# 字典

Python Reference

## 字典

字典 (dictionary,`dict`) 保存键 (key)→ 值 (value) 的成对数据。你通过键来查找值，而不是通过数字索引。

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

### 添加与更新

给一个键赋值，就能添加它，或改变已有的键。

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

### 检查与遍历

用 `in` 检查某个键是否存在。遍历所有键，或遍历 `.items()` 同时得到键和值。

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

- 当键不存在时，`.get(key, default)` 会返回一个默认值 (default)—— 不会报错。

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



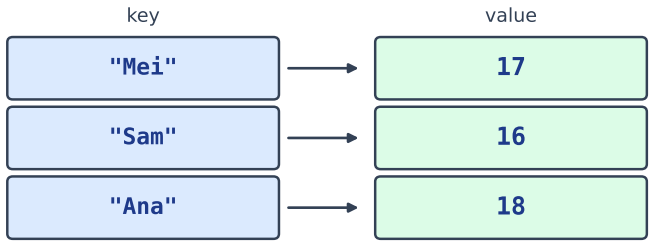
考试里的经典模式 —— 统计每个值出现多少次：

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

- 第一次遇到某个键时，`.get(letter, 0)` 会补上 0，所以不会有 KeyError。

**常见错误**

- 用 `d[key]` 读取不存在的键会抛出 KeyError；先用 `d.get(key)` 或判断 `if key in d`。
- 再次给 `d[key]` 赋值会覆盖旧值 —— 键是唯一的。
- 键必须是不可变的，比如字符串或数字 —— 列表不能当键。




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

*A dictionary maps each key to one value*