

Errors, exceptions & testing

Python Reference

Errors & debugging

Code can fail in three ways. A syntax error breaks Python's rules, so it never runs. A runtime error happens while running, like dividing by zero. A logic error 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 showing where it failed. Read it from the bottom up.

SyntaxError

code shape
wrong

NameError

name not
defined

TypeError

wrong type
of value

IndexError

index out
of range

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

Common Python errors: Syntax, Name, Type, Index

try / except / raise

Wrap risky code in `try`. If it fails, `except` catches the exception and handles 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 checks that code gives the right answer. Try normal cases **and** edge cases — 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 code does not crash on strange input; it handles it gracefully.

Common mistakes

- Do not use a bare `except:` — catch the specific error, e.g. `except ValueError:`.
- A syntax error stops the whole program before it runs, so fix those first.
- Test the edge cases (empty input, zero, the largest value), not just the easy one.