# Dictionaries

Python Reference

## Dictionaries

A dictionary 字典 (`dict`) stores key 键 → value 值 pairs. You look up a value by its key, not by a number index.

```
student = {"name": "Mei", "score": 88}
print(student["name"])     # Mei
print(student["score"])    # 88
```

### Add and update

Assign to a key to add it, or to change an existing one.

```
student = {"name": "Mei"}
student["score"] = 88      # add a new key
student["score"] = 90      # update the value
print(student)             # {'name': 'Mei', 'score': 90}
```

### Check and loop

Use `in` to test for a key. Loop over the keys, or over `.items()` to get both key and value.

```
student = {"name": "Mei", "score": 90}
print("score" in student)  # True
for key, value in student.items():
    print(key, "=", value)
# name = Mei
# score = 90
```

- `.get(key, default)` returns a default 默认值 when the key is missing — no error.

```
student = {"name": "Mei"}
print(student.get("age", 0))   # 0
```

The classic exam pattern — tally how often each value appears:

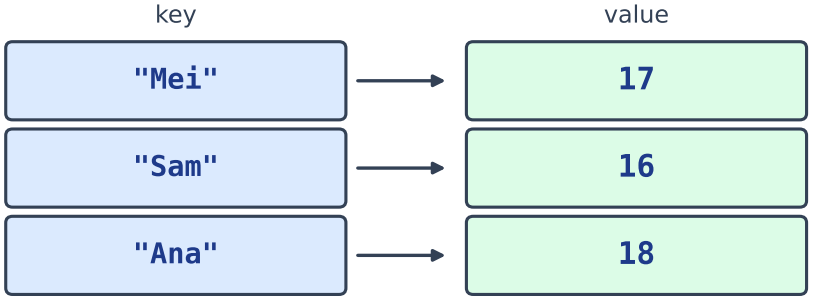
```
word = "banana"
counts = {}
```

```
for letter in word:
    counts[letter] = counts.get(letter, 0) + 1
print(counts)   # {'b': 1, 'a': 3, 'n': 2}
```

- `.get(letter, 0)` supplies `0` the first time a key is seen, so there is no KeyError.

### Common mistakes

- Reading a missing key with `d[key]` raises a KeyError; use `d.get(key)` or test `if key in d` first.
- Assigning `d[key]` again overwrites the old value — keys are unique.
- Keys must be immutable, such as a string or number — a list cannot be a key.




Exam tip: ages["Mei"] looks up the value; keys must be unique and immutable

*A dictionary maps each key to one value*