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

CS 1110

Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]
• Lots of written questions about Print vs Return
  ▪ Please see: https://edstem.org/us/courses/19140/discussion/1084754

• A common post-lecture If-Elif-Else question:
  ▪ https://edstem.org/us/courses/19140/discussion/1160274
Announcements

- **A1: a1_first.py & policy_acknowledgement submission**
  - Submit whatever you have at 2pm
  - Keep submitting as you make significant changes
  - Final submission due tonight at 11:59pm

- **Conditionals—today’s topic—not allowed in A1**
What should I wear today?

```python
def what_to_wear(temp):
    print("Today you should wear:")
    # > 60: no jacket required
    # 40-60: jacket
    # 20-40: winter coat
    # < 20: all the gear you own
```

How to we implement this in Python?
## 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.

is_rainy = False
is_windy = True
temp = 12

Boolean variables:
if is_rainy:
    print("Bring an umbrella!")

Comparison operations:
if temp < 30 and is_rainy:
    print("Roads will be icy!")

if temp > 70:
    print("Hallelujah!")

Boolean operations:
if is_windy and not is_rainy:
    print("Let's fly a kite!")
What gets printed, Round 1

\[ a = 0 \]
\[ \text{print}(a) \]
\[ a = a + 1 \]
\[ \text{print}(a) \]
\[ a = 0 \]
\[ \text{if } a == 0: \]
\[ a = a + 1 \]
\[ \text{print}(a) \]
\[ a = 0 \]
\[ \text{if } a == 1: \]
\[ a = a + 1 \]
\[ \text{print}(a) \]
\[ a = 0 \]
\[ \text{if } a == 0: \]
\[ a = a + 1 \]
\[ \text{print}(a) \]

(Let's look at these one by one.)
What gets printed? (Question)

```python
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
## Conditionals: If-Else-Statements

### Format

<table>
<thead>
<tr>
<th>if &lt;boolean-expression&gt;:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;statement&gt;</td>
<td># new record?</td>
</tr>
<tr>
<td>...</td>
<td>if curr_score &gt; high_score:</td>
</tr>
<tr>
<td>else:</td>
<td>print(“New record!”)</td>
</tr>
<tr>
<td>&lt;statement&gt;</td>
<td>else:</td>
</tr>
<tr>
<td>...</td>
<td>print(“Nice try.”)</td>
</tr>
</tbody>
</table>

### 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  # statement
else:
    s1
    s2
    s3
```

Flow
Program only takes one path during an execution (something will not be executed!)
What gets printed, Round 2

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

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

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

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

(Let's look at these one by one.)
**Program Flow (car locked, 0)**

*if* determines which statement is executed next

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

Global Space
Program Flow (car locked, 1)

if determines which statement is executed next

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

Global Space

```
car_locked = True
```
Program Flow (car locked, 2)

**if** determines which statement is executed next

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

    car_locked = True
get_in_car(car_locked)
```

**Global Space**
- car_locked: True

**Call Stack**
- get_in_car: 1
- is_locked: True
Program Flow (car locked, 3)

*if* determines which statement is executed next

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

car_locked = True
get_in_car(car_locked)
```

Global Space

```
car_locked: True
```

Call Stack

```
get_in_car: 1 2
is_locked: True
```
**Program Flow (car locked, 4)**

**if** determines which statement is executed next

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

**Global Space**
- `car_locked`: True

**Call Stack**
- `get_in_car`
- `is_locked`: True
if determines which statement is executed next

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

    car_locked = True
    get_in_car(car_locked)
Program Flow (car locked, 6)

**if** determines which statement is executed next

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

car_locked = True
get_in_car(car_locked)

Unlock car!
Open the door.
Program Flow (car not locked, 0)

`if` determines which statement is executed next

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

Global Space
**Program Flow (car not locked, 1)**

**if** determines which statement is executed next

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

Global Space

<table>
<thead>
<tr>
<th>car_locked</th>
<th>False</th>
</tr>
</thead>
</table>

24
Program Flow (car not locked, 2)

\textbf{if} determines which statement is executed next

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

    car_locked = False
    get_in_car(car_locked)
Program Flow (car not locked, 3)

`if` determines which statement is executed next

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

car_locked = False
get_in_car(car_locked)
```
Program Flow (car not locked, 4)

**if** determines which statement is executed next

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

car_locked = False
get_in_car(car_locked)
```

Global Space

```
car_locked = False
```

Call Stack

```
get_in_car
```

```
is_locked
```

```
RETURN
```

27
if determines which statement is executed next

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

car_locked = False
going_in_car(car_locked)
What does the call frame look like next? (Q)

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

max(0, 3)

Current call frame:

<table>
<thead>
<tr>
<th>max</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
</tbody>
</table>
What does the call frame look like next? (Q)

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

Current call frame:
```
def max(x, y):
    if x > y:
        return x
    return y
max(0, 3)
```

A: |
---|
| max | 1 2 |
| x   | 0   |
| y   | 3   |

B: |
---|
| max | 1 2 3 |
| x   | 0   |
| y   | 3   |
| RETURN | 0 |

C: |
---|
| max | 1 2 3 |
| x   | 0   |
| y   | 3   |
| RETURN | 3 |

D: |
---|
| max | 1 3 |
| x   | 0   |
| y   | 3   |
Program Flow and Variables

Variables created inside \texttt{if} continue to exist past \texttt{if}:

\begin{verbatim}
a = 0
if a == 0:
    b = a + 1
print(b)
\end{verbatim}

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

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

```
a = 1
if a == 0:
    b = 0
print(b)
```
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
B: 0
C: Error!
D: I do not know
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
def zero_or_one(a):
    if a == 1:
        b = 1
    else:
        b = 0
    print(b)

make sure that ALL if branches create the variable
Conditionals: If-Elif-Else-Statements (1)

**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 (2)

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 <Boolean-expression> are false`
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
What gets printed, Round 4

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

```python
a = 2
if a == 2:
a = 3
if a == 3:
a = 4
print(a)
```
def what_to_wear(raining, freezing):
    if raining and freezing:
        print("Wear a waterproof coat.")
    elif raining and not freezing:
        print("Bring an umbrella.")
    elif not raining and freezing:
        print("Wear a warm coat!")
    else:
        print("A sweater will suffice.")
Nested Conditionals to the rescue!

```python
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 sweater will suffice.")
```
Program Flow and Testing

# determine winner

if x_score > y_score:
    winner = "x"
else:
    winner = "y"

Can use print statements to examine program flow
# determine winner
print('before the if')
if x_score > y_score:
    print('inside the if')
    winner = "x"
else:
    print('inside the else')
    winner = "y"
print('after the if')

\[ x \text{ score must have been greater than } y \text{ score } \]

Can use `print` statements to examine program flow

"traces" or "breadcrumbs"

'before the if'
'inside the if'
'after the if'
# determine winner
print('before the if')

if x_score > y_score:
    print('inside the if')
    winner = "x"
    print('winner = ' + winner)
else:
    print('inside the else')
    winner = "y"
    print('winner = ' + winner)

print('after the if')