

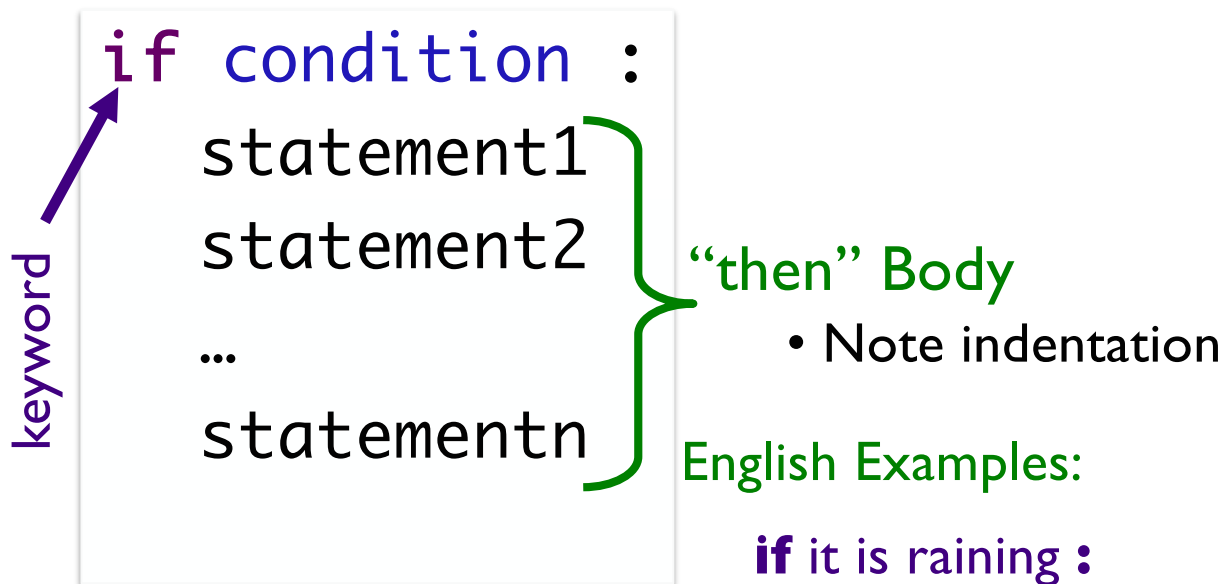
# Objectives

- More Conditionals
- Boolean Operators

# Review

- How can we make Python code execute only under certain circumstances?
  - Describe the syntax and semantics
- How do we say “otherwise” in Python?
- What are relational operators?
  - Provide examples

# Review: Simple Decision

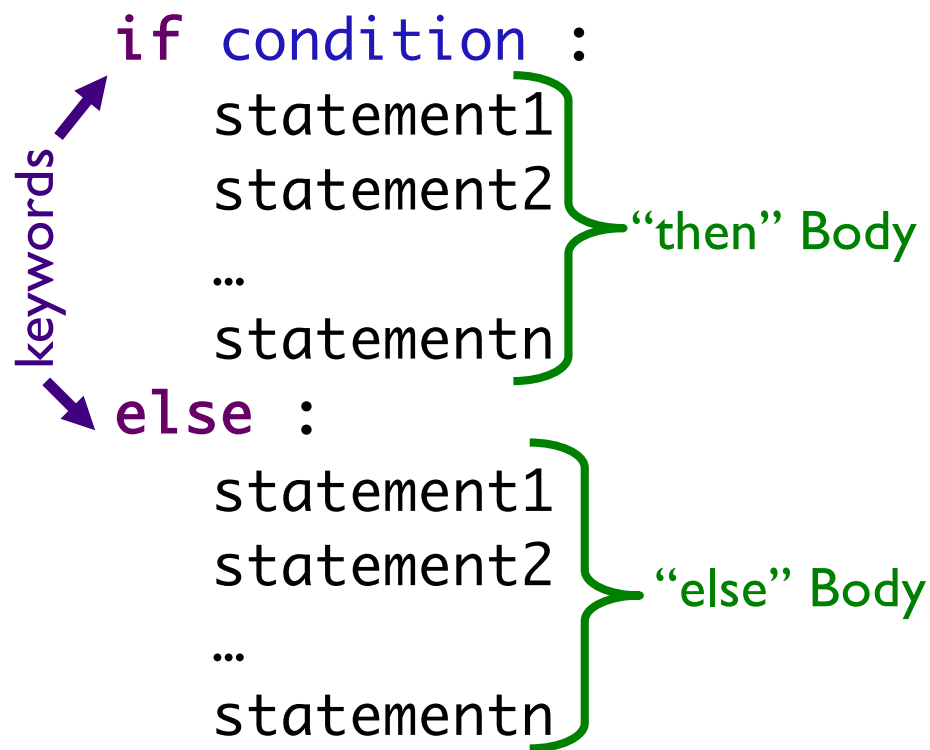


English Examples:

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

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

# Review: Two-Way Decision



## English Example:

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

# Review: Relational Operators

- Syntax: `<expr> <relational_operator> <expr>`
- Evaluates to either `True` or `False`
  - Boolean type

Relational Operator	Meaning
<code>&lt;</code>	Less than?
<code>&lt;=</code>	Less than or equal to?
<code>&gt;</code>	Greater than?
<code>&gt;=</code>	Greater than or equal to?
<code>==</code>	Equals?
<code>!=</code>	Not equals?

Low precedence  
After arithmetic operators

# 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")
```

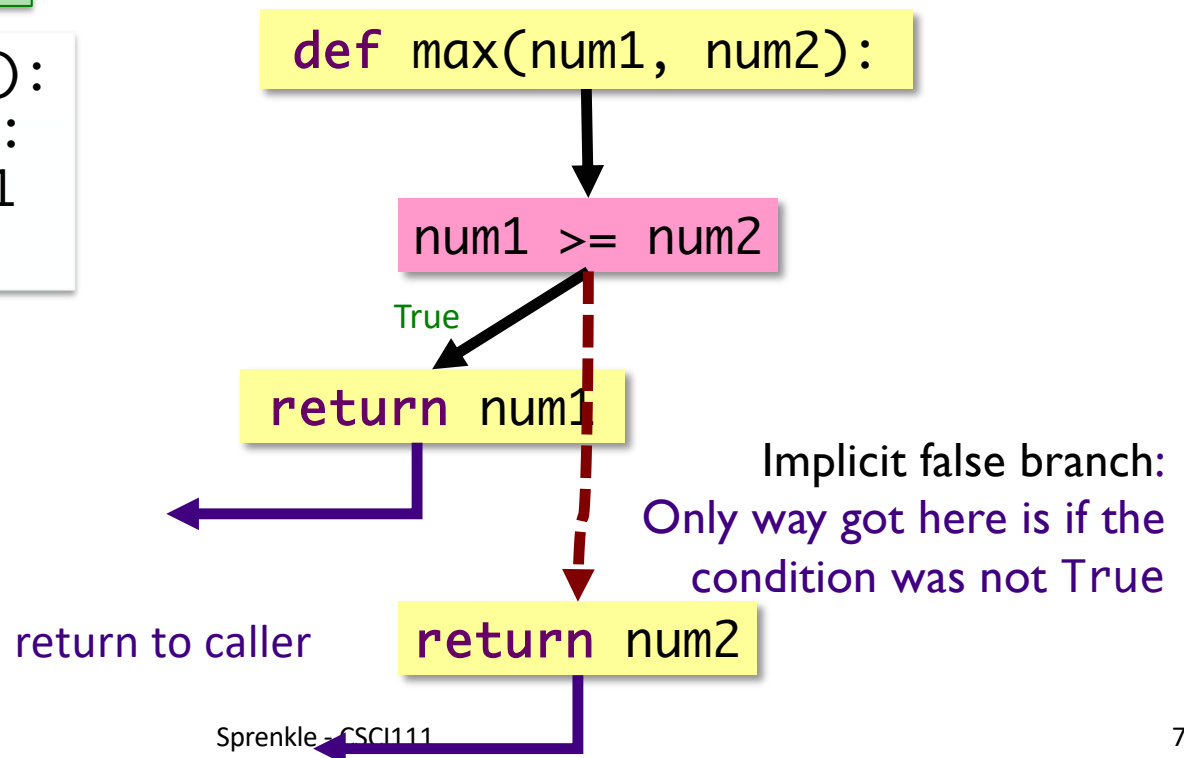
```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

This is the more efficient implementation. Why?

# Review: Flow of Control: Using `return`

Is this implementation of the function correct?

```
def max(num1, num2):  
    if num1 >= num2:  
        return num1  
    return num2
```



# 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 90 mph.
- 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?



# Test-Driven Development (TDD)

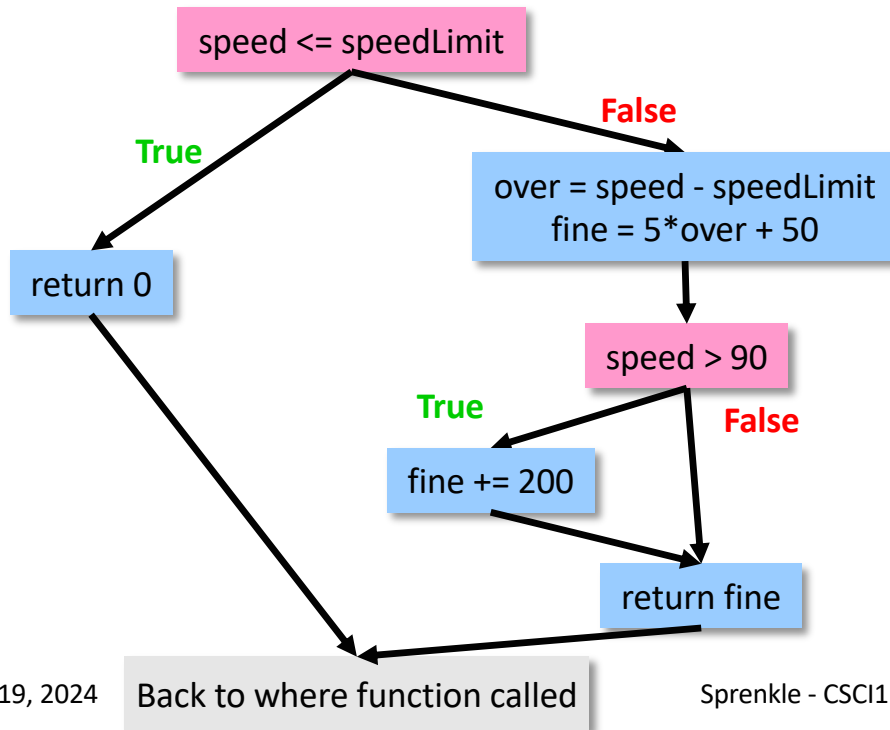
- Create test cases first
- Idea: Focus on the outcomes first
- Helps you think about the problem without thinking about the code itself

# Testing Speeding Ticket Program

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

# 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”

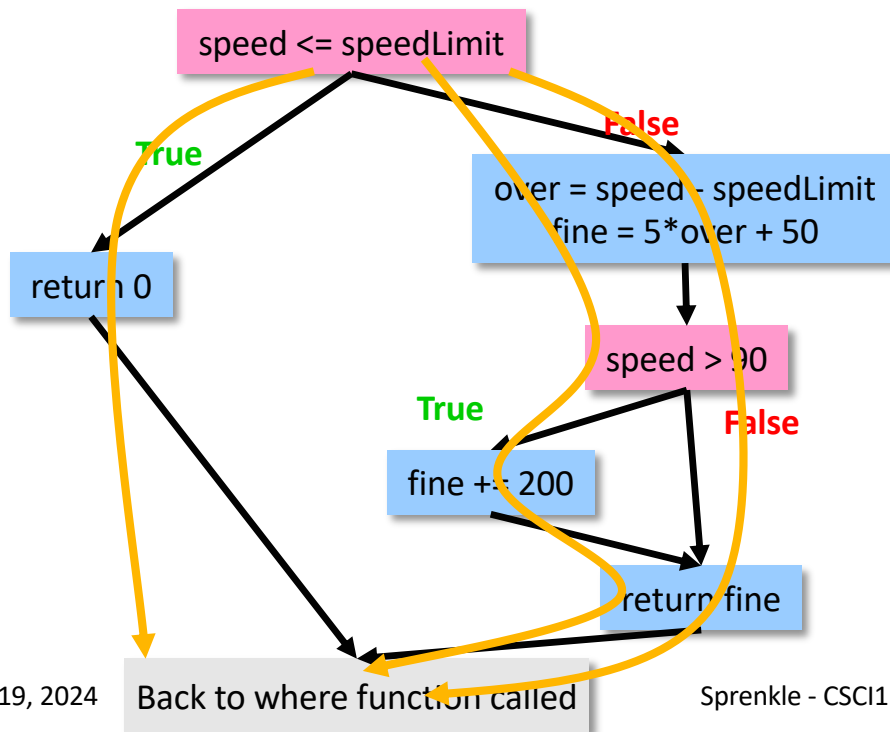


## Three execution paths

```
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: 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

# Practice: Speeding Ticket Fines

- Any speed over the limit is at least \$50, plus a penalty of \$5 for every mile over the limit.

```
def main():  
    print("This program ...")  
  
    clockedSpeed = eval(input("Enter your speed: "))  
    speedLimit = eval(input("Enter the speed limit: "))  
  
    # your code here  
  
def calculateFine(limit, speed):  
    ...
```

- Our program

- Input: speed limit and the clocked speed
- Output: appropriate output to the user, *based on their speeding/fine*

speedingticket.py

# 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: Numeric to Letter Grade

- Write a program to 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

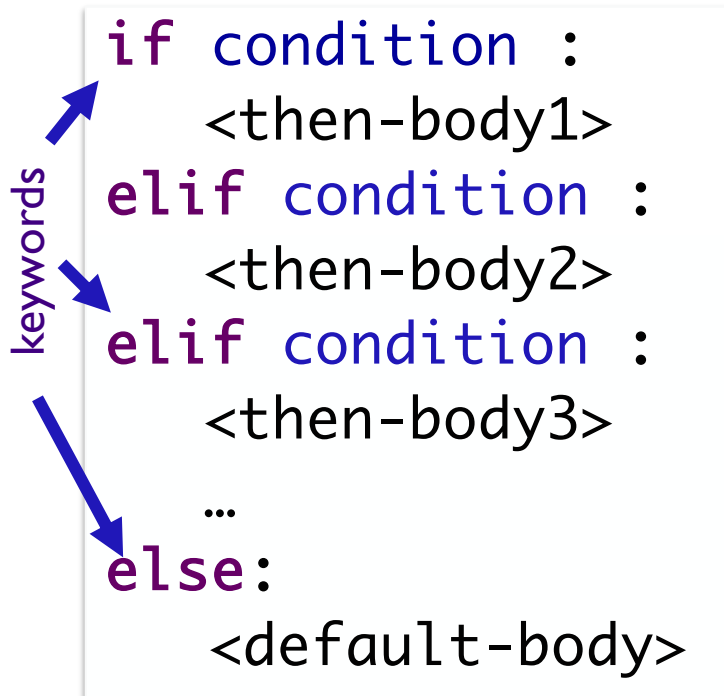
```
numericGrade = float(input("Numeric grade: "))  
# Your code here...  
print("Your grade is", letterGrade)
```



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

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

keywords

A diagram showing the syntax of an if statement. The text is enclosed in a light gray box. The keywords 'if', 'elif', and 'else' are highlighted in purple. Blue arrows point from the word 'keywords' on the left to each of these three keywords. The rest of the code is in black text.

## 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

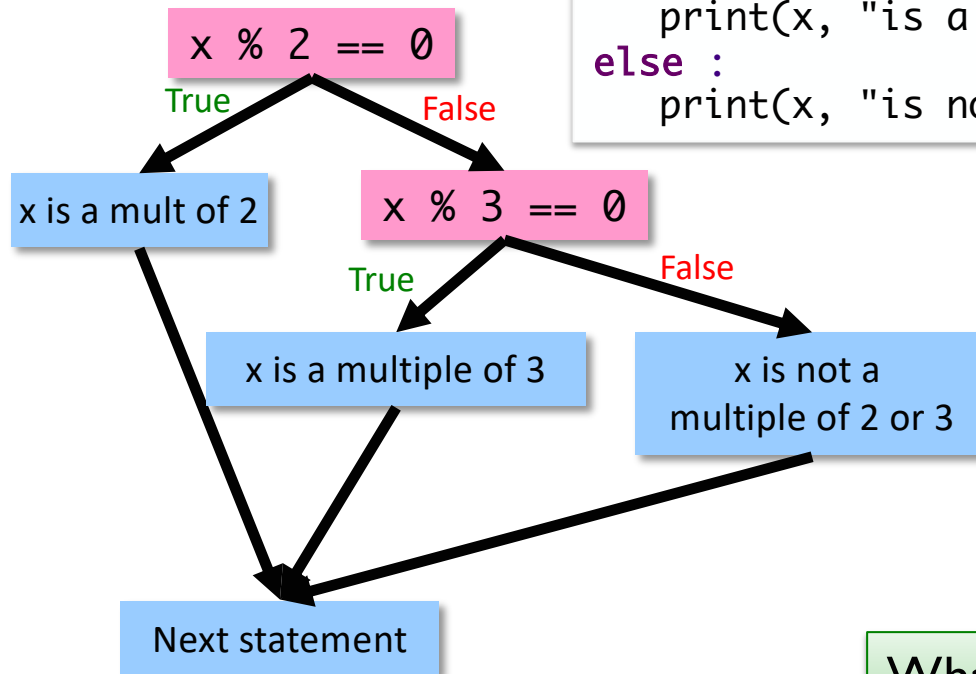
# If-Else-If statements

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

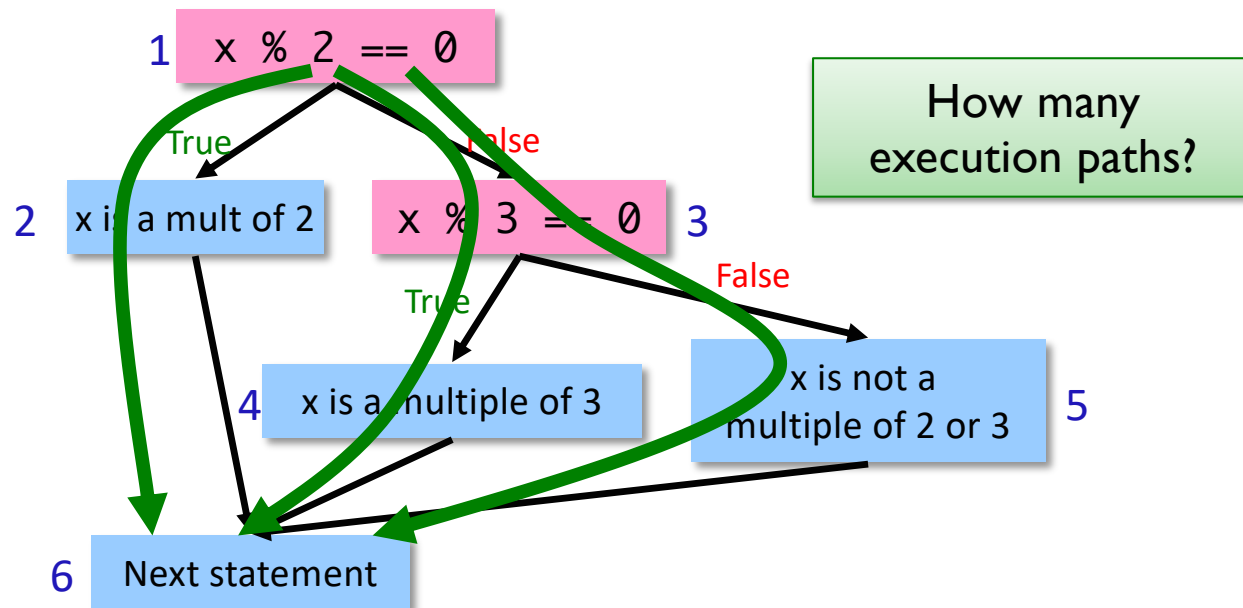
```
if x % 2 == 0 :  
    print(x, "is a multiple of 2")  
elif x % 3 == 0 :  
    print(x, "is a multiple of 3")  
else :  
    print(x, "is not a multiple of 2 or 3")
```



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”



## 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: what can tech companies do?