
```

Actual exception 实际异常

```
TypeError: unhashable type: 'list'
```

外层不可变仍不够：含有 list 的 tuple 不能作字典键。

Designing Keys and Values 键值设计

Requirement 需求	Representation 表示	Consequence 结果
Find one order	ticket ID → order	Unique ID avoids repeated names 同名不冲突
Keep all orders	student ID → list of orders	Appending preserves history 保存多次记录
Find by seat	(row, number) → order	Tuple combines a composite key 组合键
Keep latest only	student ID → latest order	Assignment deliberately replaces history 更新覆盖

先决定记录的身份和保留规则，再写增删改查。

PART 3 OF 6

Processing Text

Dictionaries can store structured records.

Input often arrives as text: split fields, clean them, join output.

split(): Text into Fields

`split()` turns one text record into **a list of string fields**.

code

```
1  line = "Ada,Math,87"
2  parts = line.split(",")
3  print(parts)
4  print(parts[0], int(parts[2]))
```

Actual output 实跑输出

```
['Ada', 'Math', '87']
Ada 87
```

> Every field is still a string. Convert the score explicitly with `int()`.

Whitespace or One Space

`split()` collapses whitespace; `split(" ")` preserves empty fields.

code

```
1  s = "  Ada   Bo\tChen  "
2  print(s.split())
3  print(s.split(" "))
```

Actual output 实跑输出

```
['Ada', 'Bo', 'Chen']
['', '', 'Ada', '', '', 'Bo\tChen', '', '']
```

空字段也可能是有效数据，按输入格式决定切分规则。

join(): Fields into Text

The separator joins **strings** into one result, with nothing extra at the ends.

code

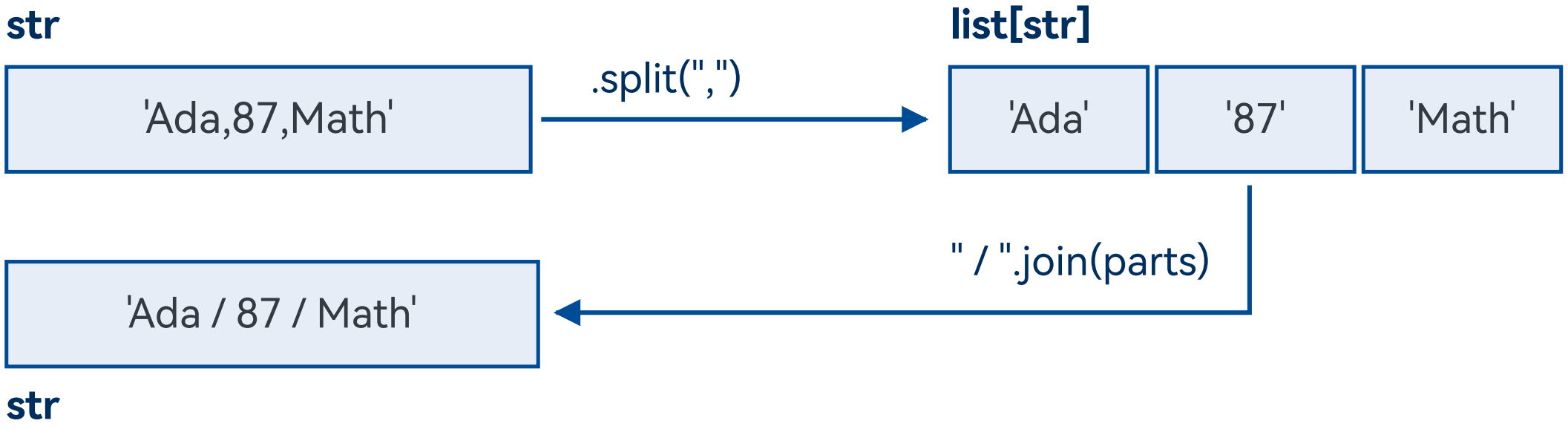
```
1  parts = ["Ada", "87", "Math"]
2  print(" / ".join(parts))
3  print("".join(parts))
4  print("\n".join(parts))
```

Actual output 实跑输出

```
Ada / 87 / Math
Ada87Math
Ada
87
Math
```

Each item must be a string. Convert numbers before joining.

Text, Fields and Text Again



split 返回列表；join 返回字符串。原字符串不变。

strip(): Only the Ends

`strip()` removes specified characters **at the ends**, leaving the middle intact.

code

```
1  s = "  Ada   Bo  \n"
2  print(repr(s.strip()))
3  print(repr(s.lstrip()))
4  print(repr(s.rstrip()))
5  print("cat!".strip(".!?"))
```

Actual output 实跑输出

```
'Ada   Bo'
'Ada   Bo  \n'
'  Ada   Bo'
cat
```

chars 参数表示一组字符，不表示必须整体匹配的前缀或后缀。

PART 4 OF 6

Dictionaries in Use

Text can now be split into words and fields.

Use a dictionary to count, group or look up the result.

Word Counts（词频统计）

Choose **word → count**; each occurrence updates one dictionary entry.

code

```
1  text = "the cat the dog the cat"
2  counts: dict[str, int] = {}
3  for word in text.split():
4      counts[word] = counts.get(word, 0) + 1
5  print(counts)
```

Actual output 实跑输出

```
{'the': 3, 'cat': 2, 'dog': 1}
```

> The first occurrence starts from 0; subsequent occurrences retrieve and increment the old count.

Sorting with a Function

key receives a function; `sorted()` compares **the values it returns**.

code

```
1  counts = {"the": 3, "cat": 2, "dog": 1}
2  def count_of(pair: tuple[str, int]) -> int:
3      return pair[1]
4
5  top = sorted(counts.items(), key=count_of, reverse=True)
6  print(top[:2])
```

Actual output 实跑输出

```
[('the', 3), ('cat', 2)]
```

`key=count_of` passes the function object without calling it at this point.

Tuple Comparison 元组比较

Tuple comparison checks **the first differing pair of elements**.

code

```
1  print((2, "cat") < (3, "ant"))
2  print((2, "cat") < (2, "dog"))
```

Actual output 实跑输出

```
True
True
```

排序键先放主要依据，再放并列时使用的次要依据。

Two Sorting Priorities

A tuple key compares **count first, word second**.

code

```
pairs = [("dog", 2), ("cat", 2), ("the", 3)]
def order(pair: tuple[str, int]) -> tuple[int, str]:
    return -pair[1], pair[0]

print(sorted(pairs, key=order))
```

Actual output 实跑输出

```
[('the', 3), ('cat', 2), ('dog', 2)]
```

先比较负次数，再比较单词；次数相同的词按字典序排列。

A Lookup Table

A table represents a fixed input–output mapping without a chain of branches.

code

```python
days = {1: 31, 2: 28, 3: 31, 4: 30}
def month_days(month: int) -> int:
    return days[month]

print(month_days(2), month_days(4))
```

Actual output 实跑输出

```
28 30
```

> Specify the supported input range and decide how missing inputs should be handled.

PART 5 OF 6

Sets

A dictionary records a key and its value.

Sometimes only membership matters: present or absent.

Repeated Scans 重复扫码

Admission asks **“has this ticket appeared before?”**



Keep ticket IDs in a **set** when only membership matters.

Keep a count or event log when **repetitions matter**.

Membership at the Gate

A set remembers **which ticket IDs have appeared**, not how many times.

code

```
seen = set()
for ticket in [17, 25, 17, 10]:
    if ticket in seen:
        print(ticket, "repeat")
    else:
        seen.add(ticket)
        print(ticket, "admit")
print(sorted(seen))
```

Actual output 实跑输出

```
17 admit
25 admit
17 repeat
10 admit
[10, 17, 25]
```

Set Lookup and Its Trade-offs 集合机制与取舍

- Hash the member, inspect a candidate slot, **compare**, then probe on collisions.
- Adding an equal member leaves **one member**, not two copies.
- Typical average membership lookup is fast; **extra storage** supports the index.
- No positional access, no guaranteed order, and **no occurrence counts**.

> set 的优势来自面向成员查找的组织方式。不要把它当作顺序表。

Distinct Members（集合）

A set stores **distinct members**, with no positional order guaranteed.

code

```
1  nums = [3, 1, 3, 2, 1]
2  s = set(nums)
3  print(sorted(s))
4  print(len(s), 2 in s)
5  print(set("banana") == {"b", "a", "n"})
```

Actual output 实跑输出

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

`sorted()` makes this output stable. It does not give the set an internal positional order.

No Positional Index

A set answers **membership**, not “which item is at position zero?”.

code

```
1  s = {1, 2, 3}
2  print(s[0])
```

Actual exception 实际异常

```
TypeError: 'set' object is not subscriptable
```

需要展示顺序时，明确排序规则，或另外保存 list。

Hashable Members

Set members follow the same **hashability requirement** as dictionary keys.

code

```
1  s = {(1, 2), "Ada", 3}
2  print((1, 2) in s)
3  s.add([4, 5])
```

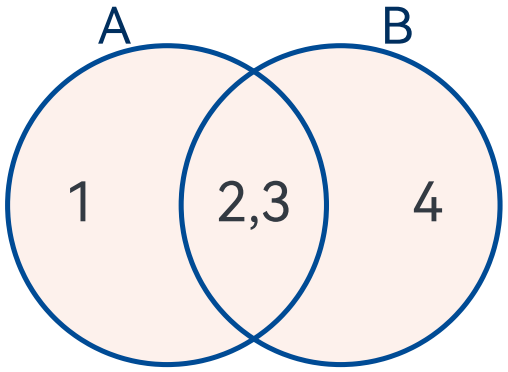
Actual exception 实际异常

```
TypeError: unhashable type: 'list'
```

set 本身不可哈希；需要固定成员集合时可以使用 frozenset。

Four Set Operations（集合运算）

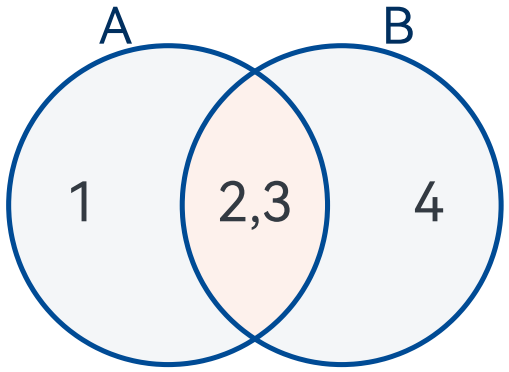
A | B



union

{1, 2, 3, 4}

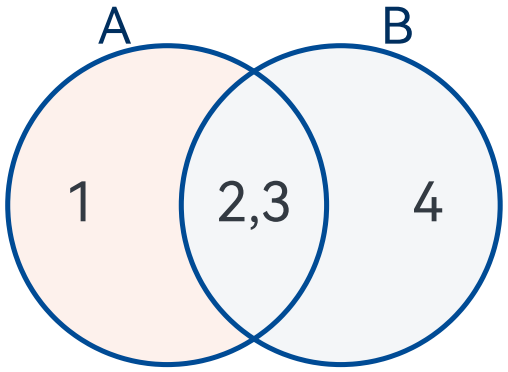
A & B



intersection

{2, 3}

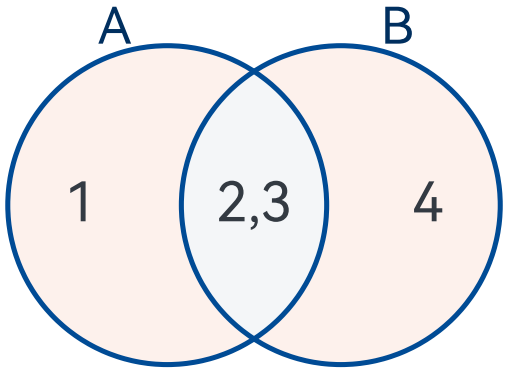
A - B



difference

{1}

A ^ B



symmetric difference

{1, 4}

Two Screenings 两场电影的观众

Choose the relation before choosing the operator.

Question 问题	Operation 运算	Meaning 含义
Attended both	A & B	Intersection 交集
Attended either	A | B	Union 并集
Only the first	A - B	Difference 差集
Exactly one screening	A ^ B	Symmetric difference 对称差

Operations on Course Lists

Set operations describe **relationships between two groups**.

code

```
a = {"Math", "Physics", "Python"}
b = {"Physics", "Art", "Python"}
print(sorted(a & b))
print(sorted(a - b))
print(sorted(a | b))
print(sorted(a ^ b))
```

Actual output 实跑输出

```
['Physics', 'Python']
['Math']
['Art', 'Math', 'Physics', 'Python']
['Art', 'Math']
```

交集是共同课程；差集只取左边独有；对称差取两侧独有。

Deduplication with Order

A **list keeps order** while a **set checks whether a value has appeared**.

code

```
1  nums = [3, 1, 3, 5, 1, 7]
2  seen: set[int] = set()
3  out: list[int] = []
4  for x in nums:
5      if x not in seen:
6          seen.add(x)
7          out.append(x)
8  print(out)
9  print(list(dict.fromkeys(nums)))
```

Actual output 实跑输出

```
[3, 1, 5, 7]
[3, 1, 5, 7]
```

顺序由 out 保存，seen 只负责成员查询。两个容器各自承担一种任务。

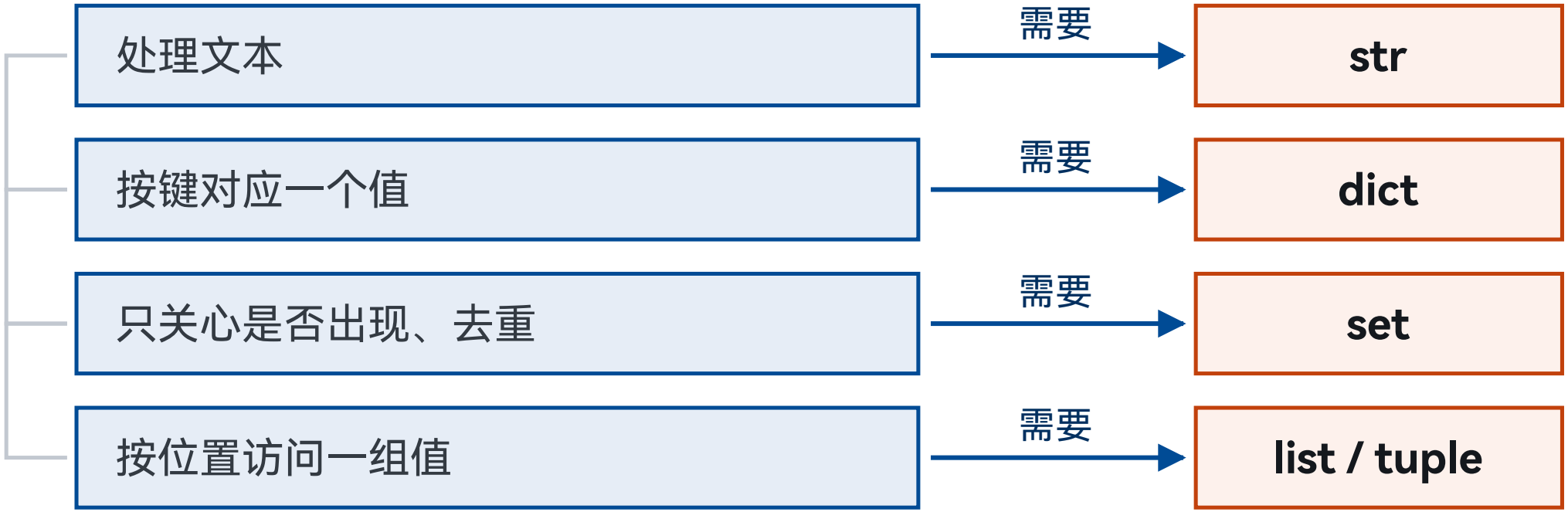
PART 6 OF 6

Choosing and Sharing Containers

Each container answers a different task.

Choose the representation, then check sharing and mutation.

A Container Decision Guide



list 可增删；tuple 的槽位固定。先按任务选择，再考虑快慢。

Container Properties 容器特点

Type	What it preserves 保存什么	What it restricts 限制
list	Positions, repeats; editable 顺序与重复	Membership search scans items 成员查找需扫描
tuple	Positions, repeats; fixed references 固定引用	No item replacement; nested objects may mutate
dict	Key → value; insertion order 对应与次序	One value per equal key; hashable keys
set	Distinct membership 不同成员	No positions, counts, or guaranteed order
str	An immutable text sequence 不可变文本	Text operations; methods do not mutate it

Operation Costs 操作代价

n = number of stored items 元素数

Operation 操作	Sequence 序列	Hash container 哈希容器
Find an item / key	list / tuple: scan up to n items	dict / set: average O(1), worst O(n)
Read by position	list / tuple: O(1)	dict uses a key; set has no index
Keep event order	list appends; keeps repetitions	dict keeps key insertion order; set does not
Build a set of n values	Reading input already takes O(n)	Index building costs time and extra space

假设哈希与相等比较代价有界。集合运算通常处理多项成员，不能一概说是 O(1)。

A Combined Representation 容器组合设计

One task can need **several containers with different responsibilities**.

- **tuple**: one seat `(row, number)`.
- **list**: scan events in arrival order, including repeats.
- **dict**: ticket ID → order; choose latest or full history.
- **set**: ticket IDs already admitted.

> 每增加一份索引，都要考虑额外空间以及数据更新是否同步。

Iteration Is Not a Snapshot

Rebinding a loop variable does not replace a container item.

code

```python
a = [1, 2, 3]
for x in a:
    x = 99
print(a)

scores = {"Ada": 87, "Bo": 92}
for name in scores:
    scores[name] += 1
print(scores)
```

Actual output 实跑输出

```
[1, 2, 3]
{'Ada': 88, 'Bo': 93}
```

更新字典中已有键的值不会改变键集合；增删键需要单独处理。

Three Minutes with Your Neighbour

按学号查询成绩，同时保留提交顺序。应该用一个容器，还是组合几种?

Explain the representation and which operations it supports.

随堂小测 · 第 2 轮

扫码作答

1. get 与 setdefault
2. split 的两种规则
3. 差集与对称差
4. 哈希碰撞后的查找
5. 重复事件与容器组合

https://taohuang.info/cs1602/zh/quiz/6/

What to Take Away

- Tuple: fixed references; nested objects may still be mutable
- Dict: key → value; insertion order; hashable keys
- String methods: no in-place mutation; different return types
- Set: membership without duplicates or guaranteed order
- Choose by the task; document sharing and changes to inputs

Lab 6 — 词频统计与容器选型

- **6-1** invert: swap keys and values; explain repeated values
- **6-2** unique_ordered: list for order, set for membership
- **6-3 / 6-4** common names and groups by first letter
- **6-5** parse text into nested dictionaries
- **6-6** top words: clean, count, sort with two priorities
- **6-7** choose containers and explain the requirements

B 段比较查找代价；C 段练习 Git 远程同步。6-7、B、C 分别有答题卡。

Next Lecture

Recursion（递归）

Solve a smaller instance of the same problem, then combine the answer.

PRACTICAL PATTERNS

Reference Pages

Grouping, shared values, paired data, text and formatting.

Complete API details remain in the lecture notes.

Dictionary Comprehension（字典推导）

A dictionary comprehension builds a **new mapping** from selected pairs.

code

```
1  scores = {"Ada": 87, "Bo": 92, "Chen": 65}
2  passed = {k: v for k, v in scores.items() if v >= 90}
3  print(passed)
4  print(scores)
```

Actual output 实跑输出

```
{'Bo': 92}
{'Ada': 87, 'Bo': 92, 'Chen': 65}
```

> The expressions before and after `:` create a key and a value. The original stays unchanged.

Grouping Orders 按名字分组

Use **key → list** when one key needs to retain several values.

code

```
orders = [("Ada", "cola"), ("Bo", "tea"),
          ("Ada", "popcorn")]
groups = {}
for name, item in orders:
    groups.setdefault(name, []).append(item)
print(groups)
```

Actual output 实跑输出

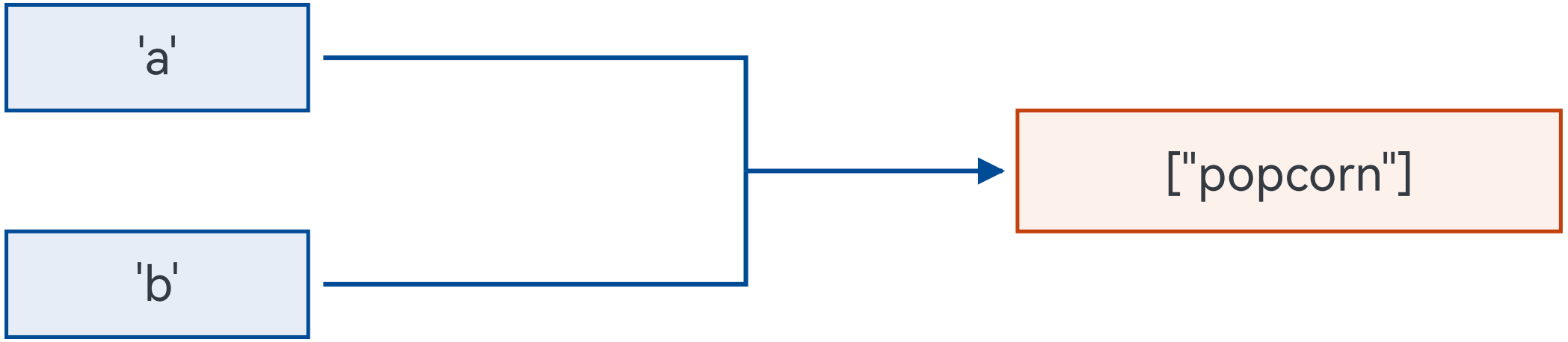
```
{'Ada': ['cola', 'popcorn'], 'Bo': ['tea']}
```

首次出现时建立列表；之后追加到已有列表，保留组内顺序与重复。

Shared Defaults 共享的初值

`fromkeys(keys, [])` assigns **one shared list** to every key.

dict.fromkeys(["a", "b"], [])



两个键引用同一个列表；通过 a 追加，b 也看得到。

Independent Default Lists

Create `[]` **inside each iteration** when groups need separate lists.

code

```
1  shared = dict.fromkeys(["a", "b"], [])
2  shared["a"].append("popcorn")
3  print(shared)
4  separate = {key: [] for key in ["a", "b"]}
5  separate["a"].append("popcorn")
6  print(separate)
```

Actual output 实跑输出

```
{'a': ['popcorn'], 'b': ['popcorn']}
{'a': ['popcorn'], 'b': []}
```

zip() and enumerate() 配对与编号

`zip()` pairs values; `enumerate()` adds an index to each pair.

code

```
1  names = ["Ada", "Bo", "Chen"]
2  marks = [87, 92]
3  pairs = list(zip(names, marks))
4  print(pairs)
5  for i, pair in enumerate(pairs, 1):
6      print(i, pair)
```

Actual output 实跑输出

```
[('Ada', 87), ('Bo', 92)]
1 ('Ada', 87)
2 ('Bo', 92)
```

zip 默认到较短输入结束；必须一一对应时，应先检查长度。

Useful String Methods 常用文本方法

Expression 写法	Result 结果	Use / boundary 用途与边界
" DOG! ".strip().lower()	'dog!'	Trim whitespace, then change case 去空白与转小写
"DOG!".strip("!?").lower()	'dog'	Remove chosen end punctuation 指定两端标点
"ticket-17".startswith("ticket-")	True	Test a prefix; return bool 判断前缀
"banana".replace("a", "o")	'bonono'	Return replacement text 返回替换结果
"-5".isdigit()	False	Classify characters, not a number 检查数字字符

方法不会修改原字符串；结果的类型决定下一步可以做什么。

Format Specification 格式说明

Read each component of **>12,.2f**.

Expression 表达式

```
value = 1234.5
print(repr(f"{value:>12,.2f}"))
```

Result · repr()

```
'    1,234.50'
```

Part 组成	Meaning 含义	Effect 效果
>	Right alignment 右对齐	Padding goes before the value
12	Minimum width 最小宽度	A longer result is not truncated
,	Grouping 千分位	1234 becomes 1,234
.2f	Fixed point with two decimals	Two digits after decimal