Exam1 conflict form – fill by Feb 16th

[220 / 319] Conditionals 2

Meena Syamkumar
Andy Kuemmel
Cole Nelson

Readings:

Parts of Chapter 5 of Think Python

Chapter 4.6 to end (skip 4.7) of Python for Everybody

Learning Objectives

Write nested conditional statements

Refactor code with Boolean operators into equivalent code with nested conditional statements

Refactor code with nested conditional statements into equivalent code with Boolean operators

Identify code blocks

- Count the number of blocks in nested code

Today's Outline

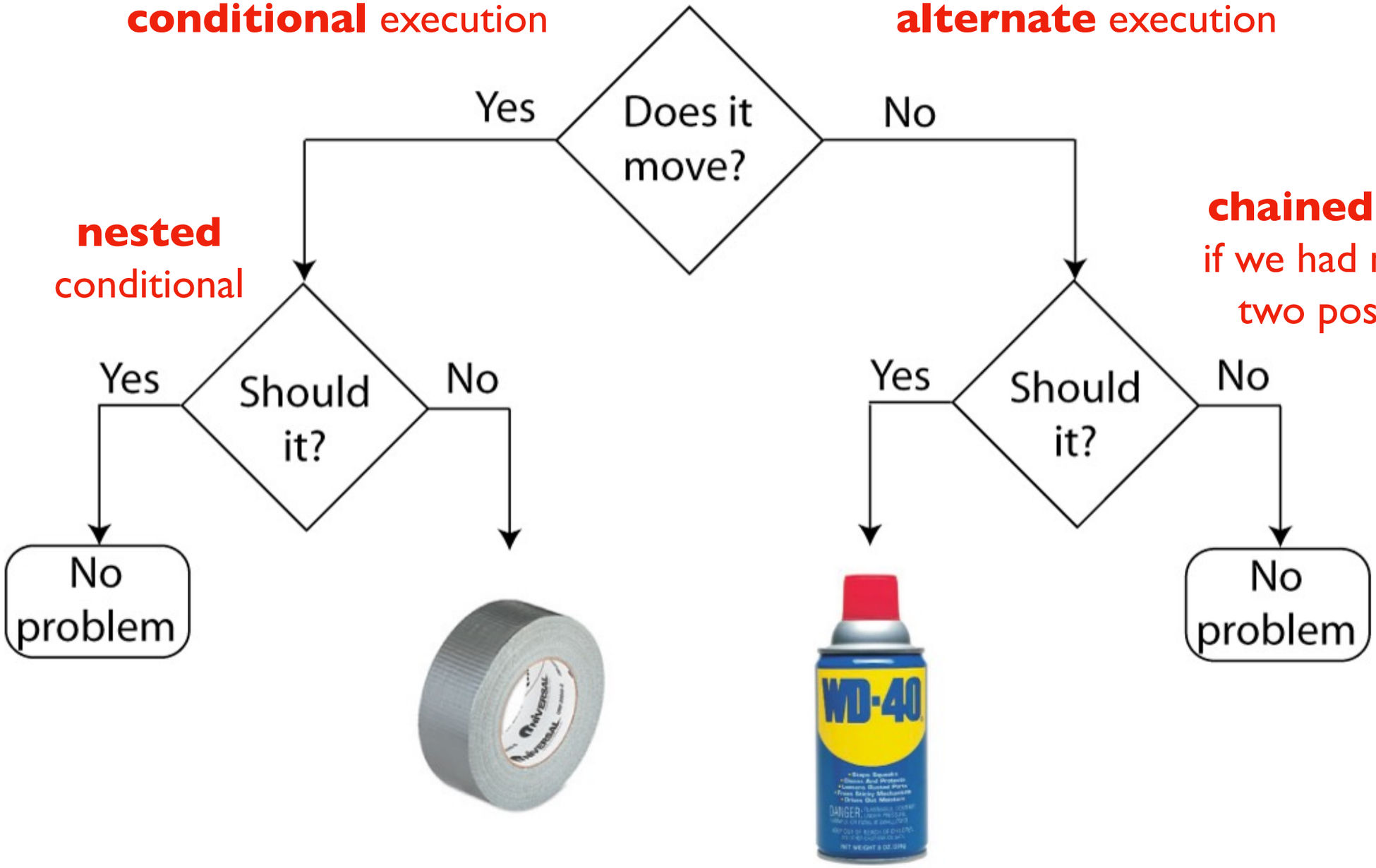
Nested Conditionals

Refactoring Conditionals

Laboratory Troubleshooting Flowchart

conditional execution

alternate execution



nested conditional

chained would be if we had more than two possibilities

in programming:

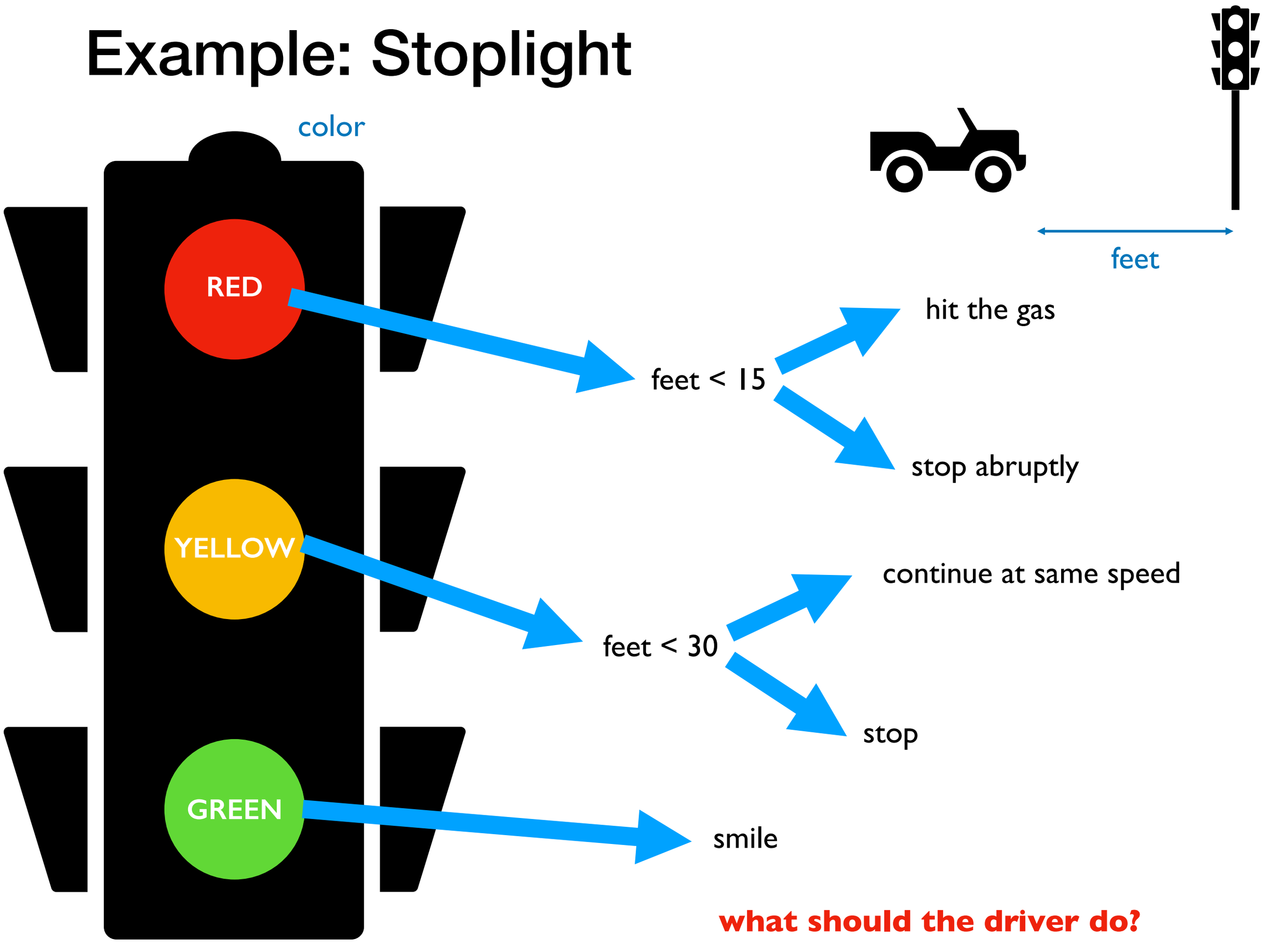
- questions are phrased as *boolean expressions*
- actions are *code/statements*

https://www.drphysics.com/prayer/flowchart.html

Nested Conditionals Example

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

Example: Stoplight

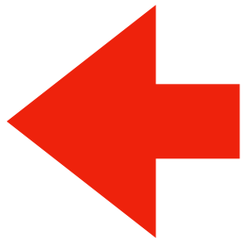


what should the driver do?

Today's Outline

Nested Conditionals

Refactoring Conditionals



How to use these slides

There are more examples here than we can cover in lecture.

However, you can walk through these examples along with the interactive exercises. You should do the following:

1. Think about what the answer is
2. Mentally step through the code using the example call when applicable
3. Step through the code with the Python Tutor examples we've setup for you. For the refactor examples, step through all three versions, and see which alternative (A or B) matches the output of the original version.
4. If you got something different than Python Tutor, tweak your mental model (talk to us if you don't understand something)

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)`

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

A

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

B

Refactor Exercise 2

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


```
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)`

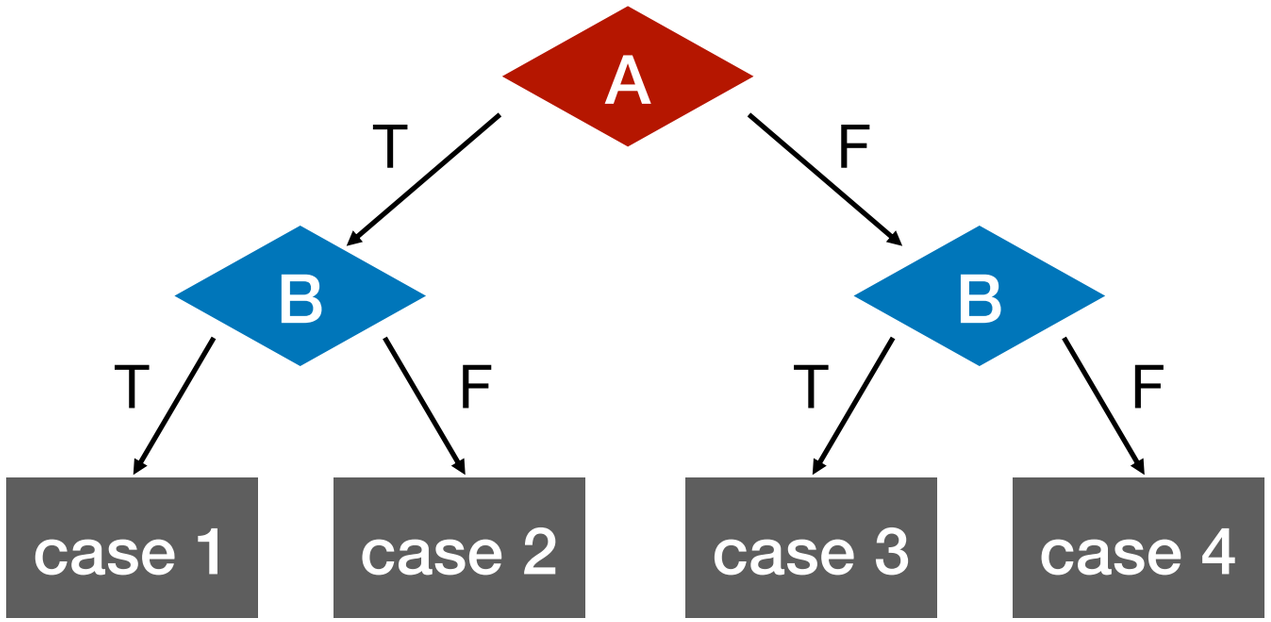
A

```
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"
```

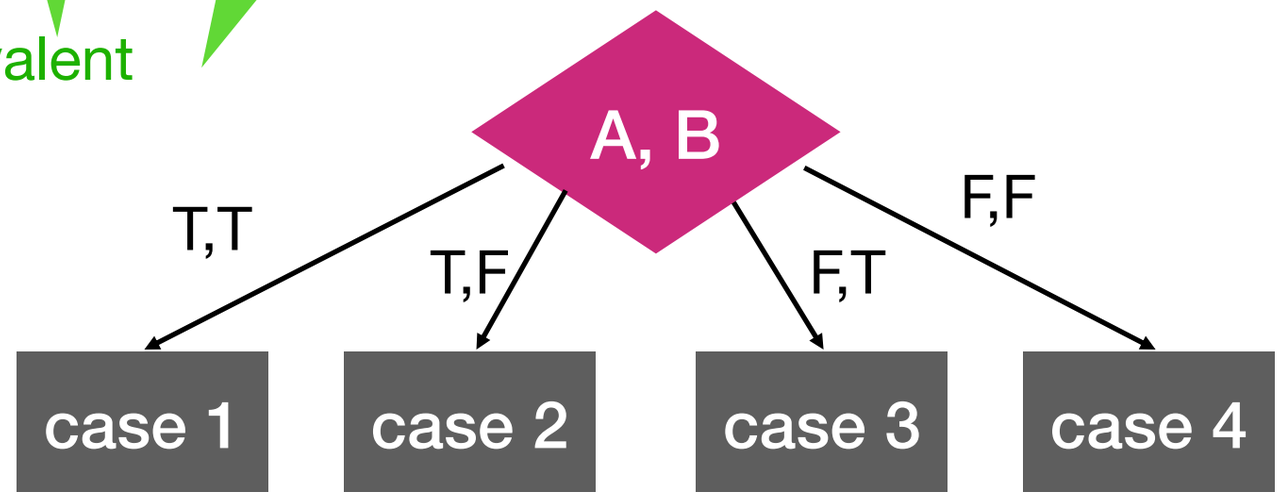
Refactor Exercise 3



Option 1: Nesting

equivalent

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_220(a, b, c):
    return a==2 and b==2 and c==0
```

which refactor is correct?

hint: `is_220(2, 2, 0)`

A

```
def is_220(a, b, c):
    if a==2:
        if c==0:
            if b==2:
                return True
    return False
```

B

```
def is_220(a, b, c):
    if a==2 or b==2 or c==0:
        return False
    return True
```

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_220(a, b, c):
    return a==2 and b==2 and c==0
```

which refactor is correct?

hint: `is_220(2, 2, 1)`

A

```
def is_220(a, b, c):
    if a==2:
        return True
    if b==2:
        return True
    if c==0:
        return True
    return False
```

B

```
def is_220(a, b, c):
    if a!=2:
        return False
    if b!=2:
        return False
    if c!=0:
        return False
    return True
```

Refactor Exercise 5

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


```
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