Errors, exceptions & testing

Python Reference

Errors & debugging

Code can fail in three ways. A syntax error 语法错误 breaks Python's rules, so it never runs. A runtime error 运行时错误 happens while running, like dividing by zero. A logic error 逻辑错误 runs but gives the wrong answer.

```
# 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 prints a traceback 回溯 showing where it failed. Read it from the bottom up.

try / except / raise

Wrap risky code in `try`. If it fails, `except` catches the exception 异常 and handles 处理 it, instead of crashing.

```
def to_int(text):
    try:
        return int(text)
    except ValueError:
        return 0

print(to_int("42"))    # 42
print(to_int("abc"))   # 0
```

- Catch a specific type (`ValueError`, `ZeroDivisionError`, ⋯).
- `raise` makes your own error on purpose.

```
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

A test 测试 checks that code gives the right answer. Try normal cases **and** edge cases 边界情形—empty input, zero, very large values.

```
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
```

- Robust 健壮 code does not crash on strange input; it handles it gracefully.