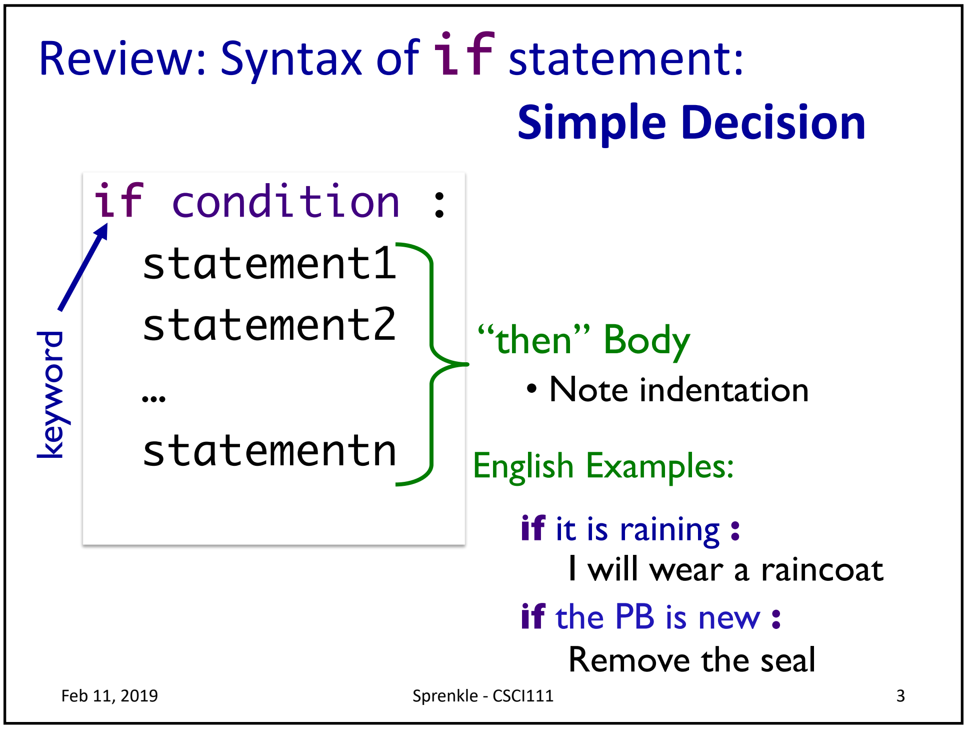
Objectives

- More conditionals
- Boolean operators

Review

- How can we make Python code execute only under certain circumstances?
- How do we say “otherwise” in Python?
- How do we write the condition that evaluates to `True` if two expressions (let’s say `expr1` and `expr2`) are equal?
  - How do we write the condition to evaluate to True only if those two expressions are *not* equal?



Review: Syntax of `if` statement: **Simple Decision**

```
if condition :
    statement1
    statement2
    …
    statementn
```

keyword

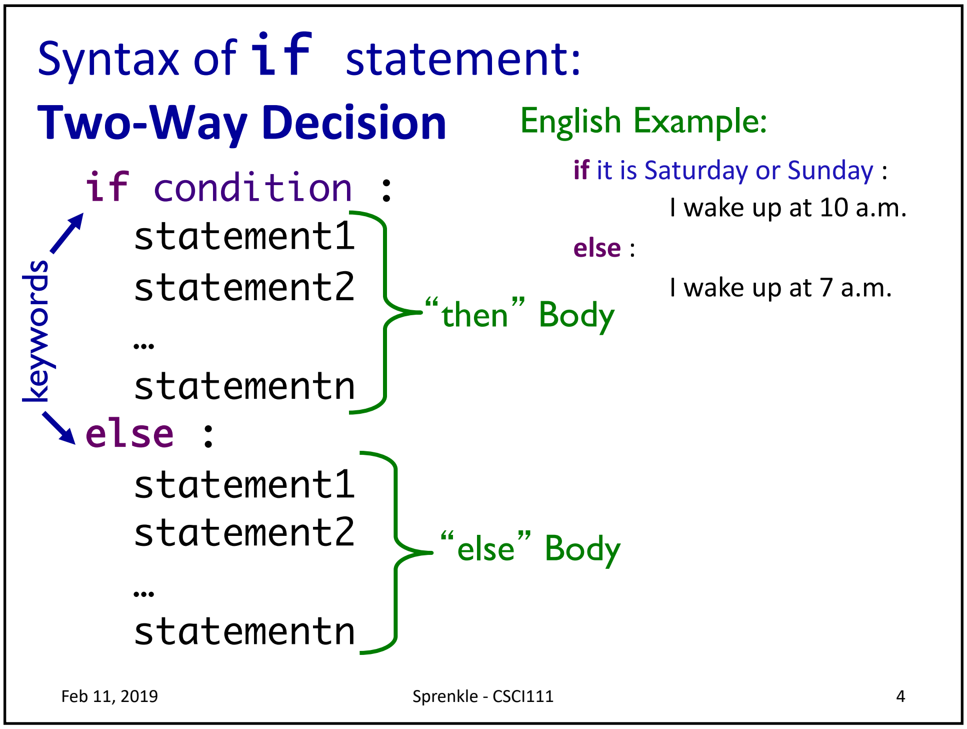
"then" Body

- Note indentation

English Examples:

if it is raining **:**
    I will wear a raincoat

if the PB is new **:**
    Remove the seal



Syntax of `if` statement: **Two-Way Decision**

```
if condition :
    statement1
    statement2
    …
    statementn
else :
    statement1
    statement2
    …
    statementn
```

keywords

"then" Body

"else" Body

English Example:

if it is Saturday or Sunday :
    I wake up at 10 a.m.
else :
    I wake up at 7 a.m.

Review: Relational Operators

- Syntax:
  - `<expr> <relational_operator> <expr>`

Low precedence

Relational Operator	Meaning
<	Less than?
<=	Less than or equal to?
>	Greater than?
>=	Greater than or equal to?
==	Equals?
!=	Not equals?

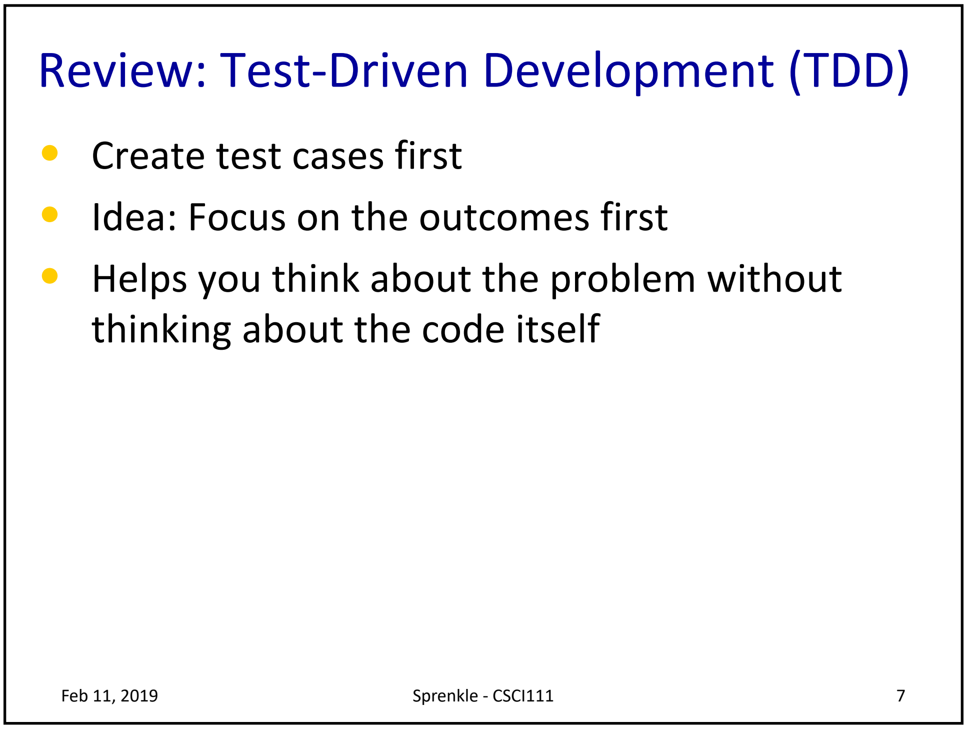
Review: Using Conditionals

- Determine if a number is even or odd

```
x = eval(input("Enter a number: "))
remainder = x%2
if remainder == 0:
    print(x, "is even")
if remainder == 1:
    print(x, "is odd")
```

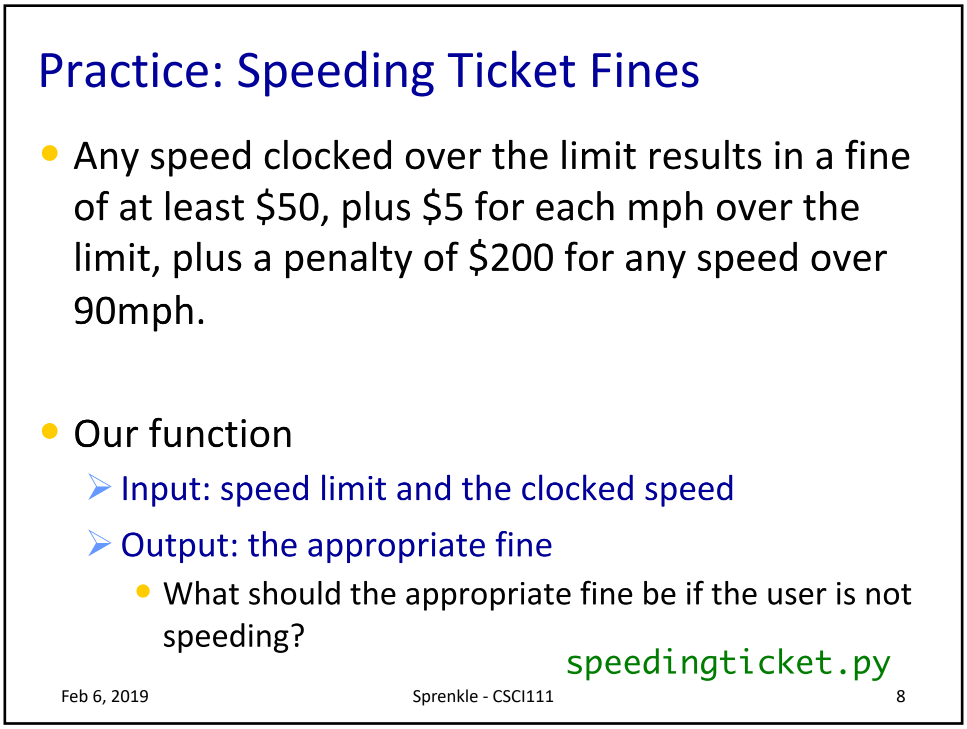
```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

This is the more efficient implementation. Why?



Review: Test-Driven Development (TDD)

- Create test cases first
- Idea: Focus on the outcomes first
- Helps you think about the problem without thinking about the code itself

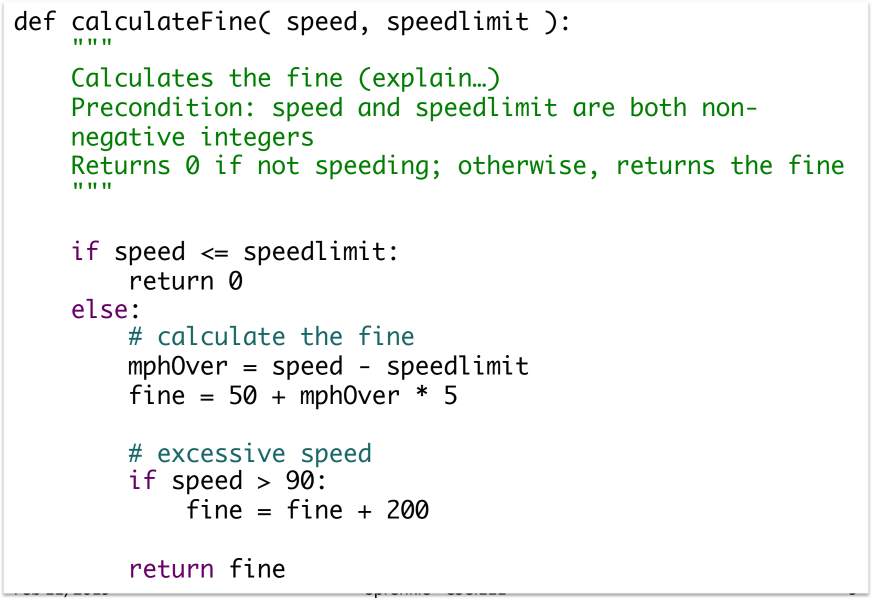


Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least $50, plus $5 for each mph over the limit, plus a penalty of $200 for any speed over 90mph.
- Our function
  - Input: speed limit and the clocked speed
  - Output: the appropriate fine
    - What should the appropriate fine be if the user is not speeding?

`speedingticket.py`

Speeding Ticket Fine



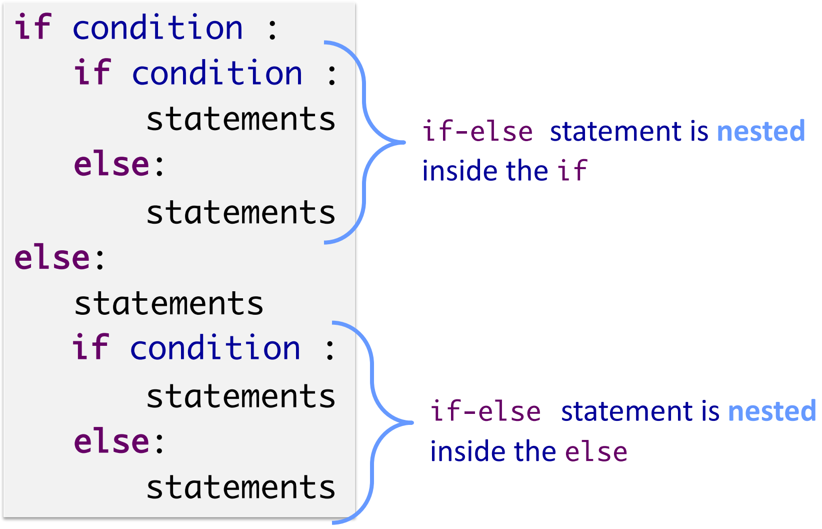
```
def calculateFine( speed, speedlimit ):
    """
    Calculates the fine (explain…)
    Precondition: speed and speedlimit are both non-
    negative integers
    Returns 0 if not speeding; otherwise, returns the fine
    """

    if speed <= speedlimit:
        return 0
    else:
        # calculate the fine
        mphOver = speed - speedlimit
        fine = 50 + mphOver * 5

        # excessive speed
        if speed > 90:
            fine = fine + 200

        return fine
```

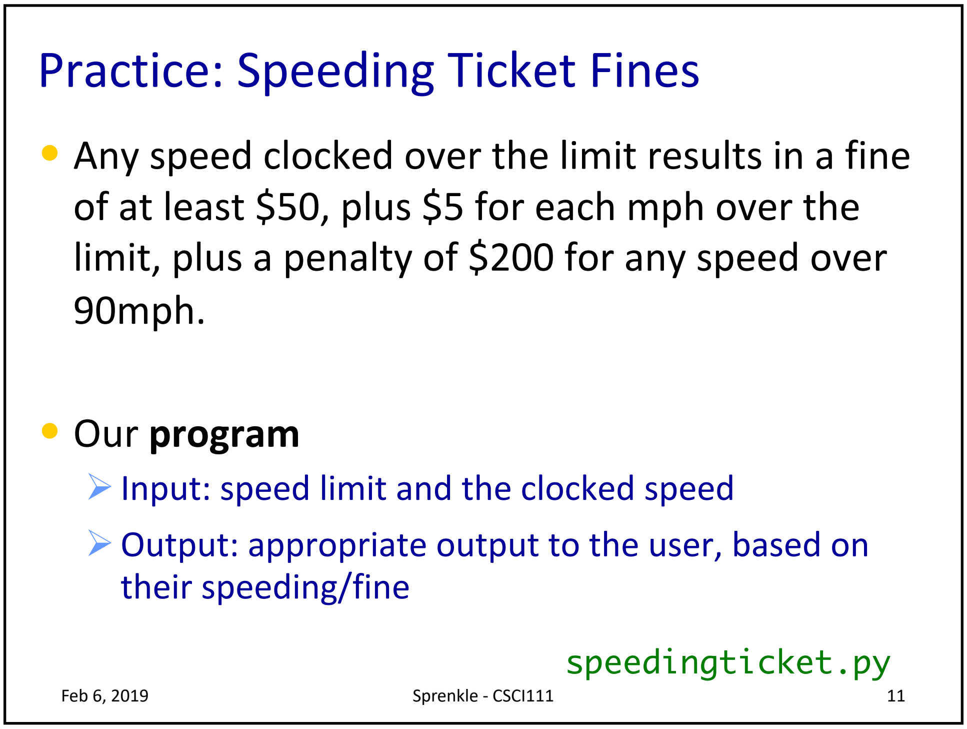
Using the building blocks: Nesting `if-else` statements



```
if condition :
    if condition :
        statements
    else:
        statements
else:
    statements
    if condition :
        statements
    else:
        statements
```

`if-else` statement is nested inside the `if`

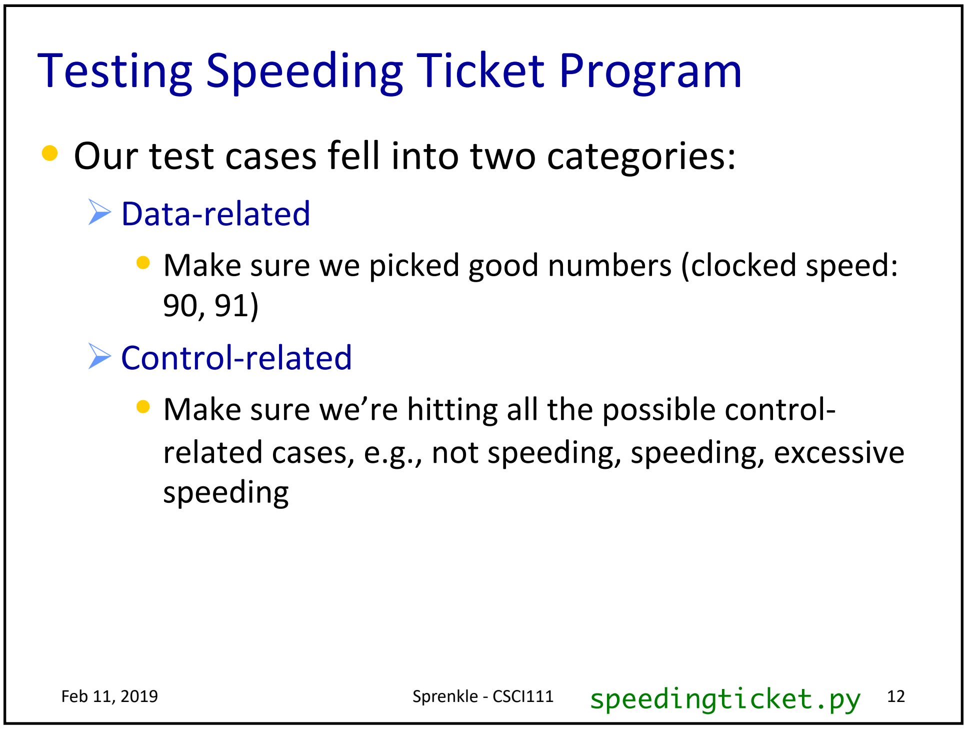
`if-else` statement is nested inside the `else`



Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least $50, plus $5 for each mph over the limit, plus a penalty of $200 for any speed over 90mph.
- Our **program**
  - Input: speed limit and the clocked speed
  - Output: appropriate output to the user, based on their speeding/fine

`speedingticket.py`



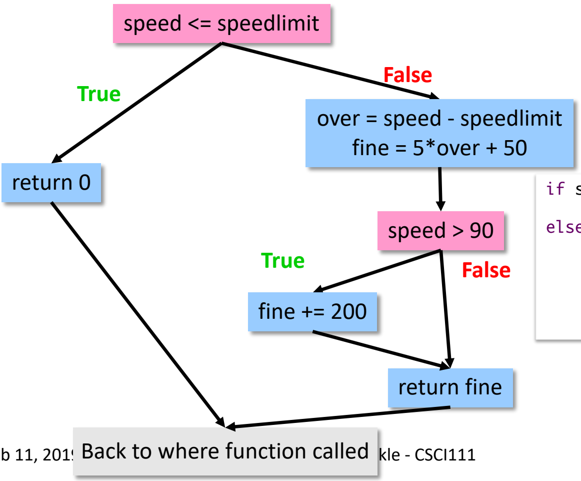
Testing Speeding Ticket Program

- Our test cases fell into two categories:
  - Data-related
    - Make sure we picked good numbers (clocked speed: 90, 91)
  - Control-related
    - Make sure we’re hitting all the possible control-related cases, e.g., not speeding, speeding, excessive speeding

`speedingticket.py`

Testing with `if` Statements

- Make sure *at least* have test cases that execute each branch in control flow diagram
    - i.e., Each execution path is "covered"

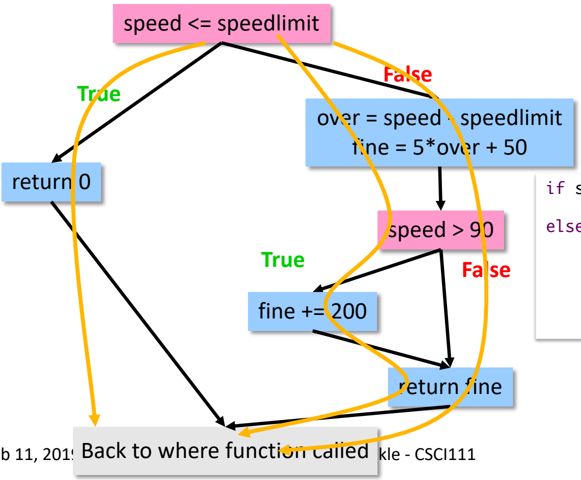


Three execution paths

```
if speed <= speedlimit:
    return 0
else:
    diff = speed - speedlimit
    fine = 50 + 5 * diff
    if speed > 90:
        fine += 200
    return fine
```

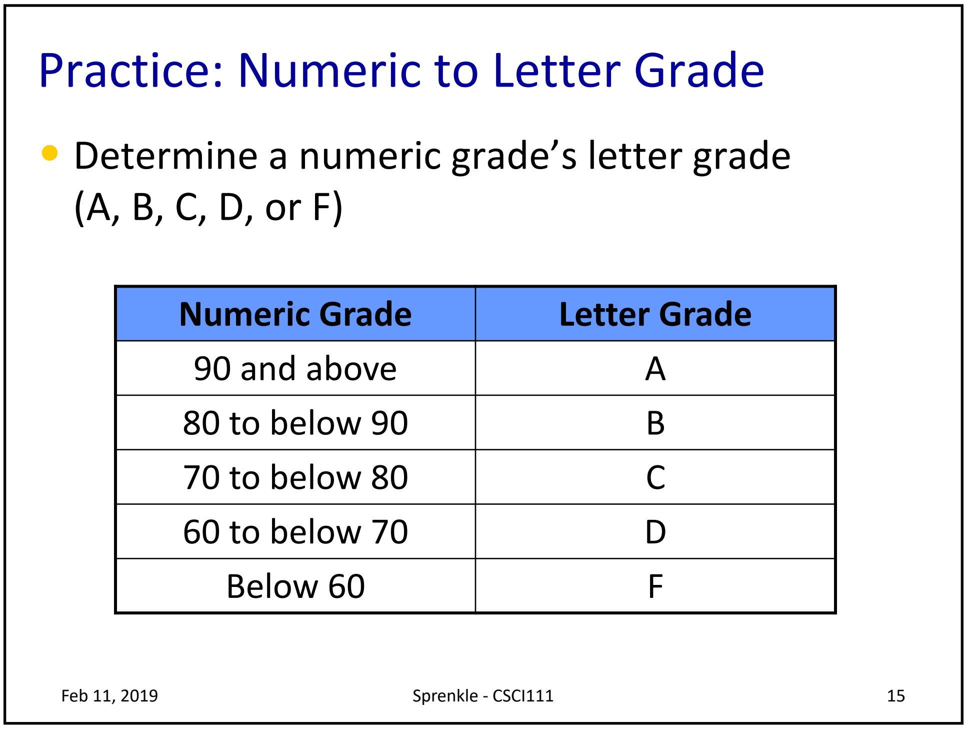
Testing with `if` Statements

- Make sure *at least* have test cases that execute each branch in control flow diagram
    - i.e., Each execution path is "covered"



Three execution paths

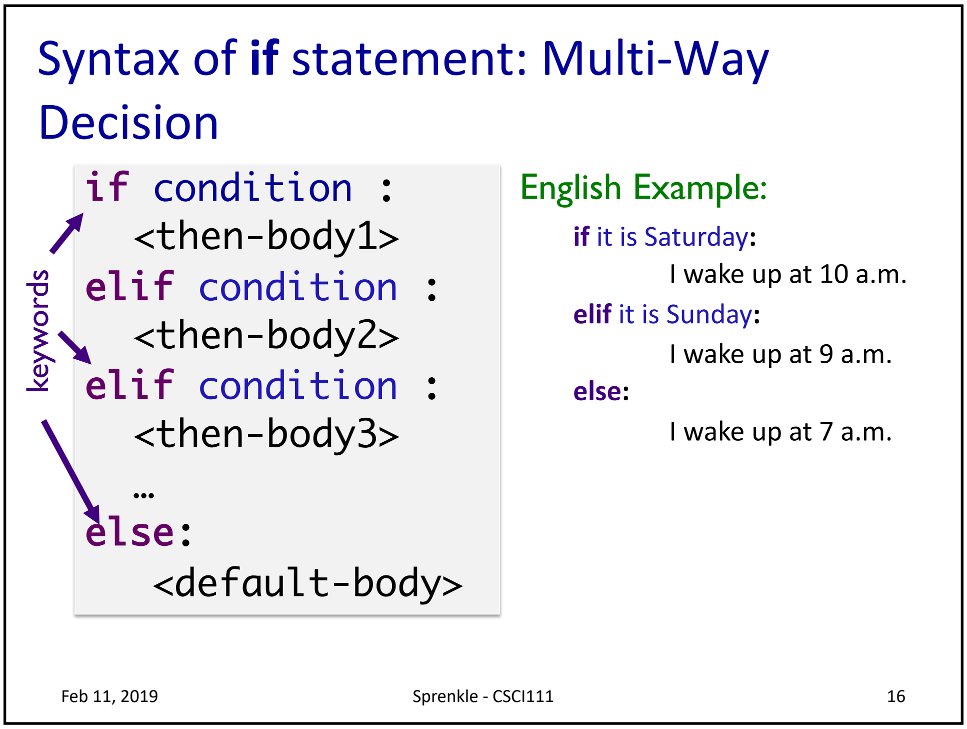
```
if speed <= speedlimit:
    return 0
else:
    diff = speed - speedlimit
    fine = 50 + 5 * diff
    if speed > 90:
        fine += 200
    return fine
```



Practice: Numeric to Letter Grade

- Determine a numeric grade’s letter grade (A, B, C, D, or F)

Numeric Grade	Letter Grade
90 and above	A
80 to below 90	B
70 to below 80	C
60 to below 70	D
Below 60	F



Syntax of **if** statement: Multi-Way Decision

```
if condition :
    <then-body1>
elif condition :
    <then-body2>
elif condition :
    <then-body3>
    …
else:
     <default-body>
```

keywords

English Example:

if it is Saturday:
    I wake up at 10 a.m.
elif it is Sunday:
    I wake up at 9 a.m.
else:
    I wake up at 7 a.m.

Using the building blocks: Nesting `if-else` statements

```
if condition:
   statements
else:
   if condition:
      statements
   else:
      statements
```

`if-else` statement is nested inside the `else`

This structure can be rewritten as an if-elif-else statement

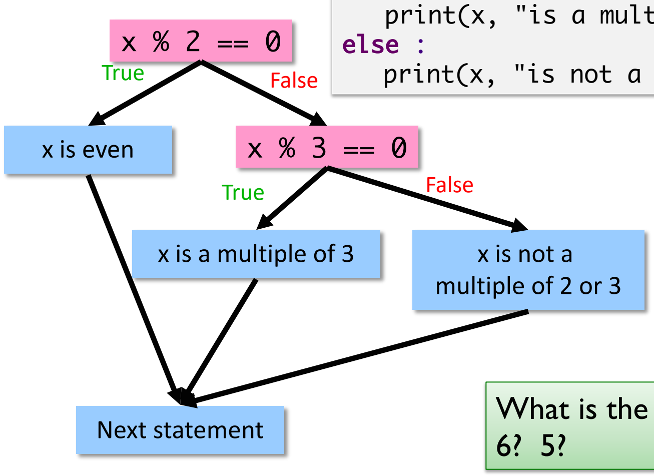
If-Else-If statements

Draw the control flow diagram

```
if x % 2 == 0 :
   print(x, "is a multiple of 2")
elif x % 3 == 0 :
   print(x, "is a multiple of 3")
else :
   print(x, "is not a multiple of 2 or 3")
```

If-Else-If statements

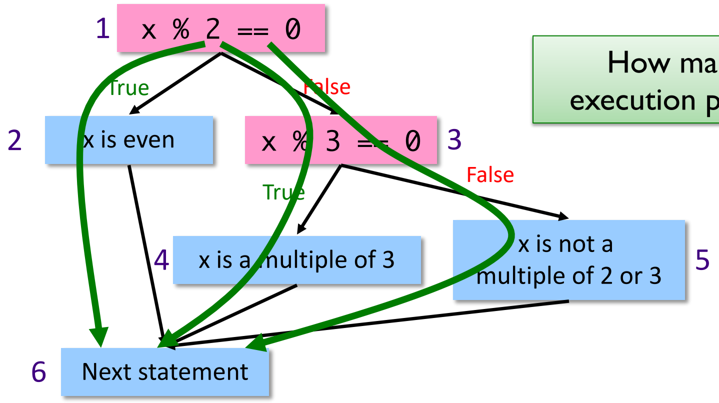
```
if x % 2 == 0 :
   print(x, "is a multiple of 2")
elif x % 3 == 0 :
   print(x, "is a multiple of 3")
else :
   print(x, "is not a multiple of 2 or 3")
```



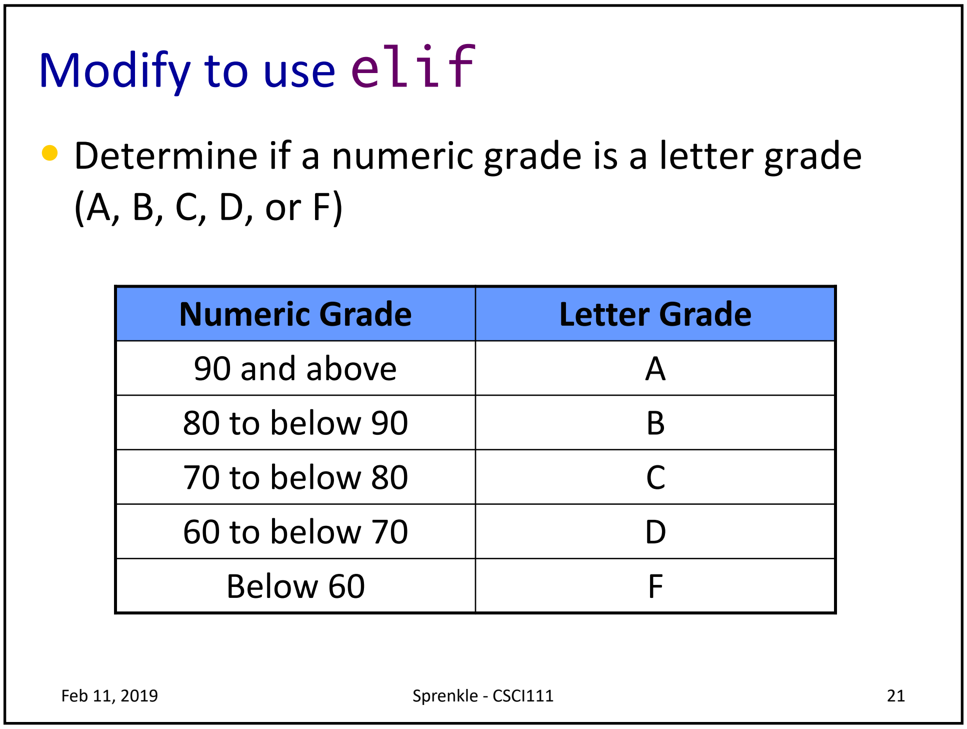
What is the output if x is 4? 6? 5?

Testing with If Statements

- Make sure have test cases that execute each branch in control flow diagram
  - i.e., Each execution path is "covered"



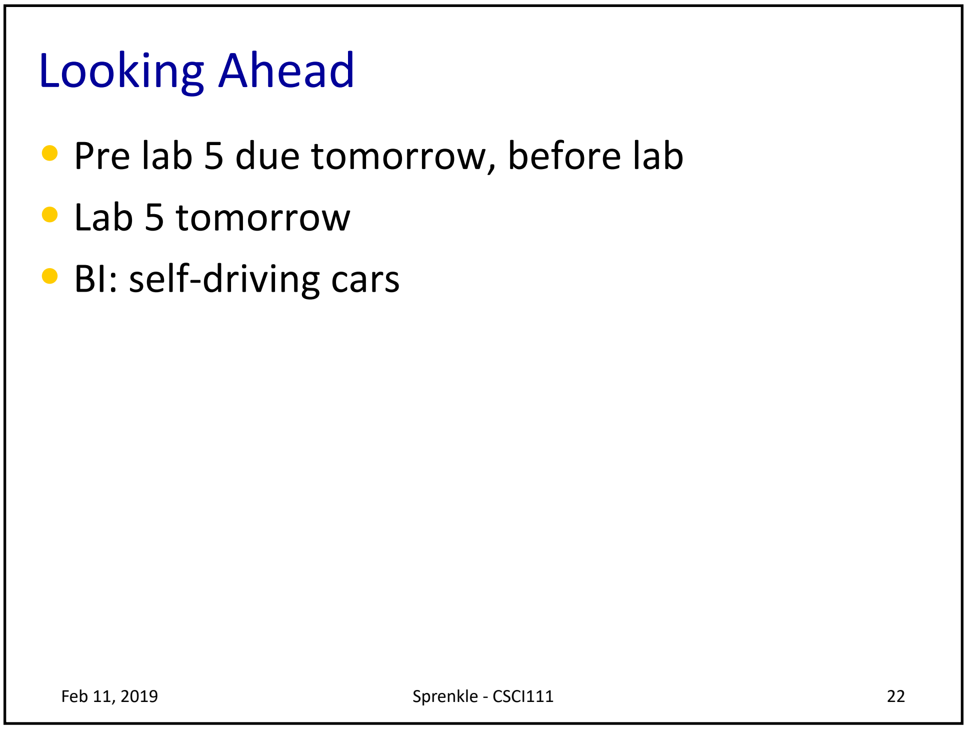
How many execution paths?



Modify to use `elif`

- Determine if a numeric grade is a letter grade (A, B, C, D, or F)

Numeric Grade	Letter Grade
90 and above	A
80 to below 90	B
70 to below 80	C
60 to below 70	D
Below 60	F



Looking Ahead

- Pre lab 5 due tomorrow, before lab
- Lab 5 tomorrow
- BI: self-driving cars