ojoen.michigan

Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/.

Copyright © 2009, Charles Severance.

You assume all responsibility for use and potential liability associated with any use of the material. Material contains copyrighted content, used in accordance with U.S. law. Copyright holders of content included in this material should contact open.michigan@umich.edu with any questions, corrections, or clarifications regarding the use of content. The Regents of the University of Michigan do not license the use of third party content posted to this site unless such a license is specifically granted in connection with particular content. Users of content are responsible for their compliance with applicable law. Mention of specific products in this material solely represents the opinion of the speaker and does not represent an endorsement by the University of Michigan. For more information about how to cite these materials visit http://michigan.educommons.net/about/terms-of-use.

Any medical information in this material is intended to inform and educate and is not a tool for self-diagnosis or a replacement for medical evaluation, advice, diagnosis or treatment by a healthcare professional. You should speak to your physician or make an appointment to be seen if you have questions or concerns about this information or your medical condition. Viewer discretion is advised: Material may contain medical images that may be disturbing to some viewers.





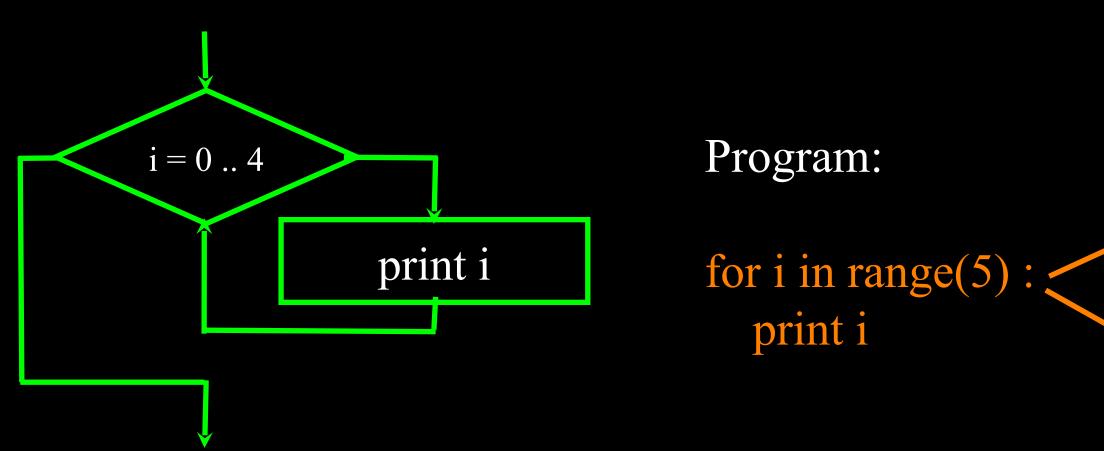
Loop Structures and Booleans Zelle - Chapter 8

Charles Severance - www.dr-chuck.com

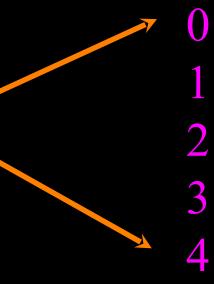
Textbook: Python Programming: An Introduction to Computer Science, John Zelle



Repeated Steps



Output:





Definite Loops

Definite Loops

- Loops that run a fixed (aka definite) number of times
- Loops that "iterate" through an ordered set
- Loops that run "for" a number of times

for abc in range(5): print "Hi" print abc

Hi Hi Hi 2 Hi 3 Hi 4

Z-39

Definite Loops

- Loops that run a fixed (aka definite) number of times
- Loops that "iterate" through an ordered set
- Loops that run "for" a number of times

for abc in range(5) : print "Hi" print abc

Colon (:) defines the start of a block. Indenting determines which lines belong to the block.

Hi $\mathbf{0}$ Hi Hi 2 Hi 3 Hi 4 Z-39

Looking at In...

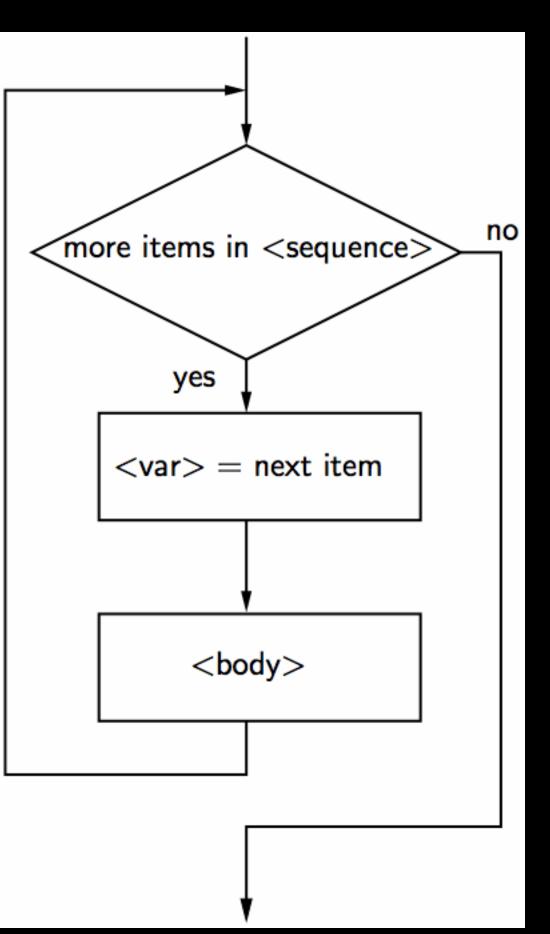
- The iteration variable "iterates" though the sequence (ordered set)
- The block (body) of code is executed once for each value in the sequence
- The iteration variable moves through all of the values in the sequence

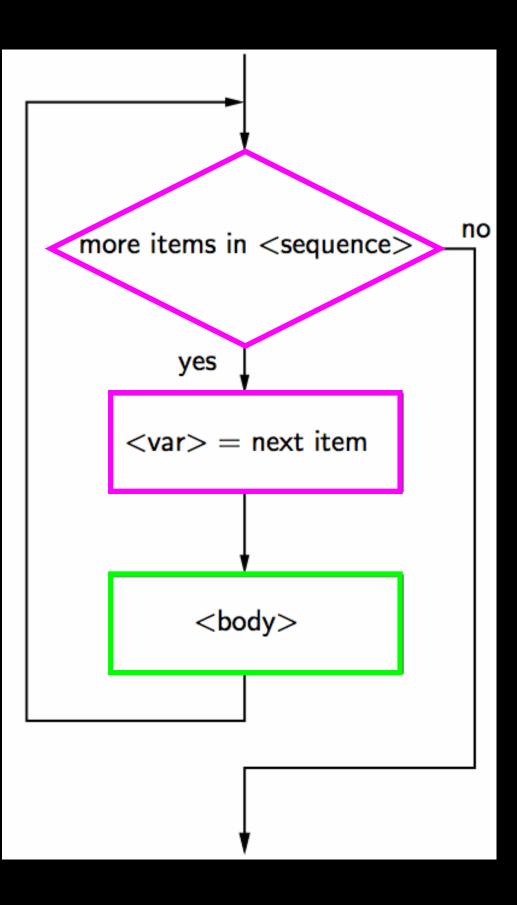
Iteration variable [0, 1, for abc in range(5) : ... block of code ...

Five-element sequence [0, 1, 2, 3, 4]

n a FlowChart

- The iteration variable "iterates" though the sequence (ordered set)
- The block (body) of code is executed once for each value in the sequence
- The iteration variable moves through all of the values in the sequence

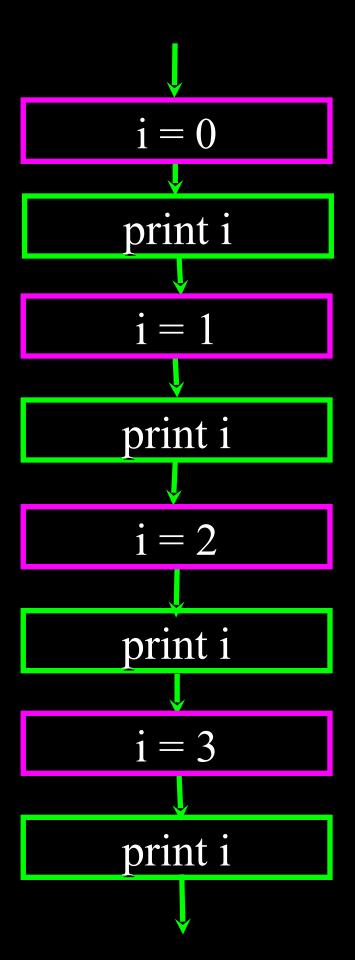




Program:

for i in range(4) : print i

Loop body is run repeatedly



What is range(10)?

- range(10) is a built in function that returns a sequence of numbers
- The for statement can iterate through any sequence
- A sequence can have values of different types

>> range(10) • • • ()

2

• • •

• • •

()

9

2

3.6

abc

>>> for i in [0, "abc", 9, 2, 3.6] : print I

 \overline{Z} -40

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]>>> for i in [0, 1, 2]: print I

File Processing

File Processing

• A text file can be thought of as a sequence of lines

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008 Return-Path: collab.sakaiproject.org> Date: Sat, 5 Jan 2008 09:12:18 -0500To: source@collab.sakaiproject.orgFrom: stephen.marquard@uct.ac.zaSubject: [sakai] svn commit: r39772 content/branches/Details: http://source.sakaiproject.org/viewsvn/? view=rev&rev=39772



Opening a File

- Before we can read the contents of the file we must tell Python which file we are going to work with and what we will be doing with the file
- This is done with the open() function
- open() returns a "file handle" a variable used to perform operations on the file
- Kind of like "File -> Open" in a Word Processor



Using open()

- handle = open(filename, mode)
 - returns a handle use to manipulate the file
 - filename is a string
 - mode is "r" if we are planning on reading the file and "w" if we are going to write to the file.

http://docs.python.org/lib/built-in-funcs.html

fhand = open("mbox.txt", "r")

Z-108

File Handle as a Sequence

- A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence
- We can use the for statement to iterate through a sequence
- Remember a sequence is an ordered set

xfile = open("mbox.txt", "r")

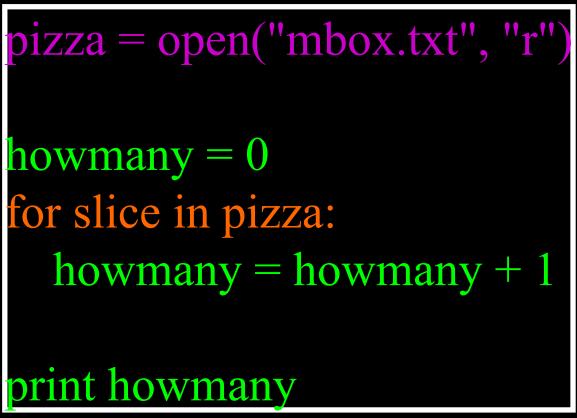
for cheese in xfile: print cheese

Counting Lines in a File

- Open a file read-only
- Use a for loop to read each line
- Count the lines and print out the number of lines

howmany = 0

print howmany





What We Do in Loops Note: Even though these examples are simple the patterns apply to all kinds of loops

Patterns in Loops

- Counting in loops
- Summing in loops
- Averaging in loops
- Searching in loops
- Detecting in loops

• Largest or smallest

Using break in a loop

Using Continue in a loop

Looping through a Set

print "Before" for thing in [3, 41, 12, 9, 74, 15]: print thing print "After"

\$ python basicloop.py

What is the Largest Number?





What is the Largest Number?

What is the Largest Number?

largest_so_far



Making "smart" loops

The trick is "knowing" something about the whole loop when you are stuck writing code that only sees one entry at a time

()

Look for something or do something to each entry separately, updating a variable.

Favorite dog food...



Set some variables to initial values

For thing in data:

Look at the variables.

Finding the largest value

Largest = -1	\$ py
print "Before", largest	Befo
For value in [3, 41, 12, 9, 74, 15]:	33
if value > largest:	41 4
largest = value	41 1
print largest, value	41 9
	74 7
Print "After", largest	74 1

We make a variable that contains the largest value we have seen so far. If the current value is larger, it becomes the new largest value we have seen so far.

ython largest.py ore -1

Counting in a Loop

zork = 0	\$ pytho
print "Before", zork	Before
for thing in [3, 41, 12, 9, 74, 15] :	13
zork = zork + 1	2 41
print zork, thing	3 12
print "After", zork	49
	574
	6 1 5
	After 6

To count how many times we execute a loop we introduce a counter variable that starts at 0 and we add one to it each time through the loop.

on countloop.py

Summing in a Loop

zork = 0	\$ pytho
print "Before", zork	Before
for thing in [3, 41, 12, 9, 74, 15]:	33
zork = zork + thing	44 41
print zork, thing	56 12
print "After", zork	65 9
	139 74

154 15 After 154

To add up a value we encounter in a loop, we introduce a sum variable that starts at 0 and we add the value to the sum each time through the loop.

on countloop.py

Finding the Average in a Loop

count = 0Before **0 0** sum = 0133 print "Before", count, sum 2 44 41 for value in [3, 41, 12, 9, 74, 15] : 3 56 12 count += 14 65 9 sum += value 5 139 74 print count, sum, value 6 1 5 4 1 5 print "After", count, sum, sum / count

> An average just combines the counting and sum patterns and divides when the loop is done.

\$ python averageloop.py

After 6 1 54 25

Searching in a Loop

print "Before" for value in [3, 41, 12, 9, 74, 15] : if value > 20: print "Large number", value print "After"

\$ python search1.py Before Large number 41 Large number 74 After

We use an if statement in the loop to catch the values we are looking for.

Did we encounter a value?

	\$ pytho
found = 0	Before
print "Before", found	03
for value in [3, 41, 12, 9, 74, 15] :	041
if value $== 9$:	012
found = 1	19
print found, value	174
print "After", found	1 15
	A ftor 1

If we just want to search and know if a value was found - we use a variable that starts at zero and is set to one as soon as we find what we are looking for.

on search1.py

Using a **Boolean** Variable

found = False print "Before", found for value in [3, 41, 12, 9, 74, 15] : if value == 9: found = True print found, value

print "After", found

False 3 False 41 False 12 True 9 True 74 True 15

If we just want to search and know if a value was found - we use a variable that starts at zero and is set to one as soon as we find what we are looking for.

- \$ python search1.py Before False
- After True

Remembering where...

found = False where = -1count = 0print "Before", found for value in [3, 41, 12, 9, 74, 15] : count = count + 1if value == 9: found = True where = count print found, where, value

print "After", found, where

- False -1 3
- **False -1 41**
- False -1 12
- True 4 9
- True 4 74
- True 4 15
- After True 4

\$ python search1.py Before False

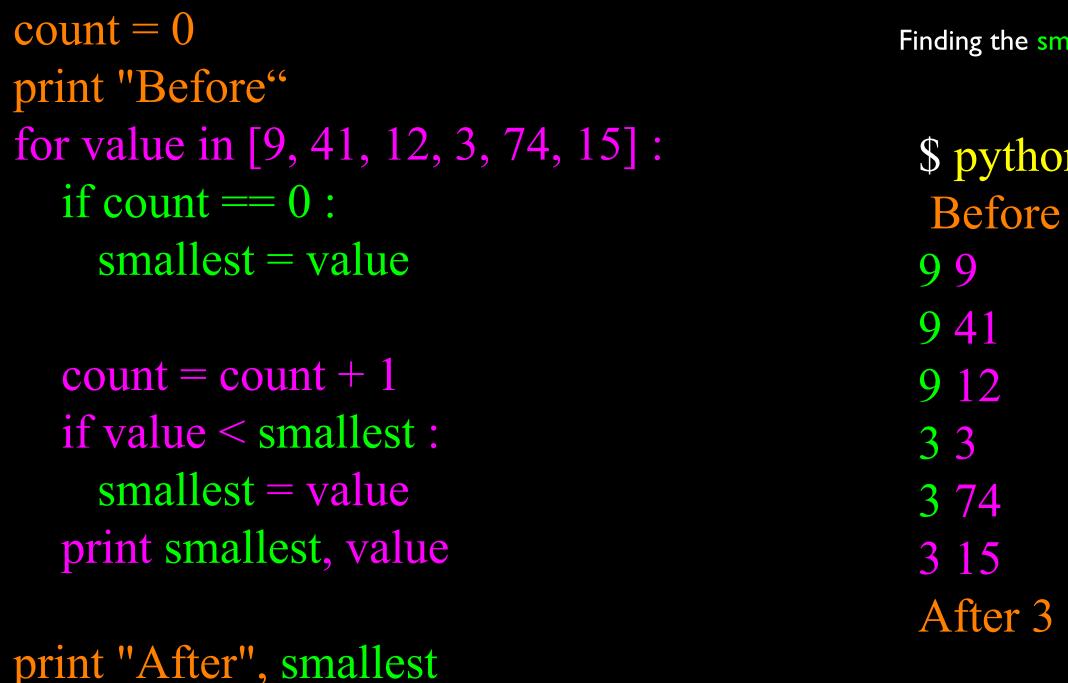
Finding the largest value

largest = -1	\$ pytho
print "Before", largest	Before
for value in [3, 41, 12, 9, 74, 15] :	33
if value > largest :	41 41
largest = value	41 12
print largest, value	41 9
print largest, value	74 74
print "After", largest	74 15
	After 74

We make a variable that contains the largest value we have seen so far. If the current value is larger, it becomes the new largest value we have seen so far.

on largest.py

74



We still have a variable that is the smallest so far. The first time through the loop we take the first value to be the smallest.

Finding the smallest value

\$ python smallest.py

Breaking out of a loop

print "Before" for value in [3, 41, 12, 9, 74, 15] : print "Loop top", value if value == 12: break print "Loop bottom", value print "After"

Before Loop top 3 Loop top 41 Loop top 12

Break immediately terminates the current loop and jumps out of the loop.

\$ python breakloop.py Loop bottom 3 Loop bottom 41 After break (out)

Breaking out of a loop

print "Before"
for value in [3, 41, 12, 9, 74, 15]:
 print "Loop top",value
 if value == 12:
 break
 print "Loop bottom", value
print "After"

\$ python breakloop.py
Before
Loop top 3
Loop bottom 3
Loop top 41
Loop bottom 41
Loop top 12
After

Break immediately terminates the current loop and jumps out of the loop.

found = False where = -1count = 0print "Before", found for value in [3, 41, 12, 9, 74, 15]: count = count + 1if value == 9: found = True where = count break print found, value print "After", found, where

Remembering where the *first* one was...

Continuing with the next iteration

print "Before" for value in [3, 41, 12, 9, 74, 15] : print "Loop top", value if value > 10: continue print "Loop bottom", value print "After"

Before Loop top 3 Loop top 41 Loop top 12 Loop top 9 <u>Loop</u> top 74 Loop top 15 After

Continue immediately terminates the current loop iteration and jumps to the top of the loop and starts the next iteration of the loop.

\$ python breakloop.py

- Loop bottom 3
- Loop bottom 9

Continuing with the next iteration

print "Before" for value in [3, 41, 12, 9, 74, 15] : print "Loop top", value if value > 10: continue print "Loop bottom", value print "After"

\$ python breakloop.py Before Loop top 3 Loop bottom 3 Loop top 41 Loop top 12 Loop top 9 Loop bottom 9 Loop top 74 Loop top 15 After

Continue immediately terminates the current loop iteration and jumps to the top of the loop and starts the next iteration of the loop.

Nested Loops

for out in [1, 2, 3] : print "Top", out for nest in ["X", "Y"] : print out, nest print "Bottom", out

Each time the outer loop runs once, the inner loop runs completely through the loop.

Top 1 $1 \mathbf{X}$ $1 \mathbf{Y}$ Bottom 1 Top 2 2 X2 Y Bottom 2 Top 3 3 X 3 Y Bottom 3

\$ python nested.py

Boolean Operators and Expressions

Boolean Operations

- We can do calculations with boolean variables just like with integer variables
- The boolean operations are: and or not
- Comparison operators < > <= >= == != return boolean (True or O False)

Boolean Operators

Q	$P \; { t and} \; Q$
Т	Т
F	F
Т	F
F	F
	F T

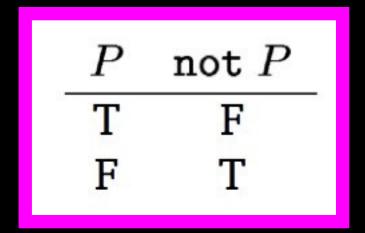
$$(x == 4)$$
 and $(y == 2)$

True if both expressions are true.

$$(x == 4)$$
 or $(y == 2)$

Evaluates to true if either expression is true.

$$\begin{array}{cccc} P & Q & P \text{ or } Q \\ \hline T & T & T \\ T & F & T \\ F & T & T \\ F & F & F \end{array}$$



not (x == 4)

Not "flips" the logic - True becomes False and False becomes True.

Boolean Operation Example



import string

for str in ["bob", "bark at the moon", "where at"]: words = string.split(str) if $len(words) \ge 2$ and words[1] == "at" :print "+++++", str else: print "-----", str

\$ python findat.py ---- bob +++++ bark at the moon++++ where at

Summary

- Loops over a set Detecting in loops File Loops ightarrowLargest or smallest Counting in loops \mathbf{O} Using break in a loop Summing in loops Using continue in a loop \bigcirc Averaging in loops Boolean operations and, or, not
- Searching in loops