[301] Refactoring Conditionals

Tyler Caraza-Harter

Today's Outline

Review

Refactoring Conditionals

TODO

Today's Outline

Review

Refactoring Conditionals

Refactor Exercise 1

```
def or2(cond1, cond2):
    return cond1 or cond2
```

which refactor is correct?

hint: `or2(False, True)`

```
def or2(cond1, cond2):
    rv = False
    rv = rv or cond1
    rv = rv or cond2
    return rv
```

A

```
def or2(cond1, cond2):
    if cond1:
        return cond2
    else:
        return False
```

B

Refactor Exercise 1

```
return b1 or b2 or b3 or ... or bN
```

equivalent

```
rv = False
rv = rv or b1
rv = rv or b2
rv = rv or b3
...
rv = rv or bN
```

Lesson: with "or", it only takes one to flip the whole thing True!

Refactor Exercise 2

```
def and2(cond1, cond2):
    return cond1 and cond2
```

which refactor is correct?

hint: `and2(True, True)`

A

```
def and2(cond1, cond2):
    rv = False
    rv = rv and cond1
    rv = rv and cond2
    return rv
```

B

```
def and2(cond1, cond2):
    if cond1:
        return cond2
    else:
        return False
```

Refactor Exercise 2

```
return b1 and b2 and b3 and ... and bN
```

equivalent

```
if b1:
    return b2 and b3 and ... and bN
else:
    return False
```

Lesson: with "and", the first one can make the whole thing False!

Refactor Exercise 3

```
def fix(moves, should):
    if moves:
        if should:
            return "good"
        else:
            return "duct tape"
    else:
        if should:
            return "WD-40"
        else:
            return "good"
```

which refactor is correct?

hint: `fix(False, False)`

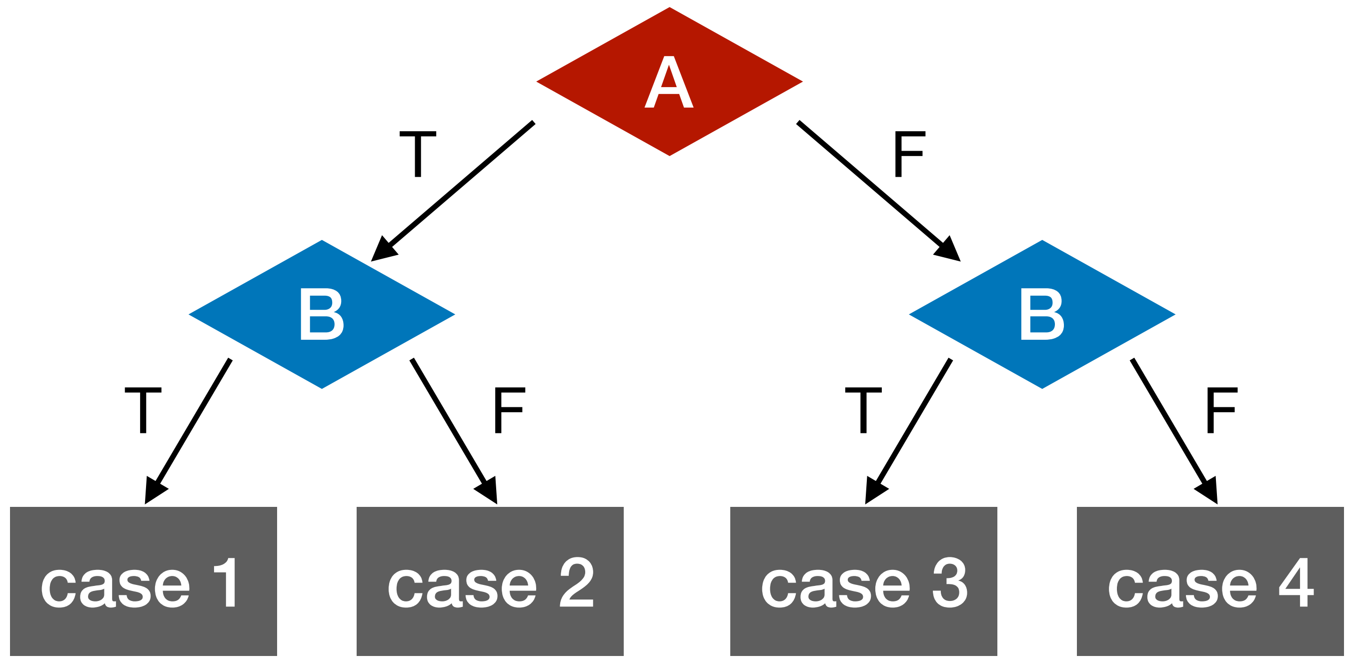
A

```
def fix(moves, should):
    if moves and not should:
        return "duct tape"
    elif: not moves and should:
        return "WD-40"
    elif moves and should:
        return "good
    elif not moves and not should:
        return "good"
```

B

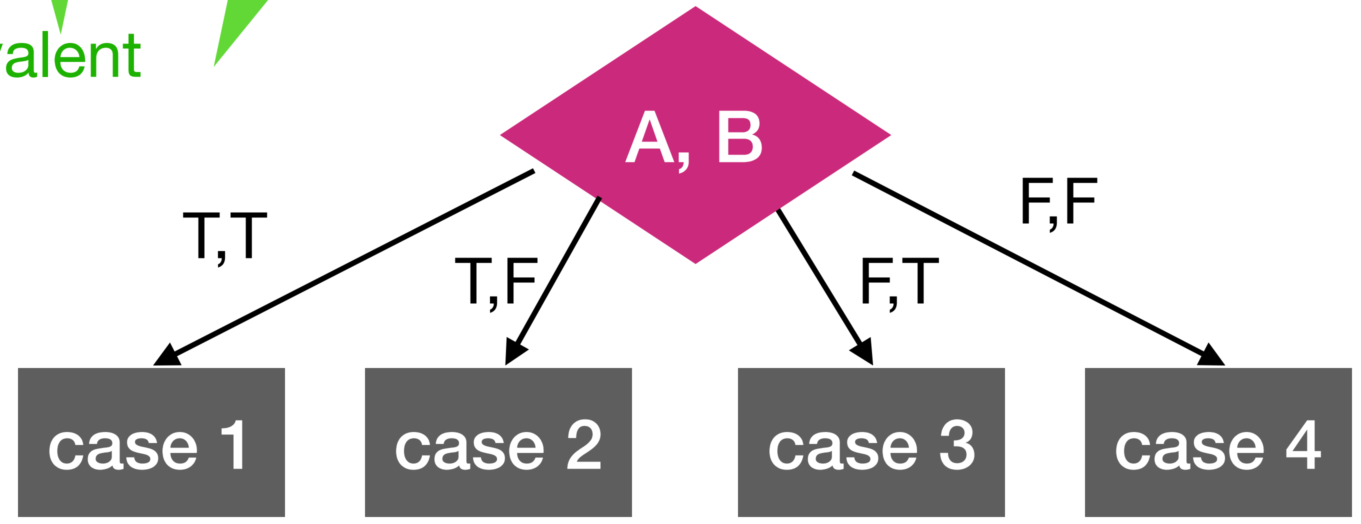
```
def fix(moves, should):
    if should:
        if moves:
            return "duct tape"
        else:
            return "good"
    else:
        if moves:
            return "good"
        else:
            return "duct tape"
```

Refactor Exercise 3



Option 1: Nesting

Option 2: Chaining



Lesson: when handling combinations of booleans, you can either do either (a) nesting or (b) chaining with `and`

Refactor Exercise 4

```
def is_301(a, b, c):
    return a==3 and b==0 and c==1
```

which refactor is correct?

hint: `is_301(3, 0, 1)`

```
def is_301(a, b, c):
    if a==3:
        if c==1:
            if b==0:
                return True
    return False
```

A

```
def is_301(a, b, c):
    if a==3 or b==0 or c==1:
        return False
    return True
```

B

Refactor Exercise 4

```
return b1 and b2 and b3 and ... and bN
```

```
if b1:
    if b2:
        if b3:
            ...
                if bN:
                    return True
return False
```

Lesson: nesting a lot of if's inside each other is equivalent to and'ing all the conditions

Refactor Exercise 5

```
def is_301(a, b, c):
    return a==3 and b==0 and c==1
```

which refactor is correct?

hint: `is_301(3, 9, 1)`

```
def is_301(a, b, c):
    if a==3:
        return True
    if b==0:
        return True
    if c==1:
        return True
    return False
```

A

```
def is_301(a, b, c):
    if a!=3:
        return False
    if b!=0:
        return False
    if c!=1:
        return False
    return True
```

B

Refactor Exercise 5

```
return b1 and b2 and b3 and ... and bN
```

equivalent

```
if not b1:
    return False
if not b2:
    return False
if not b3:
    return False
...
if not bN:
    return False
return True
```

Lesson: checking if everything is True can be translated to seeing if we can find anything False