Debugging skills

测试技能

- Treat as a search problem: looking for explanation for incorrect behavior

  看作是一个搜索问题：寻找不正确行为的解释

  - Study available data – both correct test cases and incorrect ones

    研究可用数据–正确测试案例或者不正确案例

  - Form an hypothesis consistent with the data

    伴随着数据形成一个假说

  - Design and run a repeatable experiment with potential to refute the hypothesis

    设计而且运行一个重复的实验，伴随着驳斥假说的潜能

  - Keep record of experiments performed: use narrow range of hypotheses

    保持记录实验的运行：使用一个窄范围的假说。

Debugging as search

像搜索一样测试

- Want to narrow down space of possible sources of error

  什么会缩窄可能错误源头的空间

- Design experiments that expose intermediate stages of computation (use print statements!), and use results to further narrow search

  设计一个实验：暴露中间过程的计算

  使用输出语句

  使用结果到更窄的搜索

- Binary search can be a powerful tool for this

  二分法搜索对此来说是一个强有力的工具。

```
def isPal(x):
    assert type(x) == list
    temp = x
    temp.reverse
    if temp == x:
        return True
    else:
        return False

def silly(n):
    for i in range(n):
        result = []
        elem = raw_input('Enter element: ')
        result.append(elem)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

Stepping through the tests

通过测试进行步进

- Suppose we run this code:
  - We try the input ‘abcba’, which succeeds
  - We try the input ‘palinnilap’, which succeeds
  - But we try the input ‘ab’, which also ‘succeeds’
- Let’s use binary search to isolate bug(s)

让我们使用二分法搜索来隔离bugs

- Pick a spot about halfway through code, and devise experiment

选取一个点，大概在代码的中部，然后设计一个实验

  - Pick a spot where easy to examine intermediate values

选取一个点，这个点很容易检查中部值。

```
def isPal(x):
    assert type(x) == list
    temp = x
    temp.reverse
    if temp == x:
        return True
    else:
        return False

def silly(n):
    for i in range(n):
        result = []
        elem = raw_input('Enter element: ')
        result.append(elem)
    print(result)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

Stepping through the tests

- At this point in the code, we expect (for our test case of ‘ab’), that result should be a list [‘a’, ‘b’]
- We run the code, and get [‘b’].
- Because of binary search, we know that at least one bug must be present earlier in the code
- So we add a second print

```
def isPal(x):
    assert type(x) == list
    temp = x
    temp.reverse
    if temp == x:
        return True
    else:
        return False

def silly(n):
    for i in range(n):
        result = []
        elem = raw_input('Enter element: ')
        result.append(elem)
        print(result)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

Stepping through

- When we run with our example, the print statement returns
  - ['a']
  - ['b']
- This suggests that result is not keeping all elements
  - So let's move the initialization of result outside the loop and retry

```
def isPal(x):
    assert type(x) == list
    temp = x
    temp.reverse
    if temp == x:
        return True
    else:
        return False

def silly(n):
    result = []
    for i in range(n):
        elem = raw_input('Enter element: ')
        result.append(elem)
        print(result)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

Stepping through

- So this now shows we are getting the data structure result properly set up, but we still have a bug somewhere
  - A reminder that there may be more than one problem!
  - This suggests second bug must lie below print statement; let's look at isPal
  - Pick a point in middle of code, and add print statement again

所以我们正在获得数据结构结果正确的设置，但是我们依然有一个bug。

一个提示是这里可能有超过一个问题。

这表明第二个bug必须处于输出语句的下面；让我们看一下isPal:

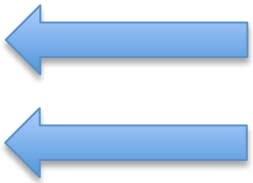
选取一个代码中部的点，而且增加再次增加一个输出语句。

```
def isPal(x):
    assert type(x) == list
    temp = x
    temp.reverse
    print(temp, x)
    if temp == x:
        return True
    else:
        return False

def silly(n):
    result = []
    for i in range(n):
        elem = raw_input('Enter element: ')
        result.append(elem)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

Stepping through

- At this point in the code, we expect (for our example of ‘ab’) that x should be [‘a’, ‘b’], but temp should be [‘b’, ‘a’], however they both have the value [‘a’, ‘b’]
- So let’s add another print statement, earlier in the code

```
def isPal(x):
    assert type(x) == list
    temp = x
    print(temp, x)
    temp.reverse
    print(temp, x)
    if temp == x:
        return True
    else:
        return False

def silly(n):
    result = []
    for i in range(n):
        elem = raw_input('Enter element: ')
        result.append(elem)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```

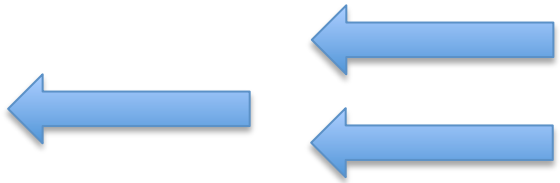


Stepping through

- And we see that temp has the same value before and after the call to reverse
- If we look at our code, we realize we have committed a standard bug – we forgot to actually invoke the reverse method
  - Need temp.reverse()
- So let's make that change and try again

提交一个标准的bug–我们忘记去实际的使用reverse方法

```
def isPal(x):
    assert type(x) == list
    temp = x
    print(temp, x)
    temp.reverse()
    print(temp, x)
    if temp == x:
        return True
    else:
        return False

def silly(n):
    result = []
    for i in range(n):
        elem = raw_input('Enter element: ')
        result.append(elem)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```



Stepping through

- But now when we run on our simple example, both x and temp have been reversed!!
- We have also narrowed down this bug to a single line. The error must be in the reverse step

我们必须缩窄这个bug到单独行。这个错误必须在取反这一步中。

- In fact, we have an aliasing bug – reversing temp has also caused x to be reversed

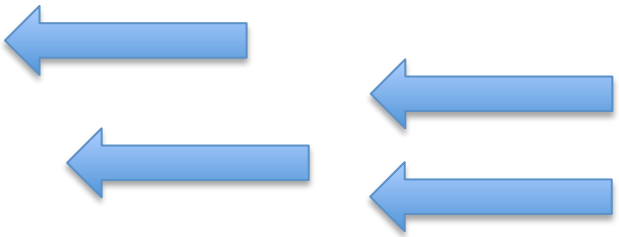
别名错误

反向的temp也会造成x的反向。

  - Because they are referring to the same object

参考同样的目标

```
def isPal(x):
    assert type(x) == list
    temp = x[:]
    print(temp, x)
    temp.reverse()
    print(temp, x)
    if temp == x:
        return True
    else:
        return False

def silly(n):
    result = []
    for i in range(n):
        elem = raw_input('Enter element: ')
        result.append(elem)
    if isPal(result):
        print('Yes')
    else:
        print('No')
```



Stepping through

- And now running this shows that before the reverse step, the two variables have the same form, but afterwards only temp is reversed.
在取反步骤的前面，这两个变量有相同的形式，但是后来只有temp取反。
- We can now go back and check that our other tests cases still work correctly

Some pragmatic hints

- Look for the usual suspects
  查看常见的方面
- Ask why the code is doing what it is, not why it is not doing what you want
- The bug is probably not where you think it is – eliminate locations
  这个bug很可能不是你想到的位置 — 消除位置
- Explain the problem to someone else
  将问题解释给其他人
- Don't believe the documentation
  不要过度相信文件说明
- Take a break and come back to the bug later
  休息一下然后稍后回道bug这里