Objectives

- Indefinite Loops
- Broader Issues

Review: What Will This Loop Do?

```
count = 1
while count > 0:
   print count
   count += 1
```

Infinite Loop Questions

- Is there ever a time that an infinite loop is wanted?
  - Yes! For example in web servers, we have something like

```
while True:
        listenForRequest()
        handleRequest()
```

- Can a computer automatically detect infinite loops?
  - No that is an **undecidable** problem
  - Best to **prevent** infinite loops (more later)
    - Benefit of Python's **`for`** loops: definite loops

Unknown Number of Iterations

- Sums numbers input by user
  - Stop when the user inputs some designated stop value (**`enter`** key --> "")

sumtillzero.py

Design Pattern: Sentinel Loop

- Sentinel: when to stop
  - "guard" to the loop

```
value = get input
while value != sentinel :
    process value
    value = get input
```

Question

- How can we make sure that the loop actually stops (is not infinite)?

Question

- How can we make sure that the loop actually stops (is not infinite)?

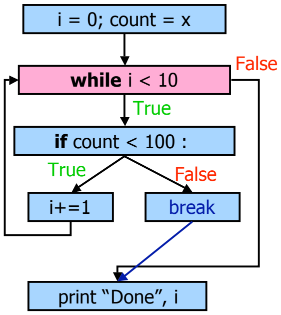
  1. Update the condition’s variable inside loop
  2. Test

- How you’ll usually detect an infinite loop…
  - “Why isn’t my program giving me any output?”
  - If the program also isn’t exiting, probably an infinite loop

Use of **break** statement

- **break** statement can “break you” out of a loop

```
i=0
count = x
while i < 10 :
    if count < 100 :
        i += 1
    else:
        break
print “Done”, i
```



while Loops: comparing use of **break**

```
# condition shows when loop
# will stop executing
x= input(“Enter a number: ”)
while x % 2 != 0 :
    x = input(“Try again.  Enter
    a number: ”)
print x, “ is an even number.”
```

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

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

While vs. For Loops

- Any **for** loop can be translated into a **while** loop
  - Not vice versa

- **while** loops are more powerful than **for** loops
  - Give an example of a **while** loop that can’t be converted to a **for**

Summary of Control-Flow Building Blocks (so far)

- Conditional statements
  - if, if-else, if-elif-else
- Loops
  - while, for

Nondeterministic Decisions

- Sometimes, we don’t want to necessarily know that a specific decision is always made
- For example, games often use randomness to make decisions
  - Roll dice
  - Coin flips
  - Location and behavior of baddies

Flipping Coins

- Simulate by randomly selecting between 0 (heads) and 1 (tails)
- Program: coinFlip.py
- Problem: How many flips does it take to get 3 consecutive heads?

consecutiveHeads.py

Broader Issue: DARPA Urban Challenge

- Challenge: automated cars in an urban setting
  - Deal with human drivers, automated drivers
  - Correctly obey traffic laws
  - Winners: 1st - $2Mill, 2nd - $1Mill, 3rd - $500K
  - Apply for $1Million in “seed money”

Groups:
- Business (Andrew, Joa, Stuart, Alex)
- Physics & Engineering (Julie, Nay, Ty)
- Undeclared (Vasil, Lucy, Clay, Dave)
- Science & Humanities (Greg, Arturo, Colin, Joe)

DARPA Urban Challenge

- Will you feel safe (safer?) with an automated driver in the lane next to you?
- Are there situations that would be particularly difficult for software to handle that a person would be better equipped to handle?
- **Student Question:** What should the next DARPA Challenge be?
  - In a year?
  - In 5 years?

Relation To Our Class

- One IF statement from victory
  - Programming you’re learning applied to major tasks
- Importance of testing
  - Need robustness, reliability of systems
  - Test the “small”