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

CS 1110

Introduction to Computing Using Python

http://www.cs.cornell.edu/courses/cs1110/2019sp
Big Picture

Statements either affect **data** or **control**

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

- **CONTROL**: tell python what line to execute next
  
  Examples:
  
  ```
  greet(name)
  ```
  
  ```
  if name == "Alex": ← 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

```
a = 0
print(a)
```

```
a = a + 1
print(a)
```

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

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

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

```
a = 0
a = a + 1
print(a)
```

```
a = 0
a = a + 1
print(a)
```

```
a = 0
a = a + 1
print(a)
```

```
0
1
1
0
2
```
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
What gets printed? (Solution)

\[
a = 0
\]
\[
\text{if } a == 0:
\]
\[
| \ a = a + 1 \quad \text{Executed}
\]
\[
\text{if } a == 0:
\]
\[
| \ a = a + 2 \quad \text{Skipped}
\]
\[
a = a + 1 \quad \text{Executed}
\]
\[
\text{print(a)}
\]

\[
A: 0
\]
\[
B: 1
\]
\[
C: 2 \quad \text{CORRECT}
\]
\[
D: 3
\]
\[
E: I \text{ do not know}
\]
Conditionals: If-Else-Statements

**Format**

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

**Example**

```python
# who is the winner?
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

if b:
    # statement
else:
    # statement

Flow
Program only takes one path each execution (something will not be executed!)
<p>| | | | |</p>
<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>if a == 0:</td>
<td>if a == 1:</td>
<td>if a == 1:</td>
<td>if a == 1:</td>
</tr>
<tr>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
</tr>
<tr>
<td>else:</td>
<td>else:</td>
<td>else:</td>
<td>else:</td>
</tr>
<tr>
<td>a = a + 2</td>
<td>a = a + 2</td>
<td>a = a + 2</td>
<td>a = a + 1</td>
</tr>
<tr>
<td>print(a)</td>
<td>print(a)</td>
<td>print(a)</td>
<td>print(a)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
not in_love (0)

if determines which statement is executed next

def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
        print("Happy Valentine’s Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")

name = ("Recipient Name: ")
write_valentine(name=="Kilian")

Recipient Name: Bob
**not in_love (2)**

**if** determines which statement is executed next

```python
def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Recipient Name: Bob
if determines which statement is executed next

```python
def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")
```

```python
name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Recipient Name: Bob
not in_love (4)

if determines which statement is executed next

def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")

Recipient Name: Bob
Let’s be friends!
if not in_love:
    print("Let’s be friends!")
    print("Happy Valentine’s Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")

Recipient Name: Bob
Let’s be friends!
Happy Valentine’s Day.
in_love (0)

if determines which statement is executed next

def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
**in_love (1)**

*if* determines which statement is executed next

```python
def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")
```

```python
name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Recipient Name: Kilian
The function `write_valentine` determines which statement is executed next.

```python
def write_valentine(in_love):
    if not in_love:
        print("Let's be friends!")
    print("Happy Valentine's Day.")
```

When `name` is set to "Kilian".

```
Recipient Name: Kilian
```

Global Space:
- `name`: "Kilian"

Call Frame:
- `write_valentine`
- `in_love`: True
- `name`: "Kilian"
if determines which statement is executed next

```python
def write_valentine(in_love):
    if not in_love:
        print("Let’s be friends!")
    print("Happy Valentine’s Day.")
```

```python
name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Recipient Name: Kilian
**in_love (4)**

if determines which statement is executed next

```python
def write_valentine(in_love):
    if not in_love:
        print("Let's be friends!")
        print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name==="Kilian")
```

Recipient Name: Kilian
What does the call frame look like next? (Q)

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

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>

max(0, 3)

A: 

<table>
<thead>
<tr>
<th>max</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
</tbody>
</table>

B: 

<table>
<thead>
<tr>
<th>max</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>RETURN</td>
<td>0</td>
</tr>
</tbody>
</table>

C: 

<table>
<thead>
<tr>
<th>max</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>RETURN</td>
<td>3</td>
</tr>
</tbody>
</table>

D: 

<table>
<thead>
<tr>
<th>max</th>
<th>3</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? (A)

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

Current call frame:

```
max
| 1
x 0
y 3
```

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

(A)
Call Frame Explanation (1)

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

$max(0, 3)$:

```
+---+---+
| max| 1 |
| x  | 0 |
| y  | 3 |
+---+---+
```
def max(x,y):
    if x > y:
        return x
    return y

max(0,3):  

max | 3  
---|---
x | 0  
y | 3  

Skip 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

```
a = 0
if a == 0:
    b = 0
print(b)
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
Control Flow and Variables (Q1)

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(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
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

# 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')

x must have been greater than y
# 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')
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>`
  - ...
- **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
```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
If-Elif-Else (Answer)

```python
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
a = 2  

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

3

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

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.")