错误、异常与测试

Python Reference

Errors & debugging

代码会以三种方式出错。语法错误 (syntax error) 违反了 Python 的规则, 所以根本无法运行。运行时错误 (runtime error) 在运行过程中发生, 比如除以零。逻辑错误 (logic error) 能运行, 但给出错误的答案。



Exam tip: read the last line of the traceback first — it names the error and the line

Common Python errors: Syntax, Name, Type, Index

```
# A runtime error, caught so this block still finishes:
try:
    print(10 / 0)
except ZeroDivisionError:
    print("cannot divide by zero")
# cannot divide by zero
```

- Python 会打印显示失败位置的回溯信息。请自下而上阅读。

try / except / raise

将高风险代码放入 `try`；若发生错误，`except` 将捕获异常而非导致程序崩溃。

```
def to_int(text):
    try:
        return int(text)
    except ValueError:
        return 0
print(to_int("abc"))   # 0
```

- 捕获具体的类型 (`ValueError`、`ZeroDivisionError` 等)。

- `raise` 用来故意抛出你自己的错误。

```
def set_age(age):
    if age < 0:
        raise ValueError("age cannot be negative")
    return age

try:
    set_age(-1)
except ValueError as err:
    print("error:", err)
# error: age cannot be negative
```

Testing & robustness

测试 (test) 用来检查代码是否给出正确答案。既要试正常情况, **也**要试边界情形 (edge case)— 空输入、零、非常大的值。

```
def average(nums):
    if len(nums) == 0:        # edge case: empty list
        return 0
    return sum(nums) / len(nums)

print(average([2, 4, 6]))     # 4.0
print(average([]))            # 0
```

- 健壮的代码能优雅地处理异常输入，而不会直接崩溃。

常见错误

- 不要用光秃秃的 `except:`— 要捕获具体错误, 例如 `except ValueError:`。
- 语法错误 (syntax error) 会在程序运行前就中止它, 所以先修这些。
- 要测试边界情况 (空输入、零、最大值), 不能只测最简单的那种。