



上海交通大学
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

| An Opening Prediction

A: a list inside a tuple

```
t = ([1, 2], 3)
t[0].append(4)
print(t)
```

B: replace a tuple item

```
t = ([1, 2], 3)
t[0] = [9]
print(t)
```

两段都想改第 0 项。哪些能执行？先预测，讲完元组后解释。

随堂小测 · 第 1 轮



扫码作答 · 这一轮一题

1 元组内部的列表能否修改

先做预测，元组一节结束后讲评，不重复作答。

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

| Four Tasks for the Same Data

- Keep a fixed record: a coordinate or an ID–score pair
- Look up a score by name, not by position
- Compare two course lists: which courses do both groups take?
- Turn text into structured data, then format the result



OVERVIEW

Today

Lists gave us one general-purpose container.

Choose a container by the task: record, lookup, text, membership.



PART 1 OF 6

Tuples

A list can grow and change.

A coordinate needs fixed positions: x and y .

| A Fixed Sequence (元组)

code

```
1 point = (3, 4)
2 print(point[0], point[-1])
3 print(point[:1])
4 print(len(point), 4 in point)
5 print(tuple([3, 4]))
```

output

```
3 4
(3,)
2 True
(3, 4)
```

| Item Assignment Is Not Supported

code

```
1 point = (3, 4)
2 point[0] = 9
```

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    point[0] = 9
    ~~~~~^^^
TypeError: 'tuple' object does not support item assignment
```

| The Comma Makes the Tuple

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())
```

output

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

| Packing and Unpacking (打包与解包)

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)
```

output

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

| Unpacking Needs Matching Lengths

code

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

traceback

```
Traceback (most recent call last):  
  File "example.py", line 1, in <module>  
    a, b = (1, 2, 3)  
    ^^^^  
ValueError: too many values to unpack (expected 2)
```

| Returning a Record

code

```
1 def bounds(nums: list[int]) -> tuple[int, int]:  
2     return min(nums), max(nums)  
3  
4 result = bounds([3, 9, 5])  
5 print(result)  
6 low, high = result  
7 print(low, high)
```

output

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

| Fixed Slots, Mutable Contents

code

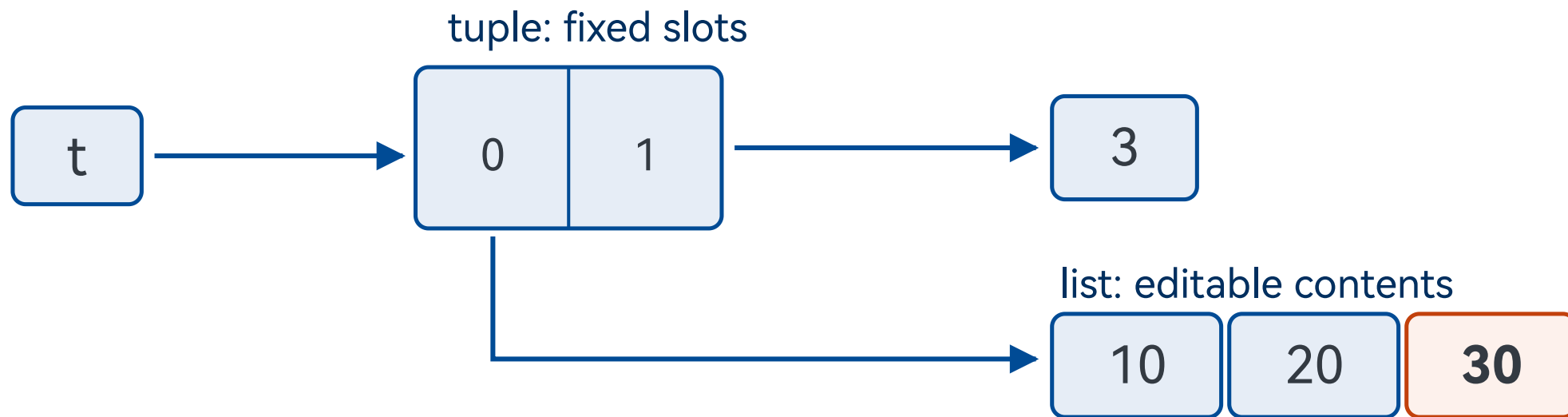
```
1 t = ([1, 2], 3)
2 t[0].append(4)
3 print(t)
```

output

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

| Two Levels of References

$t = ([10, 20], 3)$



$t[0]$ 的引用不能替换；沿这个引用找到的列表仍然可以修改。

Tuple Operations

code

```
1 t = (1, 2, 2, 3)
2 print(t.count(2), t.index(3))
3 print(t + (4,))
4 print(t * 2)
5 print((2, 9) < (3, 0))
6 print((2, 9) < (2, 10))
```

output

```
2 3
(1, 2, 2, 3, 4)
(1, 2, 2, 3, 1, 2, 2, 3)
True
True
```



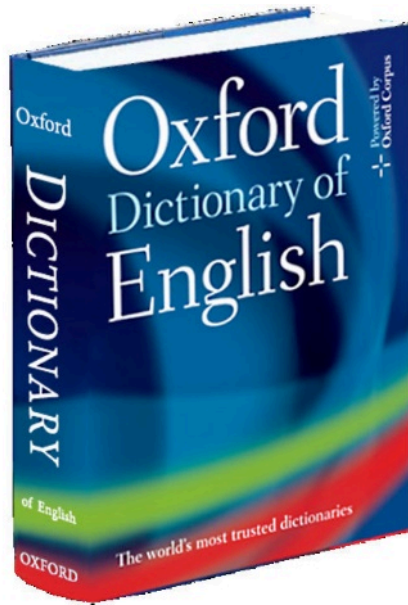
PART 2 OF 6

Dictionaries

A tuple fixes the positions in one record.

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

| From a Word to Its Meaning



A dictionary pairs a **key** with a **value**.

A word identifies its explanation; an ID identifies a record.

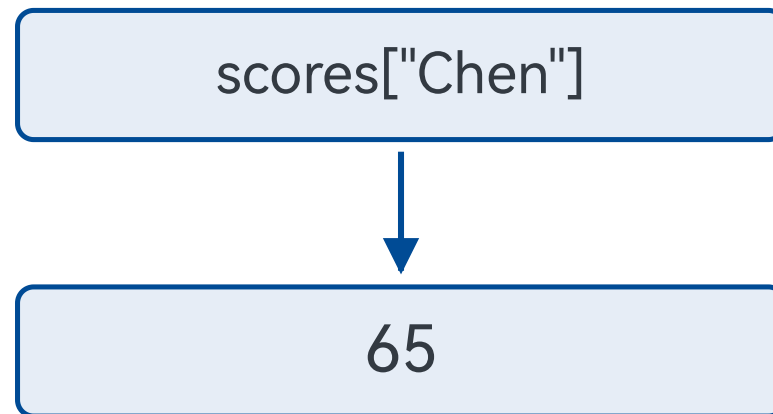
| By Position or by Key

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。

I Keys and Values (键与值)

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)
```

output

```
87
2
True
False
```

| Creating a Dictionary

code

```
1 a = {"x": 1, "y": 2}
2 b = dict(x=1, y=2)
3 c = dict([("x", 1), ("y", 2)])
4 print(a == b == c)
5 print(type({}).__name__)
6 print(type(set()).__name__)
```

output

```
True
dict
set
```

Nested Dictionaries (嵌套字典)

code

```
1 results = {"Ada": {"Math": 87, "Physics": 92}}
2 print(results["Ada"])
3 print(results["Ada"]["Math"])
4 results["Ada"]["Math"] = 90
5 print(results["Ada"]["Math"])
```

output

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

| KeyError

code

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

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    print(scores["Chen"])
          ~~~~~^~~~~~
KeyError: 'Chen'
```

I get() Does Not Insert

code

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

output

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

|.setdefault() Can Insert

code

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

output

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

| Adding, Updating and Removing

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)
```

output

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

| Insertion Order (插入顺序)

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))
```

output

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

I Keys, Values and Items

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)
```

output

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

| A View Follows Its Dictionary

code

```
1 scores = {"Ada": 87}
2 view = scores.keys()
3 snapshot = list(view)
4 scores["Bo"] = 92
5 print(list(view))
6 print(snapshot)
```

output

```
['Ada', 'Bo']
['Ada']
```

| Changing Size During Iteration

code

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

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    for name in scores:
        ^^^^^
RuntimeError: dictionary changed size during iteration
```

I Keep the Iteration Separate

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)
```

output

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

| Dictionary Comprehension (字典推导)

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)
```

output

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

| Hashable Keys (可哈希的键)

code

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

output

```
point
True
```

| An Unhashable Tuple

code

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

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    d = {key: "value"}
        ^^^^^^^^^^^^^
TypeError: unhashable type: 'list'
```

| Aliases, Copies and Merges

code

```
1  a = {"x": 1, "y": 2}
2  b = a
3  c = a.copy()
4  b["x"] = 9
5  print(a, c)
6  extra = {"y": 20, "z": 30}
7  print(a | extra)
8  print(**a, **extra)
9  print(c.update(extra))
```

output

```
{'x': 9, 'y': 2} {'x': 1, 'y': 2}
{'x': 9, 'y': 20, 'z': 30}
{'x': 9, 'y': 20, 'z': 30}
None
```



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

code

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

output

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

| Whitespace or One Space

code

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

output

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

| join(): Fields into Text

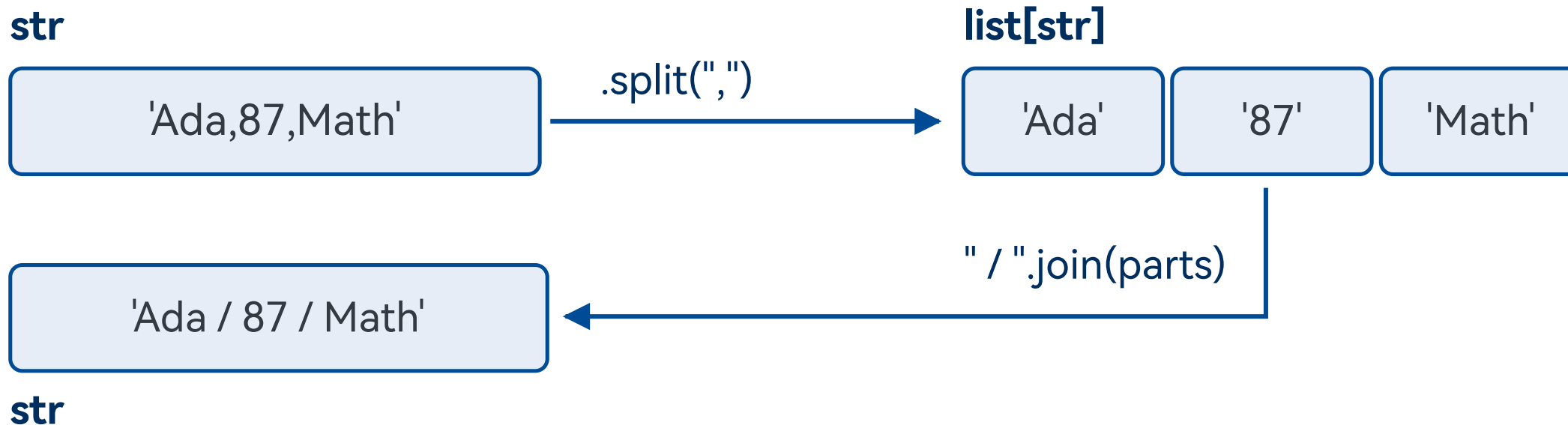
code

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

output

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

| 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

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("!.?"))
```

output

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

| Case Conversion and Prefix Tests

code

```
1 s = "Hello World"
2 print(s.upper())
3 print(s.lower())
4 print(s.title())
5 print(s.startswith("Hello"))
6 print(s.endswith("!"))
7 print(s)
```

output

```
HELLO WORLD
hello world
Hello World
True
False
Hello World
```

| Character Tests

code

```
1 for s in ["123", "12.3", "-5", "abc", ""]:
2     print(repr(s), s.isdigit(), s.isalpha())
3 print("abc123".isalnum())
```

output

```
'123' True False
'12.3' False False
'-5' False False
'abc' False True
'' False False
True
```

| find(), count() and replace()

code

```
1 s = "the quick brown fox"
2 print(s.find("quick"))
3 print(s.find("cat"))
4 print(s.count("o"))
5 print(s.replace("quick", "slow"))
6 print(s)
```

output

```
4
-1
2
the slow brown fox
the quick brown fox
```

I Types of Method Results

code

```
1 s = "Ada Bo"
2 print(type(s.split()).__name__)
3 print(type(s.find("Bo")).__name__)
4 print(type(s.isalpha().__name__)
5 print(type(s.lower().__name__)
```

output

```
list
int
bool
str
```

| Formatting beyond Decimal Places

code

```
1 n = 1234567.891
2 print(f"{n:,.2f}")
3 print(f"{0.856:.1%}")
4 print(f"{42:*^11}")
5 print(f"{True!s:<6}!")
```

output

```
1,234,567.89
85.6%
****42*****
True !
```



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)

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)
```

output

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

| Sorting with a Function

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])
```

output

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

| Two Sorting Priorities

code

```
1 pairs = [("dog", 2), ("cat", 2), ("the", 3)]
2 def order(pair: tuple[str, int]) -> tuple[int, str]:
3     return -pair[1], pair[0]
4
5 print(sorted(pairs, key=order))
```

output

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

| Grouping and Counter

code

```
1 groups: dict[str, list[str]] = {}
2 for word in ["apple", "avocado", "banana"]:
3     groups.setdefault(word[0], []).append(word)
4 print(groups)
5 from collections import Counter
6 print(Counter("the cat the cat".split()).most_common(2))
```

output

```
{'a': ['apple', 'avocado'], 'b': ['banana']}
[('the', 2), ('cat', 2)]
```

| A Lookup Table

code

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

output

```
28 30
```



PART 5 OF 6

Sets

A dictionary records a key and its value.

Sometimes only membership matters: present or absent.

| Distinct Members (集合)

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"})
```

output

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

| No Positional Index

code

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

traceback

```
Traceback (most recent call last):
  File "example.py", line 2, in <module>
    print(s[0])
          ~^^^
TypeError: 'set' object is not subscriptable
```

| Adding and Removing Members

code

```
1 s = {1, 2}
2 s.add(3)
3 s.update([3, 4])
4 s.remove(2)
5 s.discard(99)
6 print(sorted(s))
7 print(s.add(5))
8 s.clear()
9 print(len(s))
```

output

```
[1, 3, 4]
None
0
```

| Hashable Members

code

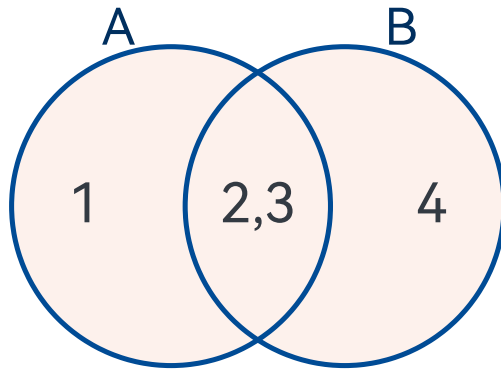
```
1 s = {(1, 2), "Ada", 3}
2 print((1, 2) in s)
3 s.add([4, 5])
```

traceback

```
Traceback (most recent call last):
  File "example.py", line 3, in <module>
    s.add([4, 5])
TypeError: unhashable type: 'list'
```

Four Set Operations (集合运算)

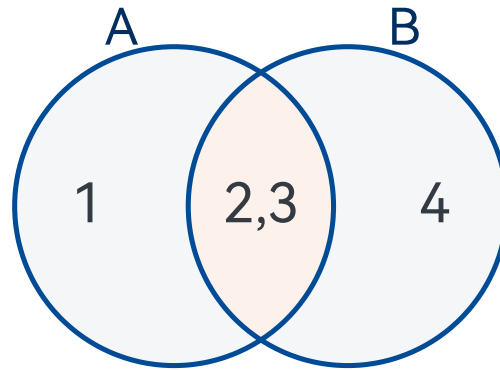
$A \cup B$



union

$\{1, 2, 3, 4\}$

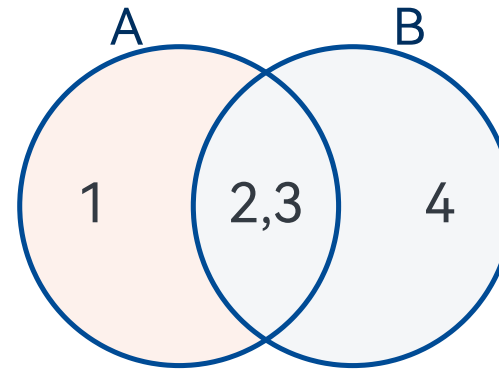
$A \cap B$



intersection

$\{2, 3\}$

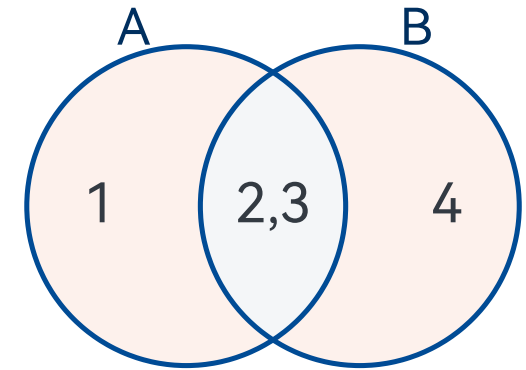
$A - B$



difference

$\{1\}$

$A \oplus B$



symmetric
difference

$\{1, 4\}$

| Operations on Course Lists

code

```
1 a = {"Math", "Physics", "Python"}
2 b = {"Physics", "Art", "Python"}
3 print(sorted(a & b))
4 print(sorted(a - b))
5 print(sorted(a | b))
6 print(sorted(a ^ b))
```

output

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



| Subset and Superset (子集与超集)

code

```
1 a = {1, 2}
2 b = {1, 2, 3}
3 print(a <= b, a < b)
4 print(a <= a, a < a)
5 print(b >= a)
6 print(a.isdisjoint({4, 5}))
```

output

```
True True
True False
True
True
```

I Deduplication with Order

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)))
```

output

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

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。

| Empty Containers Are False

code

```
1 for x in [], (), "", {}, set(), [0], "0":  
2     print(repr(x), bool(x))
```

output

```
[] False  
() False  
'' False  
{ } False  
set() False  
[0] True  
'0' True
```

| List + and +=

+: new list and rebind

```
a = [1, 2]
b = a
a = a + [3]
print(a, b)
print(a is b)
```

output

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

+=: extend this list

```
a = [1, 2]
b = a
a += [3]
print(a, b)
print(a is b)
```

output

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

列表上的 += 会修改原对象，不能简单当成 $a = a + b$ 的缩写。

Changing a Parameter

code

```
1  def add(items: list[int]) -> None:
2      items.append(3)
3  a = [1, 2]
4  add(a)
5  print(a)
6
7  def with_item(items: list[int]) -> list[int]:
8      return items + [4]
9  print(with_item(a))
10 print(a)
```

output

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

| Iteration Is Not a Snapshot

code

```
1  a = [1, 2, 3]
2  for x in a:
3      x = 99
4  print(a)
5
6  scores = {"Ada": 87, "Bo": 92}
7  for name in scores:
8      scores[name] += 1
9  print(scores)
```

output

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

| zip(): Parallel Iteration

code

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

output

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

| 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.

I 随堂小测 · 第 2 轮



扫码作答 · 这一轮五题

- 1 get 与 setdefault
- 2 split 的两种规则
- 3 差集与对称差
- 4 列表 += 的影响
- 5 zip 到哪里结束

字典、文本、集合、修改与配对各一题；答完再公布。

<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。

Recursion (递归)

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



FOR LOOKUP AFTER CLASS

Reference Pages

Further examples and syntax from the previous course materials.

| Rebinding and Deleting a Name

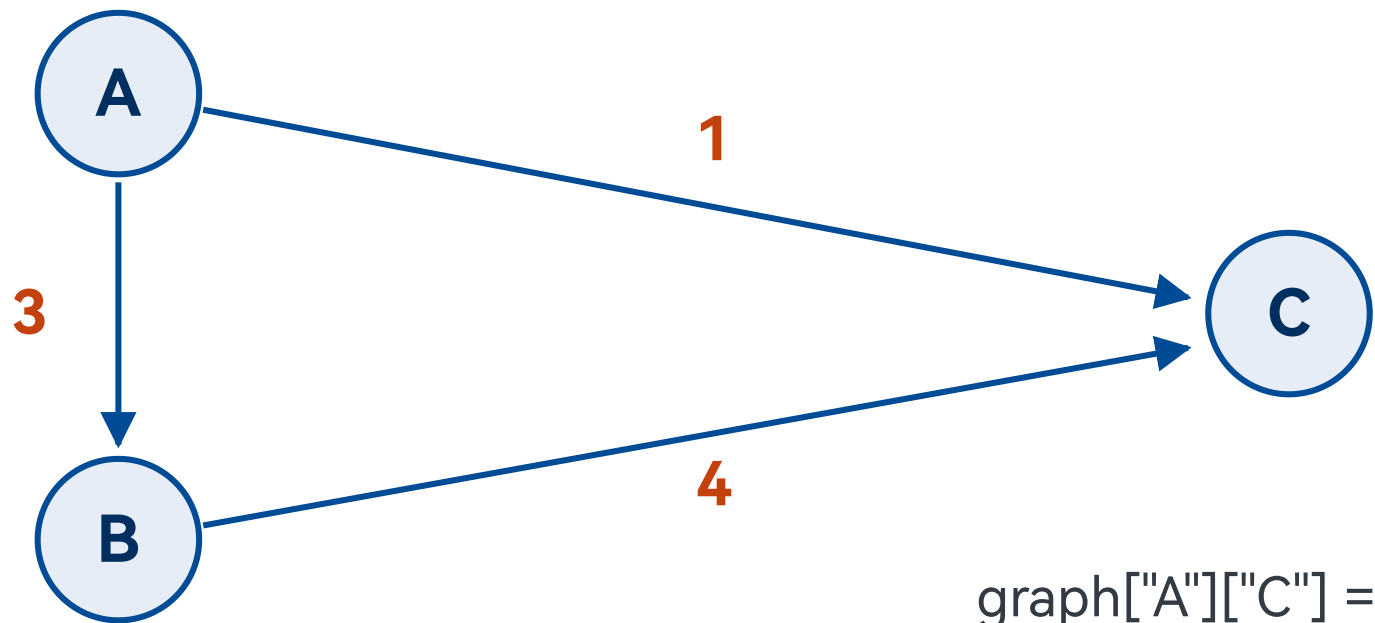
code

```
1 t = (1, 2)
2 old = t
3 t = t + (3,)
4 print(t, old)
5 del t
6 print(old)
```

output

```
(1, 2, 3) (1, 2)
(1, 2)
```

| A Graph as Nested Dictionaries



`graph["A"]["C"] = 1`

外层键选择出发点，内层键选择终点，值记录边的权重。

| Looking Up an Edge Weight

code

```
1 graph = {  
2     "A": {"B": 3, "C": 1},  
3     "B": {"C": 4},  
4     "C": {},  
5 }  
6 print(graph["A"]["C"])  
7 print(graph["B"]["C"])
```

output

```
1  
4
```

| Multi-Line Statements

code

```
1  scores = {  
2      "Ada": 87,  
3      "Bo": 92,  
4  }  
5  total = (  
6      scores["Ada"]  
7      + scores["Bo"]  
8  )  
9  print(total)
```

output

179

| Tabs and expandtabs()

code

```
1 print(repr("a\tb".expandtabs(4)))  
2 print(repr("abc\tb".expandtabs(4)))  
3 print(repr("a\tb".expandtabs(8)))
```

output

```
'a    b'  
'abc  b'  
'a          b'
```

| Constants in the string Module

code

```
1 import string
2 print(string.ascii_lowercase)
3 print(string.ascii_uppercase)
4 print(string.digits)
5 print(string.hexdigits)
6 print(repr(string.whitespace))
```

output

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
0123456789abcdefABCDEF
' \t\n\r\x0b\x0c '
```

| Format Types

code

```
1 print(f"{42:d} {42:b} {42:o} {42:x}")
2 print(f"{65:c}")
3 print(f"{1234.5:.2e}")
4 print(f"{1234.5:.4g}")
5 print(f"{42:+06d}")
6 print(f"{1234567:_}")
```

output

```
42 101010 52 2a
A
1.23e+03
1234
+00042
1_234_567
```

| Other Formatting Interfaces

code

```
1 name, mark = "Ada", 87.125
2 print(f"{name}: {mark:.2f}")
3 print("{}: {:.2f}".format(name, mark))
4 print("%s: %.2f" % (name, mark))
5 print(format(mark, ".2f"))
```

output

```
Ada: 87.12
Ada: 87.12
Ada: 87.12
87.12
```

Dictionary Method Reference

code

```
1 d = {"Ada": 87, "Bo": 92}
2 print(d.pop("Chen", None))
3 print(d.popitem())
4 print(dict.fromkeys(["x", "y"], 0))
5 print(d.clear())
6 print(d)
```

output

```
None
('Bo', 92)
{'x': 0, 'y': 0}
None
{}
```

| A Frozen Set

code

```
1 key = frozenset({1, 2})
2 d = {key: "pair"}
3 print(d[frozenset({2, 1})])
4 print(key == {1, 2})
```

output

```
pair
True
```

| eval() and exec(): Trusted Constants Only

code

```
1 print(eval("1 + 2"))
2 scope = {}
3 print(exec("x = 1 + 2", scope))
4 print(scope["x"])
```

output

```
3
None
3
```