CSE 503
Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 05
Boolean Expressions and Control Flow
Announcements

Labs begin next week (9/13)
Lab 01 should be released on Monday on the website (hopefully)
Recap

● Defining our own functions
  ○ **Header:** how the function gets called
    ■ Arguments (values) to the function call get assigned to the parameters (variables) that are declared in the header
  ○ **Body:** what the function does
    ■ Sequence of statements, executed one after another
    ■ Parameters can be used in the body
Example 1

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

Example 2

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

Example 3

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```
## Indentation Example from Piazza

<table>
<thead>
<tr>
<th>Example 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>```python</td>
</tr>
</tbody>
</table>
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
``` |

<table>
<thead>
<tr>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>```python</td>
</tr>
</tbody>
</table>
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
``` |

<table>
<thead>
<tr>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>```python</td>
</tr>
</tbody>
</table>
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
``` |

---

**For each example:**

Which statements are part of the `greet` function?

What will the output be?
**Example 1**

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

**Example 2**

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

**Example 3**

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

For each example:
Which statements are part of the `greet` function?

What will the output be?
Indentation Example from Piazza

Example 1

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

Example 2

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
print("How are you doing today?")
greet("Eric")
```

Example 3

```python
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")
```

For each example:
Which statements are part of the `greet` function? **Indentation matters**
What will the output be?
def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")

def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")

def greet(name):
    print("Hello, " + name)
    print("Welcome to the class!")
    print("How are you doing today?")
greet("Eric")

Example 1

Example 2

Example 3

For each example:
Which statements are part of the `greet` function?

**Indentation matters**

What will the output be?

...example in replit
Three boolean operators in Python: **or**, **and**, **not**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x or y</strong></td>
<td>If ( x ) is False, then ( y ), else ( x )</td>
</tr>
<tr>
<td><strong>x and y</strong></td>
<td>If ( x ) is False, then ( x ), else ( y )</td>
</tr>
<tr>
<td><strong>not x</strong></td>
<td>If ( x ) is False, then True, else False</td>
</tr>
</tbody>
</table>

Both **or** and **and** use short-circuiting: the second expression is only evaluated if it is needed.

**not** has even lower priority than non-boolean operators as well.

[https://docs.python.org/3.7/library/stdtypes.html#boolean-operations-and-or-not](https://docs.python.org/3.7/library/stdtypes.html#boolean-operations-and-or-not)
Boolean Expression Examples

True or False

True or True

a and b

x < y and y <= z
Boolean Expression Examples

True or False...evaluates to True

True or True

a and b

x < y and y <= z
Boolean Expression Examples

True or False...evaluates to True

True or True...also evaluates to True
(or is inclusive in Python)

a and b

x < y and y <= z
Boolean Expression Examples

**True or False**...evaluates to **True**

**True or True**...also evaluates to **True**
(or is inclusive in Python)

**a and b**...evaluates to **True** if both **a** and **b** evaluate to **True**

**x < y and y <= z**
Boolean Expression Examples

True or False...evaluates to True

True or True...also evaluates to True
(or is inclusive in Python)

a and b ...evaluates to True if both a and b evaluate to True

x < y and y <= z ...evaluates to True if x is less than y and y is less than or equal to z
Boolean Expression Examples

True or False ...evaluates to True

True or True ...also evaluates to True
(or is inclusive in Python)

a and b ...evaluates to True if both a and b evaluate to True

x < y and y <= z ...evaluates to True if x is less than y and y is less than or equal to z

Note that x < y <= z is allowable in Python, but not common in other languages
Control Flow

How do we define the order statements in our program are executed?

Control Flow

1. Sequencing
2. Selection
3. Repitition
Control Flow

How do we define the order statements in our program are executed?

This is what we’ve seen so far; executing statements in order.

1. Sequencing
2. Selection
3. Repitition
How do we define the order statements in our program are executed?

**Control Flow**

1. Sequencing
2. Selection
3. Repitition

This is what we’ve seen so far; executing statements in order.

...these are new
Selection Statements

- Selection statements allow our code to *conditionally* execute certain statements in our program
  - This gives our programs the ability to make decisions based on the state of the program
Selection Statements

- Selection statements allow our code to conditionally execute certain statements in our program
  - This gives our programs the ability to make decisions based on the state of the program

Go from this…

…to this
Selection Statements: if statement

if <condition>:
    <body>
Selection Statements: if statement

```
if <condition>:
<body>
```

- **Keyword**: if
- **Boolean expression**: (an expression that evaluates to True or False)
- **Block of one or more statements**: (indented), which are only executed if the condition evaluates to True
if Statement Example

if weather == "snowing":
    print("I’ll wear boots today!")
if Statement Example

Assume `weather` is a variable we’ve already defined

```python
if weather == "snowing":
    print("I’ll wear boots today!")
```
if Statement Example

Assume `weather` is a variable we’ve already defined

```python
if weather == "snowing":
    print("I’ll wear boots today!")
```

We will only execute this line if `weather == "snowing"` evaluates to True
def getReady(weather):
    print("It is " + weather + " out.")
    if weather == "snowing":
        print("I love the snow!")
        print("I’ll wear boots today.")
def getReady(weather):
    print("It is " + weather + " out.")
    if weather == "snowing":
        print("I love the snow!")
        print("I’ll wear boots today.")

Notice the indentation levels...
def getReady(weather):
    print("It is " + weather + " out.")
    if weather == "snowing":
        print("I love the snow!")
        print("I’ll wear boots today.")

...Lets try this in replit
What if it’s not snowing? else

```python
if <condition>:
    <body>
else:
    <body>
```
What if it’s not snowing? else

```
if <condition>:
    <body>
else:
    <body>
```

Another keyword

This body is only executed if `<condition>` evaluates to `False`
Example

def getReady(weather):
    print("It is " + weather + " out.")
    if weather == "snowing":
        print("I love the snow!")
        print("I’ll wear boots today.")
    else:
        print("I’ll wear sneakers today.")
What if we want more than 2 options? 

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```
What if we want more than 2 options? `elif`

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```

You guessed it...another keyword

When does body 2 execute?
What if we want more than 2 options? **elif**

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```

*You guessed it ...another keyword*

*When does body 2 execute?*

*It executes if*  
*<condition 1> is False and*  
*<condition 2> is True*
What if we want more than 2 options? `elif`

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```

You guessed it... another keyword

When does body 2 execute?

It executes if `<condition 1>` is False and `<condition 2>` is True

<br>

`<body 3>` only executes if both conditions are False
What if we want even more?

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```
What if we want even more?

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
elif <condition 3>:
    <body 3>
...
else:
    <body n>
```
What if we want even more?

We can have as many `elif`s as we want.

```python
if <condition 1>:
    <body 1>

elif <condition 2>:
    <body 2>

elif <condition 3>:
    <body 3>

...

else:
    <body n>
```
What if we want even more?

if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
elif <condition 3>:
    <body 3>
...
else:
    <body n>

We can have as many elifs as we want.

The else is optional.

The else is optional.
What if we want even more?

We can have as many `elif`s as we want.

The `else` is optional.

```python
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
elif <condition 3>:
    <body 3>
...
else:
    <body n>
```

Conditions are evaluated from top to bottom.

We execute the body associate with the first condition that evaluates to `True`.

If all conditions evaluate to `False` then we execute the body of the `else`.

Flow of Control

if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
elif <condition 3>:
    <body 3>
else:
    <body n>

Rest of the program
Selection statements...

- Must start with an `if`
- Can be followed with zero or more `elifs`
- Can end with an `else` (or not)
def getReady(weather, temp):
    print("It is " + weather + " out.")
    if weather == "snowing":
        print("I’ll wear boots today.")
    elif temp > 75:
        print("I’ll wear flip flops today.")
    else:
        print("I’ll wear sneakers today.")