Lab 6

- Review Lab 5
- Review indefinite loops, strings
- Lab 6

Common Issue: Inefficiency

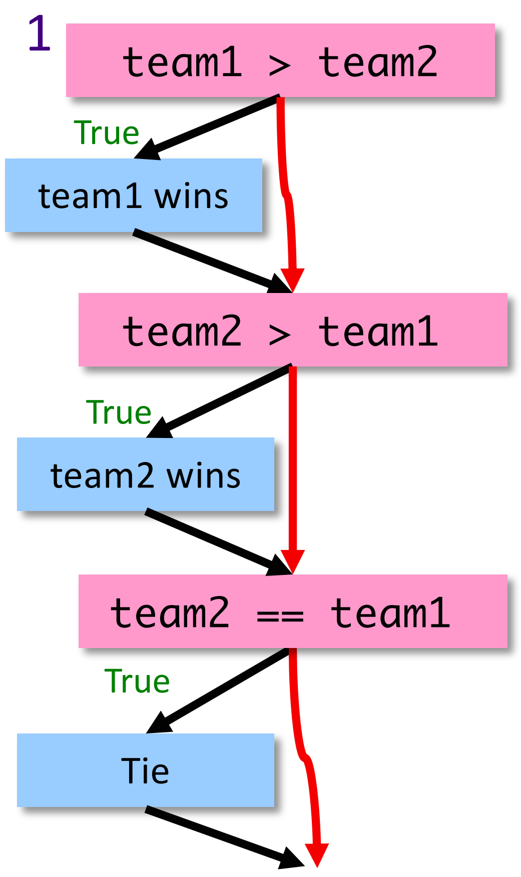
```python
if team1Score > team2Score:
    print("Team 1 wins!")
else:
    if team2Score > team1Score:
        print("Team 2 wins!")
    else:
        if team1Score == team2Score:
            print("They tied! We're going to overtime!")
```

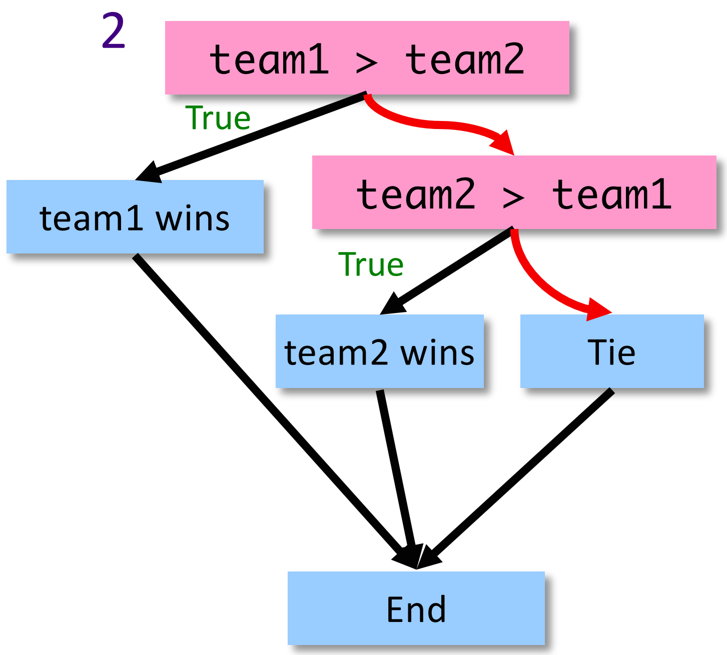
Extra if statements, not necessary

Know when hit second else that the only possibility is a tie

```python
if team1Score > team2Score:
    print("Team 1 wins!")
else:
    if team2Score > team1Score:
        print("Team 2 wins!")
    if team1Score == team2Score:
        print("They tied! We're going to overtime!")
```

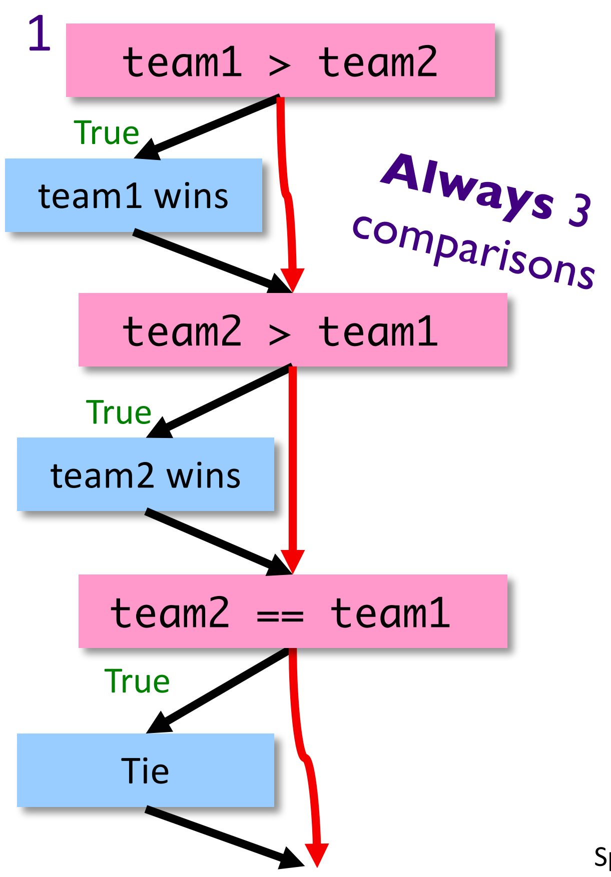
Problem 1, 2 Efficiency

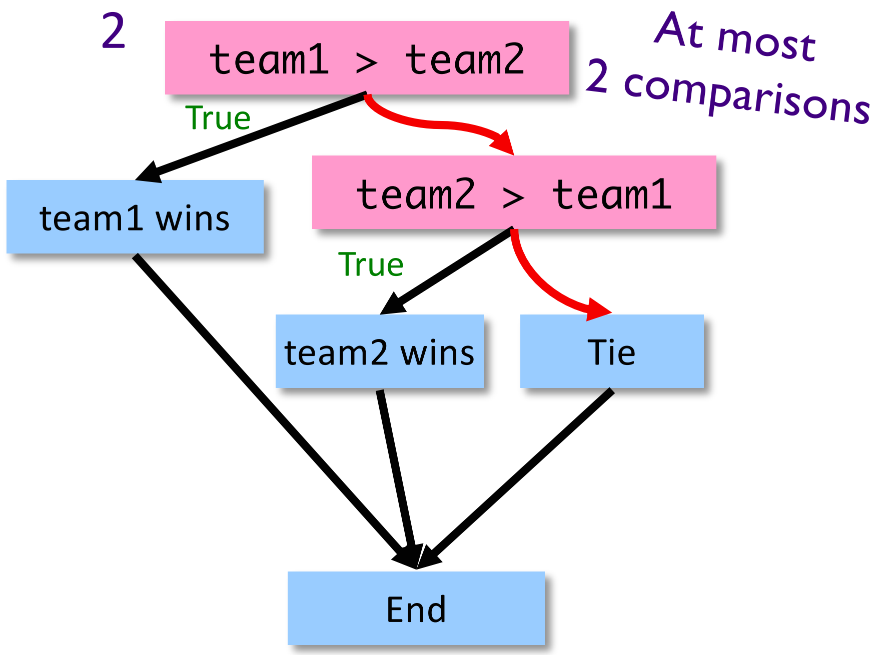




How many conditions are evaluated?

Problem 1, 2 Efficiency





Lab 5 – Greatest Hits: Less-Complicated Approaches for Customized Display

- Correct but complicated solution to handling customized display

Other, similar examples in submissions

```
if albums == 1 and extraTracks == 0:
    print("Your album requires", albums, "cd")
elif albums == 1 and extraTracks > 0:
    print("Your album requires", albums, "cd")
    print(extraTracks, "tracks will have to wait for the next …")
elif albums > 1 and extraTracks > 0:
    print("Your album requires", albums, "cds")
    print(extraTracks, "tracks will have to wait for the next …")
elif albums > 1 and extraTracks == 0:
    print("Your album requires", albums, "cds")
```

Lab 5 – Greatest Hits: Less-Complicated Approaches for Customized Display

- Less complicated solution
  - Simpler logic, conditions
  - Less duplicated code

```
if albums == 1:
    print("Your album requires", albums, "CD.")
else:
    print("Your album requires", albums, "CDs")

if extraTracks > 1:
    print(extraTracks, "tracks will have to wait for the next …")
elif extraTracks==1:
    print(extraTracks, "track will have to wait for the next …")
```

Relational Operators

- Reminder: instead of, for example,

```
num < 0 or num > 0
```

can use

```
num != 0
```

Super Bowl Simulation

- Constants
  - **Not** user inputs
  - Named using all caps
  - Located near the top of your program
    - After high-level comments and import statements

```
# high-level comment
# authorship

import …

CONSTANTS = …

Functions or code…
```

Super Bowl Extensions

A lot you could add already;
even more with a little more knowledge

- Simulate scores (rather than the difference)
- Dynamically change odds based on various stats
- Simulate playoff structure
- Today: could simulate a World Series that plays games until a team reaches four wins. How? (EC)

Design of Super Bowl Simulation

hasFavoredTeamWon

- Function: `hasFavoredTeamWon`
  - Specializes in determining if the favored team won
  - Could implement function differently
    - Examples: always return True (or False); simulate playing the game, getting touchdowns, field goals, safeties, … and determine the winner
- If the implementation of the function changes *and* its interface does not change, the `main` function does not need to change
  - Power of abstraction, separation of concerns
  - Helps to isolate changes

Review: Indefinite Loops

- What is the syntax for an indefinite loop?
- Which is more powerful: a `for` loop or an indefinite loop?
- After determining that a problem requires a loop, what are the questions to ask?
- What are the two ways to think about indefinite loop problems?

while Loops: Alternative Approaches

```
# condition says when loop
# will continue
x=eval(input("Enter number:"))
while x % 2 != 0 :
    print("Error!")
    x = eval(input("Enter number: "))
print(x, "is an even number.")
```

Loop condition says when to keep going

```
# have to look inside loop to
# know when it stops
while True :
    x = eval(input("Enter number:"))
    if x % 2 == 0 :
        break
    print("Error!")
print(x, "is an even number.")
```

"breaks" out of a loop

Internal condition says when to stop

Using break statements: Best when loop has to execute at least once.

str Review

- How can we combine strings?
  - How can we repeat a string multiple times?
- How can we find out how long a string is?
- How can you tell if one string is contained in another string?
- How can we find out the character at a certain position?
- How can we extract a substring from a string?
- How can we iterate through a string? (two ways)
- How do you call a method on a string?
- How do you find out what methods are available for strings?
- Can you change a string after it has been created?
- What is the accumulator design pattern?

Methods vs Functions

Functions

- Associated with a file or module
- All input comes from arguments/parameters
- Example: `len` is a built-in function
  - Called as `len(strobj)`

Methods

- Associated with a *class* or *type*
- Input comes from arguments ***and*** the string the method was called on
- Example:
  - `strobj.upper()`

Revised Pick4 Game

- To play: pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Previously: Simulated the magic ping-pong ball machines
- Additional Functionality:
  - ➢ Determine if the user picks the winning number
    - Why couldn't we solve this before?
      - ➢ What are valid choices for numbers?

`pick4winner.py`

Lab 6

- Advanced conditions
- Indefinite Loops
- Text-based problems

You do not need to write functions
if I do not explicitly require functions.