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

Indentation Example from Piazza

Example 1

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

Example 2

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

Example 3

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

Indentation Example from Piazza

Example 1

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

Example 2

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

Example 3

```
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

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

Example 2

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

Example 3

```
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

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

Example 2

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

Example 3

```
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?

Indentation Example from Piazza

Example 1

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

Example 2

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

Example 3

```
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?

…example in replit

Boolean Operators/Expressions

Three boolean operators in Python: **`or`**, **`and`**, **`not`**

priority ↓

`x or y`	If **`x`** is **`False`**, then **`y`**, else **`x`**
`x and y`	If **`x`** is **`False`**, then **`x`**, else **`y`**
`not x`	If **`x`** is **`False`**, then **`True`**, else **`False`**

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

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

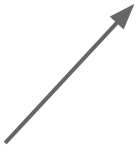
Control Flow

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

Control Flow

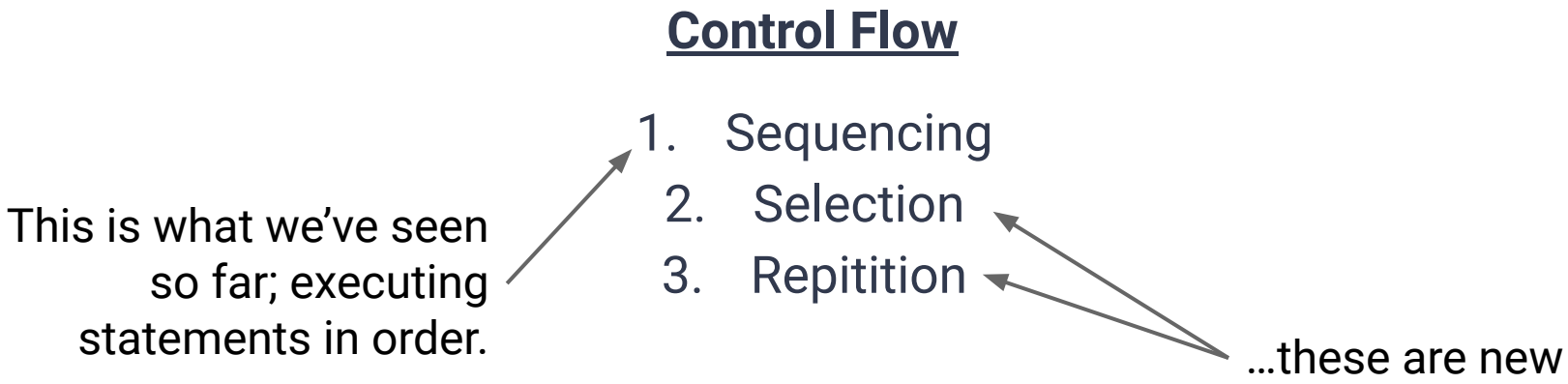
1. Sequencing
2. Selection
3. Repitition

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



Control Flow

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

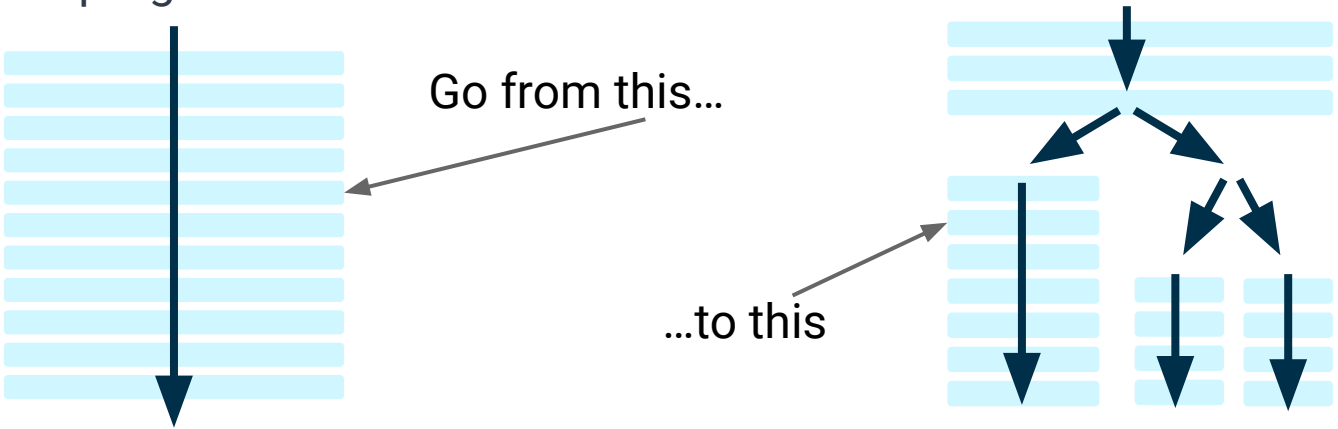


Selection Statements

- 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

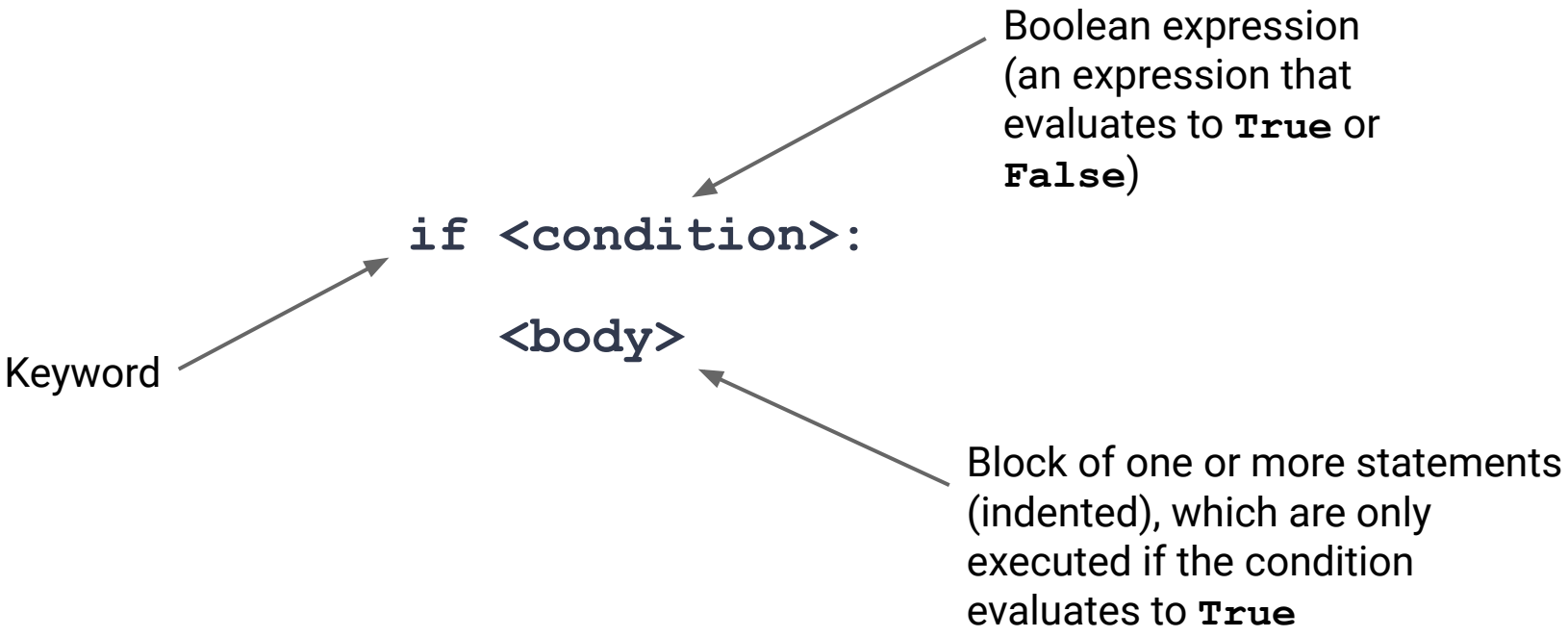
- Selection statements allow our code to *conditionally* execute certain statements in our program
  - This gives our programs the ability to make decisions based on the state of the program



Selection Statements: if statement

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

Selection Statements: if statement



if Statement Example

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

if Statement Example

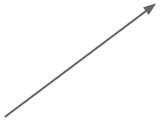
Assume **`weather`** *is a variable we've already defined*

```
if weather == "snowing":

    print("I'll wear boots today!")
```

if Statement Example

Assume **`weather`** *is a variable we've already defined*

```
if weather == "snowing":

    print("I'll wear boots today!")
```



We will only execute this line if **`weather == "snowing"`** evaluates to **`True`**

A More Involved Example

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

A More Involved Example

```
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…

A More Involved Example

```
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

```
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? elif

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

What if we want more than 2 options? elif

```
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

```
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

```
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`**

`<body 3>` only executes if both conditions are **`False`**

What if we want even more?

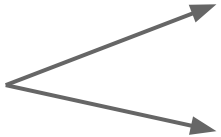
```
if <condition 1>:
    <body 1>
elif <condition 2>:
    <body 2>
else:
    <body 3>
```

What if we want even more?

```
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



```
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

The **`else`** is optional

```
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

The **`else`** is optional

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

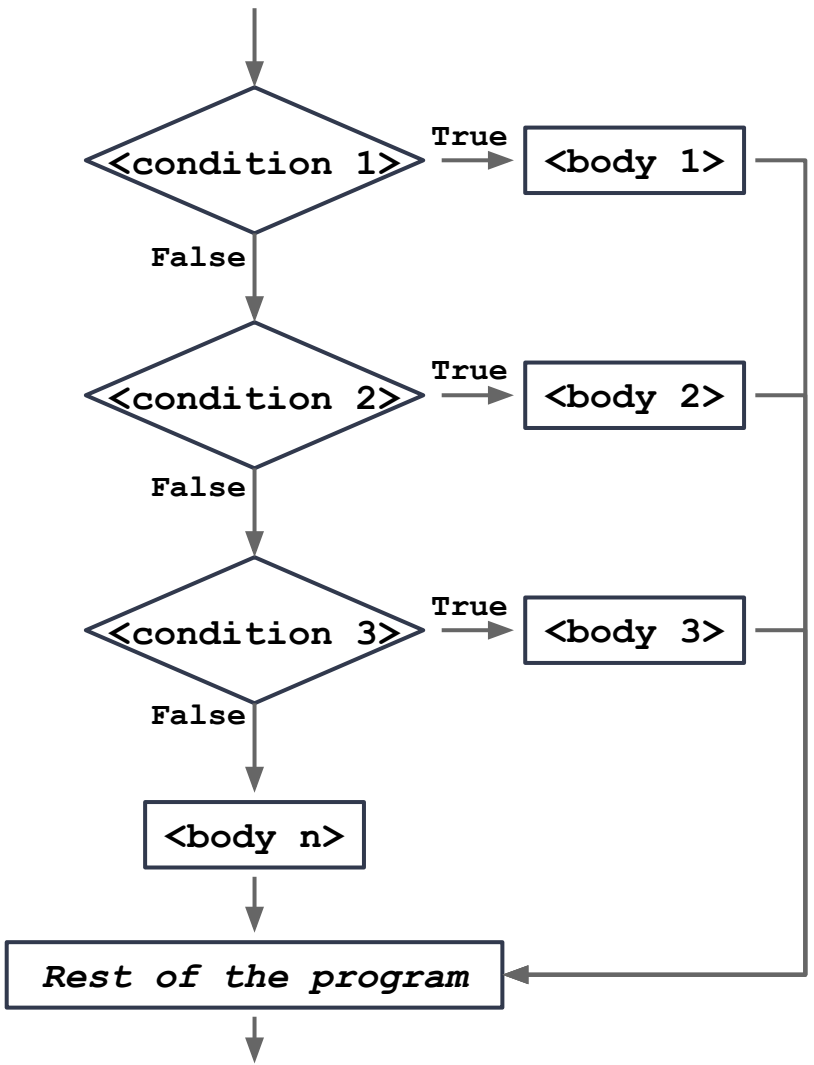
Conditions are evaluated from top to bottom.

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>
```



General Form

Selection statements…

- Must start with an **`if`**
- Can be followed with zero or more **`elif`**s
- Can end with an **`else`** (or not)

One Last Example

```
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.")
```