### Structure vs. Flow

**Program Structure**
- Way statements are presented
  - Order statements are listed
  - Inside/outside of a function
  - Will see other ways...
- Indicate possibilities over multiple executions

**Program Flow**
- Order statements are executed
  - Not the same as structure
  - Some statements duplicated
  - Some statements are skipped
- Indicates what really happens in a single execution

Have already seen this difference with functions

---

### Conditionals: If-Statements

**Format**

```
if <boolean-expression>:
    <statement>
    ...
else:
    <statement>
```

**Example**

```
# Put x in z if it is positive
if x > 0:
    z = x
```

**Execution**

If `<boolean-expression>` is true, then execute all of the statements indented directly underneath (until first non-indented statement).

---

### Conditionals: If-Else-Statements

**Format**

```
if <boolean-expression>:
    <statement>
    ...
else:
    <statement>
```

**Example**

```
# Put max of x, y in z
if x > y:
    z = x
else:
    z = y
```

**Execution**

If `<boolean-expression>` is true, then execute statements indented under if; otherwise execute the statements indented under else.

---

### Conditionals: “Control Flow” Statements

```
if b:
    | s1 # statement
    s3
```

```
if b:
    | s1
else:
    | s2
s3
```

**Flow**

Program only takes one path each execution

---

### Program Flow vs. Local Variables

```
max(x,y):
    """Returns: max of x, y""
    # swap x, y
    # put the larger in y
    if x > y:
        temp = x
        x = y
        y = temp
    return y
```

- `temp` is needed for swap
- `x = y` loses value of `x`
- "Scratch computation"
- Primary role of local vars

```
max(3,0):
```

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>y</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Program Flow vs. Local Variables

def max(x,y):
    """Returns: max of x, y""
    # swap x, y
    # put the larger in y
    if x > y:
        temp = x
        x = y
        y = temp
    return temp

- Value of max(3,0)?
  A: 3  
  B: 0  
  C: Error!  
  D: I do not know

- Local variables last until
  - They are deleted or
  - End of the function
- Even if defined inside if

Program Flow and Testing

- Must understand which
  flow caused the error
  - Unit test produces error
  - Visualization tools show
    the current flow for error
- Visualization tools?
  - print statements
  - Advanced tools in IDEs
    (Integrated Dev. Environ.)

# Put max of x, y in z
print 'before if'
if x > y:
    print 'if x>y'  
        z = x
else:
    print 'else x<y'  
        z = y
print 'after if'

Watches vs. Traces

<table>
<thead>
<tr>
<th>Watch</th>
<th>Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization tool</td>
<td>Visualization tool</td>
</tr>
<tr>
<td>(e.g. print statement)</td>
<td>(e.g. print statement)</td>
</tr>
<tr>
<td>Looks at variable value</td>
<td>Looks at program flow</td>
</tr>
<tr>
<td>Often after an assignment</td>
<td>Before/after any point</td>
</tr>
<tr>
<td>What you did in lab</td>
<td>where flow can change</td>
</tr>
</tbody>
</table>

Traces and Functions

Example: flow.py

def shift(p):
    print 'Start shift()'  
    p.x = p.y
    print p.x
    p.y = p.z
    print p.y
    p.z = p.x
    print p.z
    print 'End shift()'

Watches

Traces

Local Variables Revisited

- Never refer to a variable that might not exist
- Variable "scope"
  - Block (indented group) where it was first assigned
  - Way to think of variables; not actually part of Python
- Rule of Thumb: Limit variable usage to its scope

def max(x,y):
    """Returns: max of x, y""
    # swap x, y
    # put larger in temp
    temp = y
    if x > y:
        if temp > x:
            temp = x
    return temp

Conditionals: If-Elif-Else-Statements

Format                              | Example
---                                 | ---
If <boolean-expression>:            | # Put max of x, y, z in w
    <statement>                    | if x > z and y > z:
elif <boolean-expression>:         |     w = x
    <statement>                    |     if y > z:
else:                               |     w = y
    ...                            |     else:
    else:                           |     w = z
    ...                            | ...