Lecture 8: Conditionals & Control Flow (Sections 5.1-5.7)  
CS 1110  
Introduction to Computing Using Python

Announcements
- Optional 1-on-1 with a staff member to help just you with course material. Sign up for a slot on CMS under “SPECIAL: one-on-ones.”
- A1 part A first submission due Mar 5 Fri at 11:59pm
- A1 part B first submission due Mar 8 Mon at 11:59pm
- Conditionals—today’s topic—not allowed in A1

Conditionals: If-Statements

Format

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

Example

```python
# 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

<table>
<thead>
<tr>
<th>a = 0</th>
<th>a = 0</th>
<th>a = 0</th>
<th>a = 0</th>
<th>a = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(a)</td>
<td>a = a + 1</td>
<td>if a == 0:</td>
<td>if a == 1:</td>
<td>if a == 0:</td>
</tr>
<tr>
<td></td>
<td>print(a)</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
</tr>
<tr>
<td></td>
<td>print(a)</td>
<td>print(a)</td>
<td>a = a + 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>print(a)</td>
<td></td>
</tr>
</tbody>
</table>

What gets printed? (Question)

```python
a = 0
if a == 0:
    a = a + 1
if a == 0:
    a = a + 2
a = a + 1
```

<table>
<thead>
<tr>
<th>A: 0</th>
<th>B: 1</th>
<th>C: 2</th>
<th>D: 3</th>
<th>E: I do not know</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>print(a)</td>
</tr>
</tbody>
</table>
## Conditionals: If-Else-Statements

**Format**

\[
\text{if } <\text{boolean-expression}>: \\
\quad <\text{statement}> \\
\text{else:} \\
\quad <\text{statement}>
\]

**Example**

\[
\text{if } \text{curr_score} > \text{high_score}: \\
\quad \text{print(‘New record!’)} \\
\text{else:} \\
\quad \text{print(‘Try again next time’)}
\]

**Execution:**

if (boolean-expression) is true, then execute statements indented
under if; otherwise execute the statements indented under else

## Conditionals: “Control Flow” Statements

**Format**

\[
\text{if } b: \\
\quad s1 \quad \# \text{ statement} \\
\text{else:} \\
\quad s2 \quad \# \text{ statement}
\]

**Example**

\[
\text{if } \text{new record?}: \\
\quad \text{print(‘New record!’)} \\
\text{else:} \\
\quad \text{print(‘Try again next time’)}
\]

**Execution:**

if ⟨boolean-expression⟩ is true, then execute statements indented
under if; otherwise execute the statements indented under else

### What gets printed, Round 2

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 0</td>
<td>a = 0</td>
<td>a = 0</td>
<td>a = 0</td>
<td></td>
</tr>
<tr>
<td>if a == 0:</td>
<td>if a == 1:</td>
<td>if a == 1:</td>
<td>if a == 1:</td>
<td></td>
</tr>
<tr>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td></td>
</tr>
<tr>
<td>else:</td>
<td>else:</td>
<td>else:</td>
<td>else:</td>
<td></td>
</tr>
<tr>
<td>a = a + 2</td>
<td>a = a + 2</td>
<td>a = a + 2</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
</tr>
<tr>
<td>print(a)</td>
<td>print(a)</td>
<td>print(a)</td>
<td>a = a + 1</td>
<td>print(a)</td>
</tr>
</tbody>
</table>

### Program Flow (car locked, 1)

if determines which statement is executed next

\[
\text{def get_in_car(car_locked):} \\
1 \quad \text{if car_locked:} \\
2 \quad \text{print(‘Unlock car!’)} \\
3 \quad \text{print(‘Open the door.’)}
\]

\[
\text{car_locked = True} \\
\text{get_in_car(car_locked)}
\]

### Program Flow (car locked, 2)

if determines which statement is executed next

\[
\text{def get_in_car(car_locked):} \\
1 \quad \text{if car_locked:} \\
2 \quad \text{print(‘Unlock car!’)} \\
3 \quad \text{print(‘Open the door.’)}
\]

\[
\text{car_locked = True} \\
\text{get_in_car(car_locked)}
\]

### Program Flow (car locked, 3)

if determines which statement is executed next

\[
\text{def get_in_car(car_locked):} \\
1 \quad \text{if car_locked:} \\
2 \quad \text{print(‘Unlock car!’)} \\
3 \quad \text{print(‘Open the door.’)}
\]

\[
\text{car_locked = True} \\
\text{get_in_car(car_locked)}
\]
```
```
```python
if determines which statement is executed next

def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
    car_locked = True
    get_in_car(car_locked)
```
**Program Flow and Variables**

Variables created inside if continue to exist past if:
- `a = 0`
- `if a == 0:
  - `b = a + 1`
  - `print(b)`

...but are only created if the program actually executes that line of code

---

**Control Flow and Variables (Q1)**

```
def max(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
- B: 0
- C: Error!
- D: I do not know

---

**Control Flow and Variables (Q2)**

```
def max(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

---

**Conditionals: If-Elif-Else-Statements**

**Format**

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

**Example**

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

**Format**

```
if <Boolean expression>:
    <statement>
... else:
    <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 `<Boolean-expression>` are false

---

What gets printed, Round 3

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

Where is the robot?

- Angle of the robot relative to the sensor is `d` degrees, where `d` is non-negative
- Robot is in which quadrant?
- To avoid ambiguity, use this convention:
  - 1 if `0 ≤ d < 90`
  - 2 if `90 ≤ d < 180`
  - 3 if `180 ≤ d < 270`
  - 4 if `270 ≤ d < 360`

WARNING

Robot Operating in Quadrant 1

Can solve using **if-elif-elif**... Other options?

---

If-Elif-Else (Question)

```python
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

---

Nesting Conditionals

- Separate choices into 2 general categories
- Subdivide each category into subcategories
- Subdivide each subcategory further...

```python
if <above x-axis>:
    if <left of y-axis>:
        ... else:
    else:
        if <left of y-axis>:
            ... else:
```

Program Flow and Testing

Can use print statements to examine program flow

```python
if x > y:
    z = x
else:
    z = y
```

# Put max of x, y in z

---

See quadrants.py
Can use print statements to examine program flow

`before if`
`inside if x>y`
`after if`

x must have been greater than y

Program Flow and Testing

# 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')

Traces (control) and Watches (data)

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

TRACES
Trace program flow
What code is being executed?
Place them at the beginning of a block of code that might be skipped.

WATCHES
Watch data values
What is the value of a variable?
Place them after assignment statements.