Control flow

Michael Ernst
UW CSE 140
Winter 2013
Repeating yourself

Making decisions
Temperature conversion chart

Recall exercise from previous lecture

```python
fahr = 30
cent = (fahr-32)/9.0*5
print fahr, cent
fahr = 40
cent = (fahr-32)/9.0*5
print fahr, cent
fahr = 50
cent = (fahr-32)/9.0*5
print fahr, cent
fahr = 60
cent = (fahr-32)/9.0*5
print fahr, cent
fahr = 70
cent = (fahr-32)/9.0*5
print fahr, cent
print "All done"
```

Output:

```
30 -1.11
40 4.44
50 10.0
60 15.56
70 21.11
All done
```
A better way to repeat yourself:

```python
for f in [30, 40, 50, 60, 70]:
    print f, (f-32)/9.0*5
print "All done"
```

Output:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30</td>
<td>-1.11</td>
<td>40</td>
<td>4.44</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>10.0</td>
<td>60</td>
<td>15.56</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>21.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>All done</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How a loop is executed: Transformation approach

Idea: convert a `for` loop into something we know how to execute

1. Evaluate the sequence expression
2. Write an assignment to the loop variable, for each sequence element
3. Write a copy of the loop after each assignment
4. Execute the resulting statements
How a loop is executed: Direct approach

1. Evaluate the sequence expression
2. While there are sequence elements left:
   a) Assign the loop variable to the next remaining sequence element
   b) Execute the loop body

for i in [1, 4, 9]:
    print i

State of the computer:

Printed output:

1
4
9
The body can be multiple statements

Execute whole body, then execute whole body again, etc.

```python
for i in [3, 4, 5]:
    print "Start body"
    print i
    print i*i
```

Output:
```
Start body 3
Start body 4
Start body 5
```

Loop body: 3 statements

**Convention:** often use i or j as loop variable

This is an exception to the rule that variable names should be descriptive
Indentation is significant

• Every statement in the body must have exactly the same indentation
• That’s how Python knows where the body ends

```python
for i in [3, 4, 5]:
    print "Start body"
    print i
    print i*i
    print "All done"
```

Error!

• Compare the results of these loops:

```python
for f in [30, 40, 50, 60, 70]:
    print f, (f-32)/9.0*5
print "All done"
```

```python
for f in [30, 40, 50, 60, 70]:
    print f, (f-32)/9.0*5
print "All done"
```
The body can be multiple statements

How many statements does this loop contain?

```python
for i in [0, 1]:
    print "Outer", i
    for j in [2, 3]:
        print " Inner", j
        print " Sum", i+j
    print "Outer", i
```

What is the output?

```
Output:
Outer 0
Inner 2
Sum 2
Inner 3
Sum 3
Outer 0
Outer 1
```

“nested” loop body: 2 statements

Loop body: 3 statements
Understand loops through the transformation approach

Key idea:

1. Assign each sequence element to the loop variable
2. Duplicate the body

```python
for i in [0,1]:
    i = 0
    print "Outer", i
    for j in [2,3]:
        print " Inner", j
        i = 1
        print "Outer", i
        for j in [2,3]:
            print " Inner", j
```
Fix this loop

# Goal: print 1, 2, 3, ..., 48, 49, 50
for tens_digit in [0, 1, 2, 3, 4]:
    for ones_digit in [1, 2, 3, 4, 5, 6, 7, 8, 9]:
        print tens_digit * 10 + ones_digit

What does it actually print?
How can we change it to correct its output?

Moral: Watch out for edge conditions (beginning or end of loop)
Test your understanding of loops

Puzzle 1:
   for i in [0,1]:
       print i
   print i

Puzzle 2:
   i = 5
   for i in []:  # Reusing loop variable (don’t do this!)
       print i

Puzzle 3:
   for i in [0,1]:
       print "Outer", i
       for i in [2,3]:
           print "Inner", i
       print "Outer", i

Output:

0
1
1

(no output)
The range function

A typical for loop does not use an explicit list:

```python
for i in range(5):
    ... body ...
```

- `range(5) = [0, 1, 2, 3, 4]`
- `range(1, 5) = [1, 2, 3, 4]`
- `range(1, 10, 2) = [1, 3, 5, 7, 9]`

- Upper limit (exclusive)
- Lower limit (inclusive)
- Step (distance between elements)
Decomposing a list computation

• To compute a value for a list:
  – Compute a partial result for all but the last element
  – Combine the partial result with the last element

Example: sum of a list:

\[ [3, 1, 4, 1, 5, 9, 2, 6, 5] \]

\[
\begin{align*}
\text{sum(List a)} &= \text{sum(List a)} + 5 \\
\text{sum(List b)} &= \text{sum(List c)} + 6 \\
…
\text{sum(List y)} &= \text{sum(List z)} + 3 \\
\text{sum(empty list)} &= 0
\end{align*}
\]
How to process a list: One element at a time

• A common pattern when processing a list:
  ```python
  result = initial_value
  for element in list:
      result = updated result
  ... use result
  ```

• `initial_value` is a correct result for an empty list

• As each element is processed, `result` is a correct result for a prefix of the list

• When all elements have been processed, `result` is a correct result for the whole list

# Sum of a list
```python
result = 0
for element in mylist:
    result = result + element
```
Examples of list processing

• Product of a list:
  ```python
  result = 1
  for element in mylist:
    result = result * element
  ```

• Maximum of a list:
  ```python
  result = mylist[0]
  for element in mylist:
    result = max(result, element)
  ```

• Approximate the value 3 by $1 + 2/3 + 4/9 + 8/27 + 16/81 + \ldots = (2/3)^0 + (2/3)^1 + (2/3)^2 + (2/3)^3 + \ldots + (2/3)^{10}$
  ```python
  result = 0
  for element in range(11):
    result = result + (2.0/3.0)**element
  ```

The first element of the list (counting from zero)
Making decisions

• How do we compute absolute value?
  abs(5) = 5
  abs(0) = 0
  abs(-22) = 22
Absolute value solution

If the value is negative, negate it.
Otherwise, use the original value.

```python
val = -10
if val < 0:
    result = - val
else:
    result = val
print result
```

```python
val = -10
if val < 0:
    print - val
else:
    print val
```
The if body can be any statements

```python
# height is in km
if height > 100:
    print "space"
else:
    if height > 50:
        print "mesosphere"
    else:
        if height > 20:
            print "stratosphere"
        else:
            print "troposphere"
```

Execution gets here only if "height > 100" is false

```python
# height is in km
if height > 50:
    print "mesosphere"
else:
    if height > 20:
        print "stratosphere"
    else:
        print "troposphere"
```

Execution gets here only if "height > 100" is false AND "height > 50" is true
The then clause or the else clause is executed

```python
if is_prime(x):
    y = x / 0
else:
    y = x**x
```