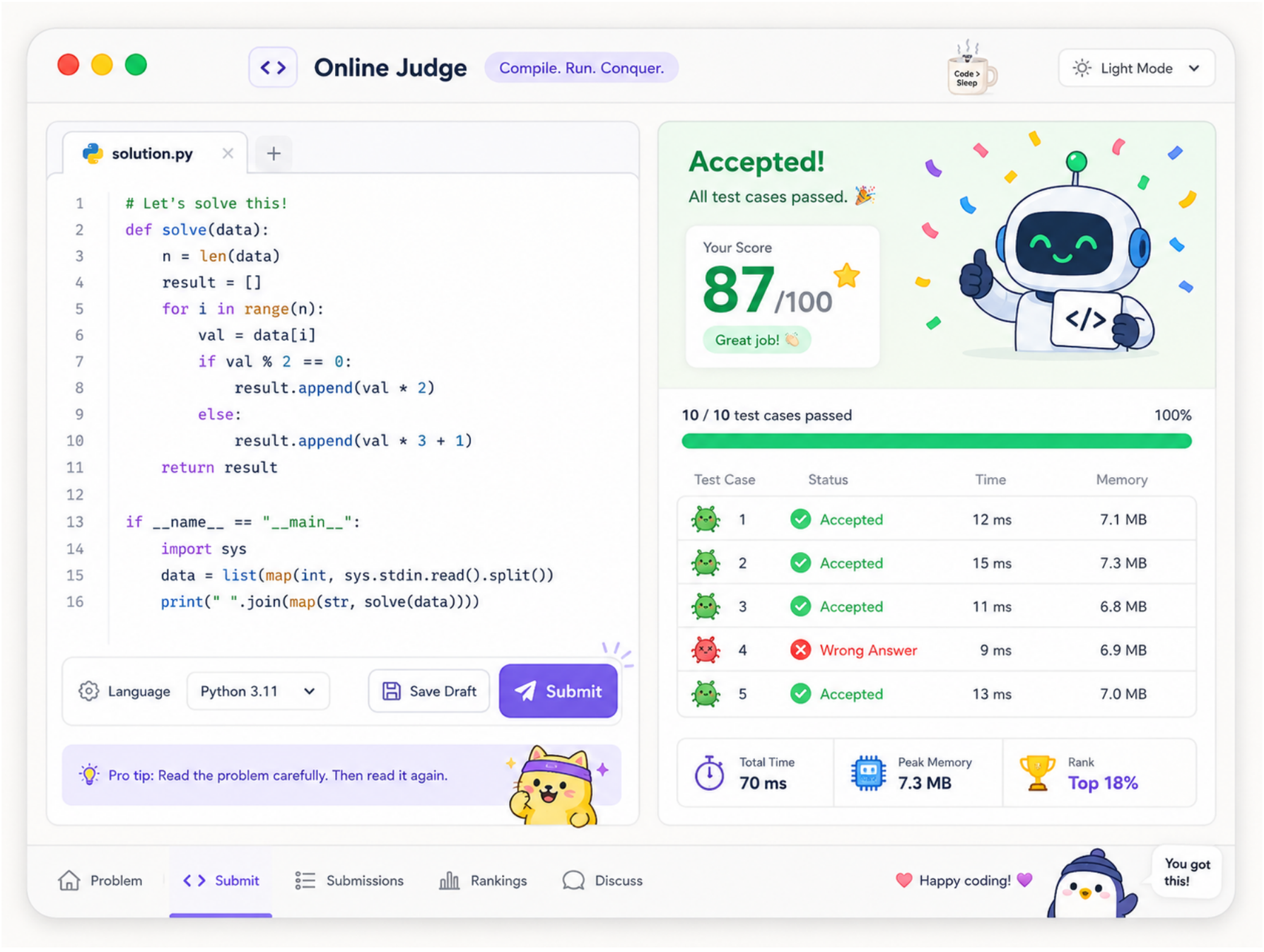
软件测试概论

Introduction to Software Testing

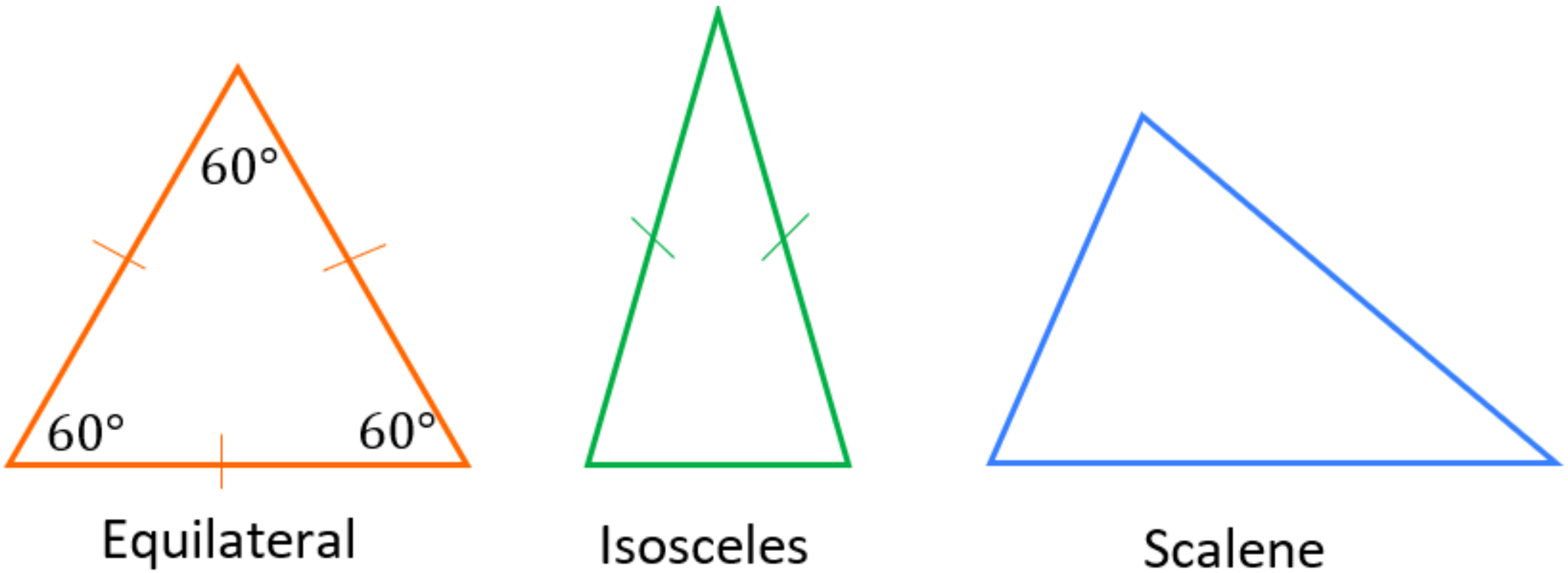
2026

先从一个简单的测试开始



先从一个简单的测试开始

编写一个三角形分类程序 `classify_triangle(x, y, z)`，
其功能是判断三角形是等腰三角形、等边三角形还是普通三角形



先从一个简单的测试开始

编写一个三角形分类程序 `classify_triangle(x, y, z)`，
其功能是判断三角形是等腰三角形、等边三角形还是普通三角形

✅ 能识别有效的不规则三角形、等边三角形

✅ 能识别有效的等腰三角形 e.g., (3, 3, 4), (3, 4, 3) ...

✅ 能处理两边之和小于第三条边 e.g., (1, 2, 4), (1, 4, 2) ...

✅ 能处理两边之和恰好等于第三条边 e.g., (1, 2, 3), (1, 3, 2) ...

✅ 能处理输入的边长为非整数值

✅ 能处理某边长度等于 0 / 三条边长度都等于 0

✅ 能处理某边长度等于负数 / 三条边长度都为负数

❓

测试何时可以停止

When to Stop

一个软件的实现是否“正确”

Rationalists vs. Empiricists

René Descartes

勒内·笛卡尔

Francis Bacon

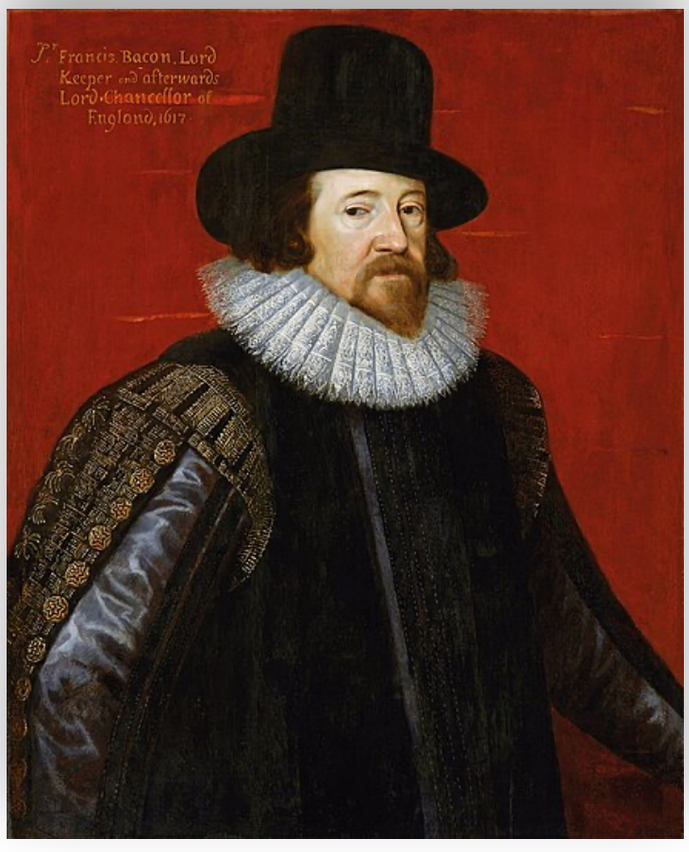
弗朗西斯·培根

一个软件的实现是否“正确”

Rationalists vs. Empiricists

“It is correct because I proved that certain errors do not exist in the system”

Formal Verification (Static)



“It is correct because I tried it several times and it ran okay”

Software Testing (Dynamic)

一个软件的实现是否“正确”

Over-Approximation of Static Analysis

😸 考虑程序在所有可能输入下的行为

🙀 但并非所有输入都会在实际执行中出现

😿 受限于分析的范围和精度、以及只能通过动态执行观察的行为

(e.g., energy consumption)

```python
def compute_density(a, b):
    return a / b

def process_data(x, y):
    if y <= 0:
        do_something()
    else:
        d = compute_density(x, y)
        do_something_else(d)
```

一个软件的实现是否“正确”

Under-Approximation of Dynamic Analysis

- 😸 考虑程序在实际执行中观察到的行为
- 🙀 没有用于执行的输入可能导致程序的不同结果
- 😿 受限于其它没有使用的输入

```python
def compute_density(a, b):
    return a / b

def test_compute_density():
    assert compute_density(1, 2) == 0.5
    assert compute_density(2, 8) == 0.25
```

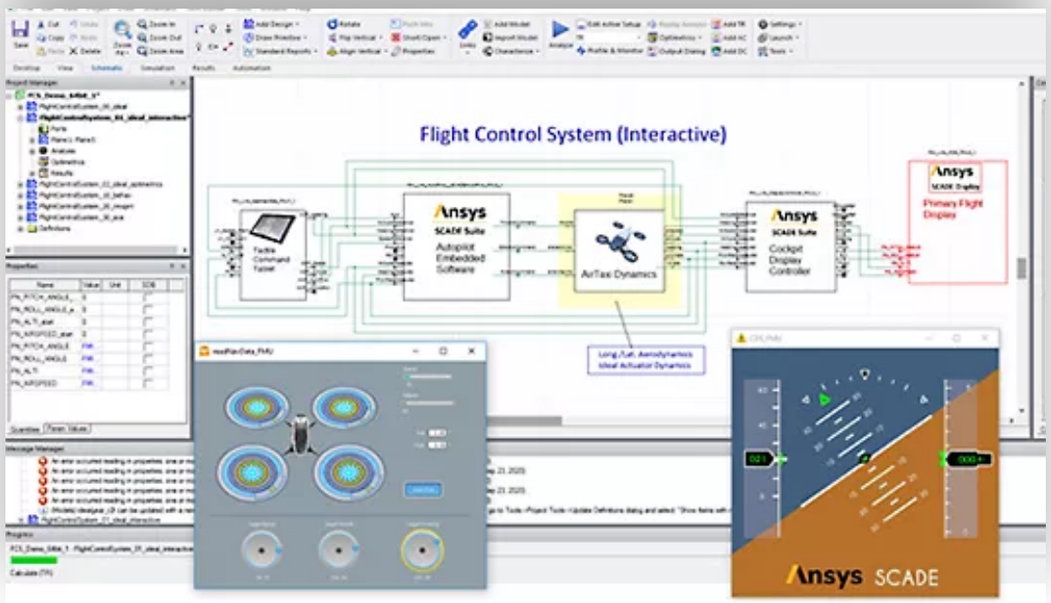
一个软件的实现是否“正确”

A Famous Quote

“Program testing can be used to show the presence of bugs, but never to show their absence.”

一个软件的实现是否“正确”

A Thought Experiment



飞行控制系统 A

从来没有在任何航班上测试过

但形式化证明为正确

飞行控制系统 B

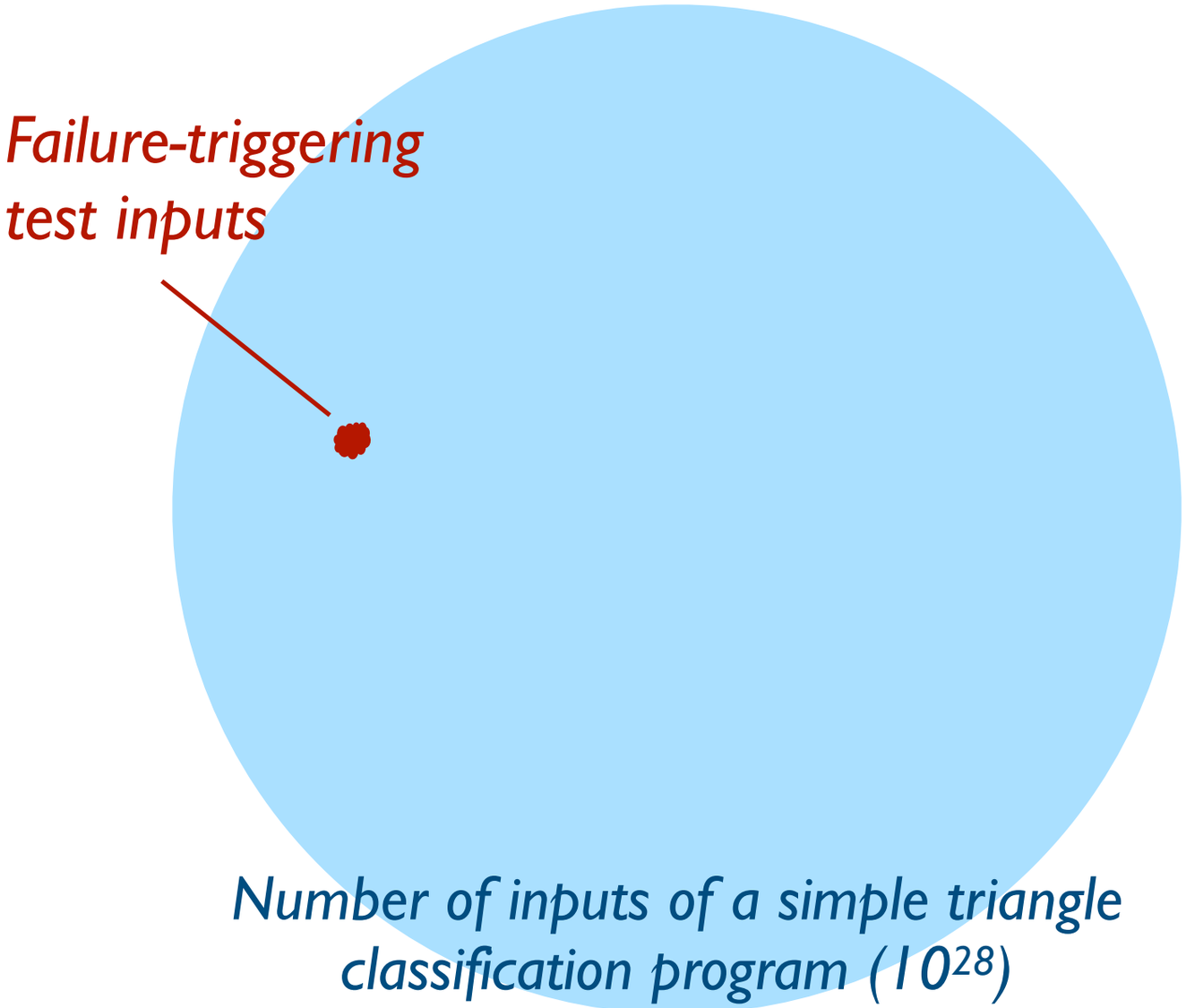
从来没有被形式化分析过

但已在数亿航班上被测试

一个软件的实现是否“正确”

A Simple Proof

穷尽测试 (Exhaustive Testing)：使用程序所有可能接受的输入来进行测试，以确保程序在任何时候都能正确运行



测试输入空间采样

软件测试就是一个采样 (sampling) 的过程

- 任何实践可行的软件测试方法都是从潜在巨大的测试输入空间中选择一部分最有代表性的测试输入来执行
- 其中的困难和创造性之处就在于如何选择 “好” 的测试数据
  - 需要对软件行为和实现有深刻洞察
- 以最小的测试成本 (cost) 达到最大的测试效果 (effectiveness)

测试充分性准则

测试充分性 (Test Adequacy) 准则提供了一种测试用例设计的指导

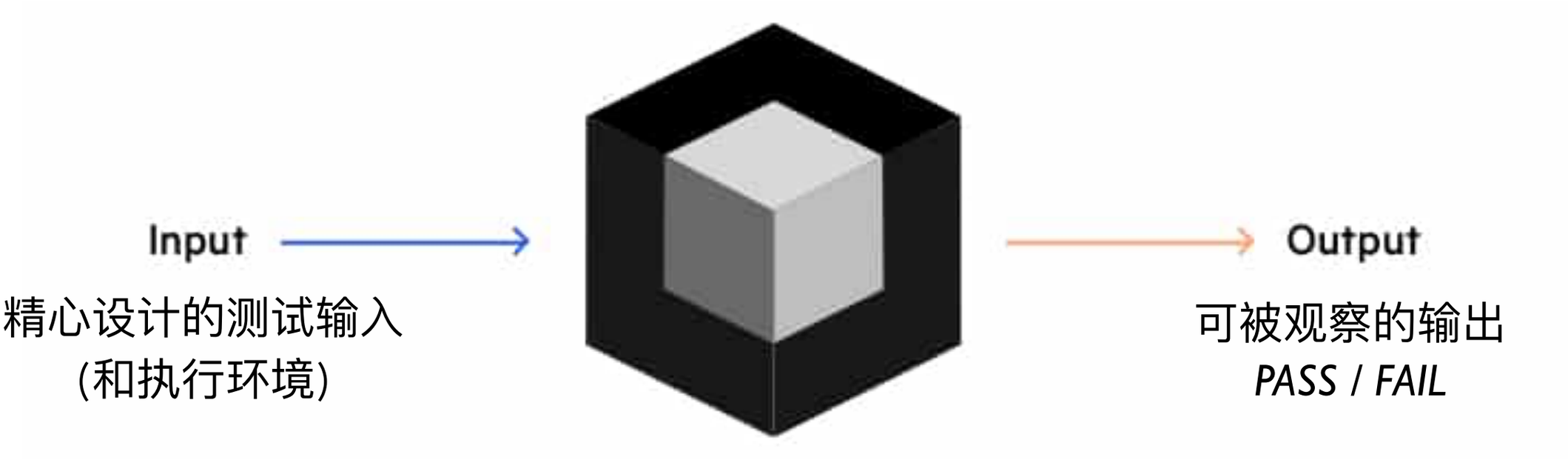
- 对某种东西 (例如代码行、功能点等) 的系统性覆盖
  - 可以用于评价两个候选测试用例的优劣
  - 可以作为一种停止准则，在达到 100% Coverage 时停止
    - 但达到 100% Coverage 就一定充分吗?
- 在很多自动化测试场景中会以测试成本 (budget) 作为停止准则
  - 此时测试充分性准则仍可作为测试用例设计的优化目标 (maximise some coverages)

软件行为是否正确

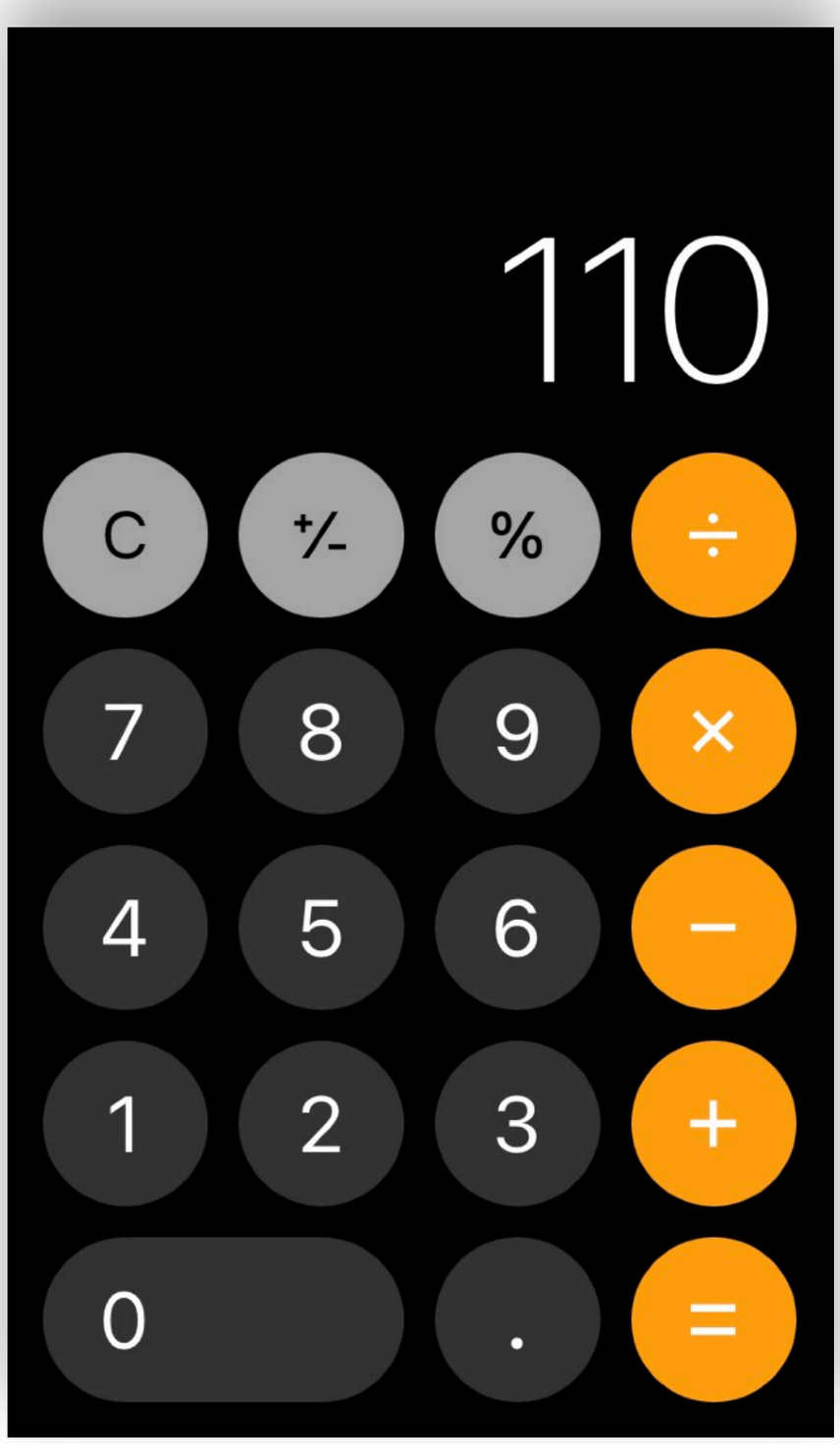
What is Right

测试预期输出

Test Oracle 是一种用于判断待测软件行为是否符合预期的机制



测试预期输出



依次按 “100 + 10 %”，然后按 “=”

- 100.1 ?
- 110 ?

测试预期输出

我们需要依赖待测软件的某些信息 (需求说明、领域知识等) 来实现可靠的 Test Oracle

- **Implicit Oracle:** 任何程序语义下都不应该出现的情况 (e.g., crash)
- **Explicit Oracle:** 不满足软件规格说明 (specification) 的要求
  - 未达到规格说明中指明的目标 (功能错误、运行缓慢、难以理解、不易使用等)
  - 出现了规格说明中指明不会出现的错误
  - 实现了规格说明书中未提到的功能 (trustworthy)

自动化判定

Automated Test Oracle

我们可以依靠领域专家给出预期输出结果，但如何自动化这一活动?

```
[TestMethod]
public void Filters_OnRefreshAndCacheNotExpired_AreResetToDefaults()
{
    // Arrange
    var vm = new CatalogViewModel(_container);
    var tracker = new PropertyChangeTracker(vm);
    vm.Initialize();
    tracker.WaitForChange("LastUpdateTime", 1);
    vm.Include70s = false;
    vm.Include80s = false;
    vm.Include90s = false;
    vm.Include00s = false;

    // Act
    tracker.Reset();
    vm.RefreshCatalog();
    tracker.WaitForChange("LastUpdateTime", 1);

    // Assert
    Assert.IsTrue(vm.Include70s, "Include70s filter was not reset");
    Assert.IsTrue(vm.Include80s, "Include80s filter was not reset");
    Assert.IsTrue(vm.Include90s, "Include90s filter was not reset");
    Assert.IsTrue(vm.Include00s, "Include00s filter was not reset");
}
```

不稳定结果

Flaky Tests

如果同一个测试输入在不同的测试执行时产生了不同的结果，那测试到底是 Pass 还是 Fail？

Test flakiness at Spotify

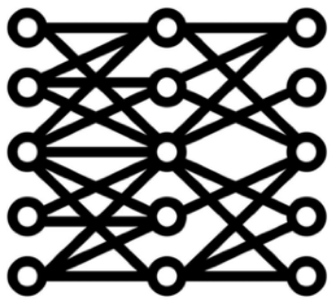
不可测程序

Untestable Programs

如果连人工都无法准确判断测试预期输出 (e.g., 待测软件编写出来就是为了解决某个我们都不知道答案的问题)，那我们还能如何进行测试?

3874634234529 →  → 5 ?

计算 PI 的第 N 位

→  → Cat / Dog ?

基于 DNN 的猫/狗图片分类器

软件测试的两个关键问题

测试用例 (Test Case) = 测试输入 + 预期输出

1️⃣ 知道测试何时可以停止

- 测试充分性准则和采样策略
- 任何一次测试活动的基本限制

2️⃣ 知道软件行为是否正确

- 测试预期输出判定
- 提高测试自动化程度的必然需求

👨‍💻 任何一个软件测试方法 (软件测试人员) 都需要以他自己的方式来回答这两个问题