Lecture 8: Conditionals & Control Flow (Sections 5.1-5.7)

CS 1110

Introduction to Computing Using Python

http://www.cs.cornell.edu/courses/cs1110/2018sp

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]
Feb 22: CS 1110: Announcements

- Sign up for a one-on-one!
  - CMS: **OPTIONAL**: one-on-ones

- Prelim 1 is March 13. You have until March 1st, 11:59pm to register a conflict or a need for accommodation. There is no single makeup session. See website: “Assessment ➔ Exams”
  - CMS: Prelim 1 conflicts

- A2 tentatively: released Wed 2/28, due Wed 3/7
Big Picture

Statements either affect data or control

- **DATA:** change the value in a box, create a box, etc.
  
  Examples:
  
  \[ x = x + 1 \]
  
  \[ name = \text{"Alex"} \]

- **CONTROL:** tell python what line to execute next
  
  Examples:
  
  \[ \text{greet(name)} \]
  
  \[ \text{if name == \text{"Alex"}}: \quad \leftarrow \text{today's Lecture} \]
## Conditionals: If-Statements

### Format

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

### Example

```
# is there a new high score?
if curr_score > high_score:
    high_score = curr_score
    print(“New high score!”)
```

### Execution:

If `<Boolean-expression>` is true, then execute all of the statements indented directly underneath (until first non-indented statement)
What are Boolean expressions?

Expressions that evaluate to a Boolean value.

```python
is_student = True
is_senior = False
num_credits = 25
```

**Boolean operations:**
```python
if is_student and is_senior:
    print(“Hi senior student!”)
```

**Boolean variables:**
```python
if is_student:
    print(“Hi student!”)
```

**Comparison operations:**
```python
if num_credits > 24:
    print(“Are you serious?”)
```
What gets printed, Round 1

\[
\begin{align*}
&\quad a = 0 \quad a = 0 \quad a = 0 \quad a = 0 \quad a = 0 \\
&\text{print}(a) \quad a = a + 1 \quad \text{if } a == 0: \quad \text{if } a == 1: \quad \text{if } a == 0: \\
&\quad \quad \text{print}(a) \quad a = a + 1 \quad a = a + 1 \quad a = a + 1 \\
&\quad \quad \text{print}(a) \quad \quad \text{print}(a) \quad a = a + 1 \\
&\quad \quad \quad a = a + 1 \\
&\quad \quad \quad \quad \text{print}(a)
\end{align*}
\]
What gets printed? (Question)

```
a = 0
if a == 0:
    a = a + 1
if a == 0:
    a = a + 2
a = a + 1
print(a)
```

A: 0
B: 1
C: 2
D: 3
E: I do not know
What gets printed? (Solution)

```
a = 0
if a == 0:
    a = a + 1
if a == 0:
    a = a + 2
a = a + 1
print(a)
```

| a = 0       | Executed |
| if a == 0:  | Executed |
| a = a + 1   | Executed |
| if a == 0:  | Executed |
| a = a + 2   | Skipped  |
| a = a + 1   | Executed |

A: 0
B: 1
C: 2  CORRECT
D: 3
E: I do not know
Conditionals: If-Else-Statements

**Format**

<table>
<thead>
<tr>
<th>if &lt;boolean-expression&gt;:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;statement&gt;</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>else:</td>
</tr>
<tr>
<td>&lt;statement&gt;</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

**Example**

# who is the winner?

```python
if score1 > score2:
    winner = "Player 1"
else:
    winner = "Player 2"
```

**Execution:**

if <Boolean-expression> is true, then execute statements indented under if; otherwise execute the statements indented under else
Conditionals: “Control Flow” Statements

\[
\text{if } b:\ \begin{align*}
\text{True} & \quad s1 \\
\text{False} & \quad s3
\end{align*}
\]

\[
\text{if } b:\ \begin{align*}
\text{True} & \quad s1 \\
\text{False} & \quad s2
\end{align*}
\]

Flow
Program only takes one path each execution
Structure vs. Flow

Program Structure
- Order in which statements are written in scripts and modules
- Not necessarily the order in which Python executes them

if $b$:
  $s_1$
else:
  $s_2$
$s_3$

Control Flow
- Order in which statements are actually executed at runtime
  - Statements may be:
    - skipped
    - executed more than once

Diagram:
- True branch
  - $b$ (True)
  - $s_1$
  - $s_3$

- False branch
  - $b$ (False)
  - $s_2$
  - $s_3$
What gets printed, Round 2

\[
\begin{align*}
\text{a} &= 0 \\
\text{if } a == 0: & \quad \text{if } a == 1: \\
& \quad a = a + 1 \\
& \quad a = a + 1 \\
\text{else:} & \quad \text{else:} \\
& \quad a = a + 2 \\
& \quad a = a + 2 \\
\text{print(a)} & \quad \text{print(a)} \\
& \quad a = a + 1 \\
& \quad a = a + 1 \\
1 & \quad 2 \\
\end{align*}
\]
if determines which statement is executed next

def get_in_car(car_locked):
    if car_locked:
        print("Unlock the car!")
    print("Open the door.")

is_my_car_locked = True
get_in_car(is_my_car_locked)
if determines which statement is executed next

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock the car!")
        print("Open the door.")
    is_my_car_locked = True
    get_in_car(is_my_car_locked)
```

Program Flow (car locked, step 2)
**Program Flow (car not locked, step 1)**

`if` determines which statement is executed next

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock the car!")
        print("Open the door.")
    is_my_car_locked = False
    get_in_car(is_my_car_locked)
```
Program Flow (car not locked, step 2)

**if** determines which statement is executed next

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock the car!")
    print("Open the door.")
```

is_my_car_locked = False
get_in_car(is_my_car_locked)
```
```python
def max(x, y):
    if x > y:
        return x
    return y
```

Current call frame:
```
max
+---+
|   | 1|
+---+
    +---+
    |   | 0|
    +---+
    +---+
        +---+
        |   | 3|
        +---+
```

What does the call frame look like next? (Q)

A:
```
max
+---+
|   | 2|
+---+
    +---+
    |   | 0|
    +---+
    +---+
        +---+
        |   | 3|
        +---+
```

B:
```
max
+---+
|   | 0|
+---+
    +---+
    |   | 0|
    +---+
    +---+
        +---+
        |   | 3|
        +---+
        +---+
            +---+
            |   | 3|
            +---+
            +---+
                +---+
                |   | RETURN|
                +---+
```

C:
```
max
+---+
|   | 3|
+---+
    +---+
    |   | 0|
    +---+
    +---+
        +---+
        |   | 3|
        +---+
```

D:
```
max
+---+
|   | 3|
+---+
    +---+
    |   | 0|
    +---+
    +---+
        +---+
        |   | 3|
        +---+
```

What does the call frame look like next? (A)

def max(x,y):
    if x > y:
        return x
    return y

max(0,3)

Current call frame:

A:

B:

C:

D:
Call Frame Explanation (1)

```python
def max(x, y):
    if x > y:
        return x
    return y
```

max(0, 3):

```
<table>
<thead>
<tr>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>y</td>
</tr>
</tbody>
</table>
```

19
def max(x, y):
    if x > y:
        return x
    return y

max(0, 3):

    max
    x 0
    y 3

Skips line 2
def max(x,y):
    if x > y:
        return x
    return y

max(0,3):
Variables created inside **if** continue to exist past **if**:  
```python
a = 0
if a == 0:
    b = a + 1
print(b)
```

...but are only created if the program actually executes that line of code
What gets printed, Round 3

```python
a = 0
if a == 0:
    b = 0
print(b)
```

```python
a = 0
if a == 1:
    b = 0
print(b)
```

0

Error!
def zero_or_one(a):
    if a == 1:
        b = 1
    else:
        b = 0
    print(b)

make sure that ALL if branches create the variable
def max(x, y):
    """Returns: max of x, y""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(3, 0)
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(3,0)

Value of maximum?

A: 3  CORRECT
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
**Control Flow and Variables (Q2)**

def max(x,y):
    
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(0,3)

Value of maximum?

A: 3
B: 0
C: Error!
D: I do not know
Control Flow and Variables (A2)

def max(x,y):
    """Returns: max of x, y""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(0,3)

Value of maximum?

A: 3
B: 0
C: Error!  CORRECT
D: I do not know

• Variable existence depends on flow
• Generally terrible idea to refer to variables defined inside an if clause
Can use print statements to examine program flow

```python
# Put max of x, y in z
print('before if')
if x > y:
    print('inside if x>y')
    z = x
else:
    print('inside else (x<=y')
    z = y
print('after if')
```

'before if'
'inside if x>y'
'after if'
x must have been greater than y
# Conditionals: If-Elif-Else-Statements

## Format

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>if &lt;Boolean expression&gt;</code>:</td>
<td>The <code>if</code> statement checks the boolean expression. If the expression is true, the associated block of code is executed.</td>
</tr>
<tr>
<td><code>elif &lt;Boolean expression&gt;</code>:</td>
<td>The <code>elif</code> statement is used when multiple conditions need to be checked, and each condition is independent of the others. The <code>elif</code> statement is used when the previous condition was false.</td>
</tr>
<tr>
<td><code>else:</code></td>
<td>If none of the <code>if</code> or <code>elif</code> conditions are met, the <code>else</code> block is executed.</td>
</tr>
</tbody>
</table>

## Example

```python
# Find the winner
if score1 > score2:
    winner = "Player 1"
elif score2 > score1:
    winner = "Player 2"
else:
    winner = "Both Players"
```
# Conditionals: If-Elif-Else-Statements

## Format

```python
if <Boolean expression>:
    <statement>
    ...
elif <Boolean expression>:
    <statement>
    ...
else:
    <statement>
    ...
```

## Notes on Use

- No limit on number of `elif`
  - Must be between `if, else`
- `else` is optional
  - `if-elif` by itself is fine
- Booleans checked in order
  - Once Python finds a true `<Boolean-expression>`, skips over all the others
  - `else` means **all** are false
If-Elif-Else (Question)

```
a = 2

if a == 2:
a = 3
elif a == 3:
a = 4
print(a)
```

What gets printed?

A: 2
B: 3
C: 4
D: I do not know
a = 2

if a == 2:
a = 3
elif a == 3:
a = 4
print(a)

What gets printed?

A: 2
B: 3  CORRECT
C: 4
D: I do not know
What gets printed, Round 4

```
a = 2

if a == 2:
    a = 3

elif a == 3:
    a = 4

print(a)
```

```
a = 2

if a == 2:
    a = 3

if a == 3:
    a = 4

print(a)
```

3

4
def what_to_wear(raining, freezing):
    if raining:
        if freezing:
            print("Wear a waterproof coat.")
        else:
            print("Bring an umbrella.")
    else:
        if freezing:
            print("Wear a warm coat!")
        else:
            print("A jacket will suffice.")


dude_wheres_my_prelim.py