Return values, multiple assignment

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 9, Spring 2021

Monday March 1
More on boolean operators
## Compound boolean operations

Allow us to check if multiple conditions are true. Assume $x = 1, y = 2, z = False$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>not $x$</td>
<td>True if $x$ is false, False if $x$ is true</td>
<td>if not z:</td>
</tr>
<tr>
<td></td>
<td></td>
<td># Do stuff</td>
</tr>
<tr>
<td>$x$ and $y$</td>
<td>True only if <em>both</em> $x$ and $y$ is true, otherwise False</td>
<td>if $x &gt; 0$ and $y &lt; 3$:</td>
</tr>
<tr>
<td></td>
<td></td>
<td># Do stuff</td>
</tr>
<tr>
<td>$x$ or $y$</td>
<td>True if <em>at least one of</em> $x$ or $y$ is true</td>
<td>if $x &gt; 5$ or $y &lt; 3$:</td>
</tr>
<tr>
<td></td>
<td>False if they are <em>both</em> false</td>
<td># Do stuff</td>
</tr>
</tbody>
</table>
Precedence Rules: Higher in the table get evaluated earlier

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>math operators</td>
<td>* / + - etc</td>
</tr>
<tr>
<td>relational operators</td>
<td>&lt; &gt; == != etc</td>
</tr>
<tr>
<td>not x</td>
<td>True if x is false, False if x is true</td>
</tr>
<tr>
<td>x and y</td>
<td>True only if both x and y is true, otherwise False</td>
</tr>
<tr>
<td>x or y</td>
<td>True if at least one of x or y is true</td>
</tr>
<tr>
<td></td>
<td>False if they are both false</td>
</tr>
</tbody>
</table>
Lazy evaluation (1/2)

If there is more than one of the same compound boolean operator in a conditional statement, Python evaluates left-to-right.

In a long string of and conditions, Python stops checking and considers the entire expression False if it encounters one False:

```python
if a and b and c and d and e and f and g:
    print('yes')
```

If a is False, Python does not even bother checking b, c, etc.
Lazy evaluation (2/2)

In a long string of or conditions, Python stops checking and considers the entire expression True if it encounters one True:

```python
if a or b or c or d or e or f or g:
    print('yes')
```

If a is True, Python does not even bother checking b, c, etc.
Returning values from a function
Returning values from a function

```python
import math
def distance(x, y):
    dist = math.sqrt(x**2 + y**2)

dist = 0
x = 3
y = 4
distance(x, y)
print(dist)
```

- What is printed at the end of the program?
- `dist` is not changed in the main symbol table and its value computed in the function "disappears"! How can we get the value to "live on"???
import math

def distance(x, y):
    dist = math.sqrt(x**2 + y**2)
    return dist

dist = 0
x = 3
y = 4
dist = distance(x, y)
print(dist)

• Let's make the changes and run the code.
A **very** common error: attempting to assign a value from a function that DOES NOT return a value:

```python
def distance(x, y):
    dist = math.sqrt(x**2 + y**2)

# main code
z = distance(x, y)
```

There is no return statement, so `distance` has nothing to assign to the variable `z` in the main symbol table.

- Python tries its best to avoid crashing and assigns `z` a special value called `None`.
- Python keeps executing the rest of the program.
None (2/3)

• You can assign variables of your own to None: `myVar = None`.

• You can checking if a variable is set to None:
  • `if z is None: ...`
  • `or see it printed print(z)`
Same problem happens if you tried to assign a value to a function call that has an "empty" return statement, but doesn’t return anything:

```python
def distance(x, y):
    return

# main code
z = distance(x, y)
```
Example: Returning values to other functions

Let's write a program that prints the happy birthday song.
Symbol table and return values

Example: Calculating car's miles-per-gallon (MPG).