Objectives

- Indefinite Loops
- Exam review

Lab Review

- 1 “Challenge” problem
- 1 Application problem

Indefinite Loops

- `for` loops are ***definite*** loops
  - Execute a *fixed* number of times
- Indefinite loops: keeps iterating until certain conditions are met
  - Depending on condition, no guarantee in advance of how many times the loop body will be executed

---

While Loop Syntax

```
while condition :
    statement1
    statement2
    …
    statementn
```

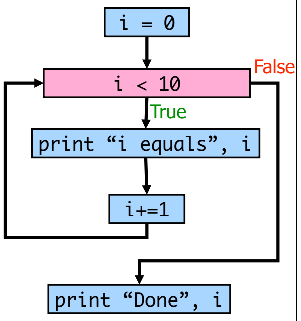
keyword

body of while loop

- Like a looped `if` statement
  - Execute statements **only** when condition is true

While Loop

```
i = 0
while i < 10 :
    print “i equals ”, i
    i+=1
print “Done”, i
```



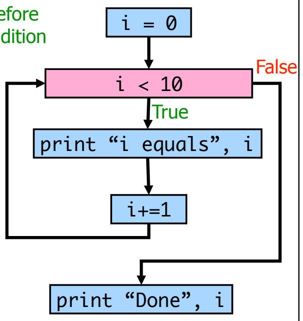
Questions:
- How many times will “i” get printed out?
- How many times is the condition evaluated?
- What is the value of i after the loop?

while.py

While Loop

```
i = 0
while i < 10 :
    print “i equals ”, i
    i+=1
print “Done”, i
```

Initialize i before using in condition



Questions:
- How many times will “i” get printed out?
- How many times is the condition evaluated?
- What is the value of i after the loop?

while.py

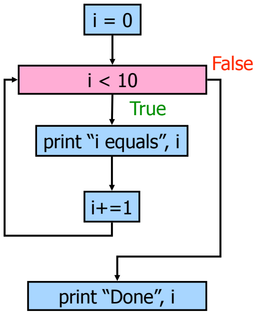
While vs. For Loops

- Any `for` loop can be translated into a `while` loop
  - **Not** vice versa
- `while` loops are more **powerful** than `for` loops

Convert to a `for` loop

We can convert this while loop into a for loop because it executes a ***fixed*** number of times.

```
i = 0
while i < 10 :
    print “i equals ”, i
    i+=1
print “Done”, i
```



Comparing while and for

- What are the main differences between these loops?
- What are the advantages and disadvantages of each?

```
i = 0
while i < 10 :
    print “i equals ”, i
    i+=1
print “Done”, i
```

```
for i in xrange(10):
    print "i equals", i

print "Done", i
```

whilevsfor.py

What Will This Loop Do?

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

loop.py

Infinite Loop

- Condition will never be False so keeps executing

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

- To stop an executing program in Linux use
  - Control-C

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

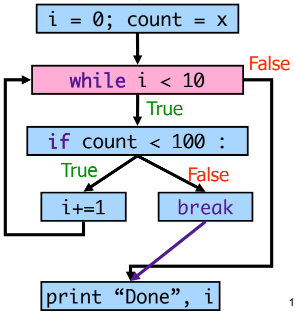
- How can we make sure that the loop actually stops (is not infinite)?
  - Update the condition’s variable inside loop
  - Test
- How you’ll usually detect an infinite loop…
  - “Why isn’t my program giving me any output?”
  - If the program also isn’t exiting, probably an infinite loop

---

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

Review: String Formatting

- What does this do?

```
print “%.2f deg F is %.2f deg C” % (degreesF, degreesC)
```

Review: String Formatting

- Equivalent Statements

```
print “%.2f deg F is %.2f deg C” % (degreesF, degreesC)
```

```
print “%.2f” % degreesF, “deg F is %.2f deg C” % degreesC)
```

```
print “%.2f” % degreesF, “deg F is”, “%.2f” % degreesC, “deg C”
```

Midterm Prep

- Cumulative up to today
  - We keep using the ideas from the first day of class
  - Basic Linux commands used during every lab
- Similar problems as in handouts, class discussion, labs
- Read code and explain what it does
  - What it displays as output
- Sections: Very Short Answer, Short Answer, Write Code
- Online prep document

Grading Overview

- Labs: 38%
- 2 Exams: 29%
- Final: 20%
- Broader Issues: 8%
- Participation: 5%