



CSCI 141

Lecture 9:

Repetition: Repetition, the `while` statement,
Repetition, Repetition, Modules

Announcements

Announcements

- A3 will be out today or tomorrow at the latest.

Announcements

- A3 will be out today or tomorrow at the latest.
 - Due next Wednesday 5/1

Announcements

- A3 will be out today or tomorrow at the latest.
 - Due next Wednesday 5/1
- The prime number question on A2 was updated Friday.

Announcements

- A3 will be out today or tomorrow at the latest.
 - Due next Wednesday 5/1
- The prime number question on A2 was updated Friday.
 - Tip: write a program to check your answer!

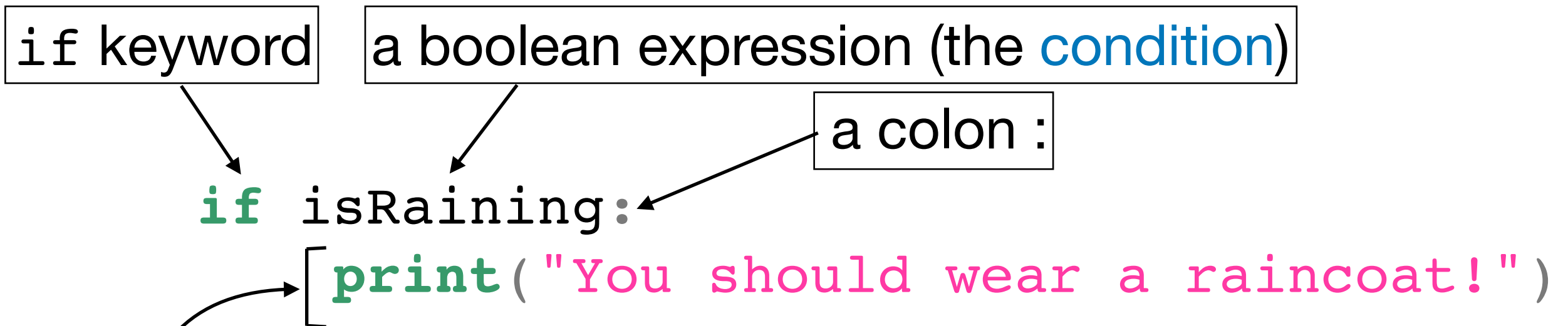
Announcements

- A3 will be out today or tomorrow at the latest.
 - Due next Wednesday 5/1
- The prime number question on A2 was updated Friday.
 - Tip: write a program to check your answer!
- Office hours today: I'll be late, but will also be in my office 4—5 today.

Goals

- Understand the syntax and behavior of the `while` statement (also known as `while` loop).
- Know how to use `import` statements to get access to `modules` containing functions that other people have written.
- Know how to use the `random` module's `randrange` function.

Last time: `if` statements



an indented **code block**: one or more statements to be executed if the boolean expression evaluates to **True**

Last Time: Chained Conditionals

elif keyword



```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    [print("No rain gear needed!)]
```

Last Time: Chained Conditionals


elif keyword



```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    [print("No rain gear needed!)]
```

an indented
code block
to be executed if:

- **none** of the above conditions was True
- **and** this `elif`'s condition is True



Last Time: Chained Conditionals

elif keyword

```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    print("No rain gear needed!")
```

an indented code block to be executed if:

- **none** of the above conditions was True
- **and** this `elif`'s condition is True

an indented code block to be executed if the **none** of the above conditions was true

Last Time:

Chained Conditionals

elif keyword

```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    print("No rain gear needed!")
```

an indented code block to be executed if:

- **none** of the above conditions was True
- **and** this `elif`'s condition is True

an indented code block to be executed if the **none** of the above conditions was true

(this behaves exactly like nesting an if inside each else)

Last Time:

Chained Conditionals

elif keyword

```
if isRaining and not isWindy:  
    print("Bring an umbrella!")  
elif isRaining and isWindy:  
    print("Wear a raincoat!")  
else:  
    print("No rain gear needed!")
```

an indented code block to be executed if:

- **none** of the above conditions was True
- **and** this `elif`'s condition is True

(this behaves exactly like nesting an if inside each else)

an indented code block to be executed if the **none** of the above conditions was true

(the else clause is optional)

Today's Quiz

- 3 minutes

Today's Quiz

- 3 minutes
- Working with a neighbor: do your answers agree? (2 minutes)

Today: Repetition

- So far, we've seen how to:
 - Print things to the screen and replace your calculator
 - Represent complicated boolean expressions and execute different code based on their truth values.
- So far we *haven't* seen how to:
 - Do anything that you couldn't do yourself, given pencil and paper and a few minutes to step through the code.

Motivation

Anyone really good at tongue twisters?

Pad kid poured curd pulled cod.

Pad kid poured curd pulled cod.

Pad kid poured curd pulled cod.

Pad kid poured curd pulled cod.

Pad kid poured curd pulled cod.

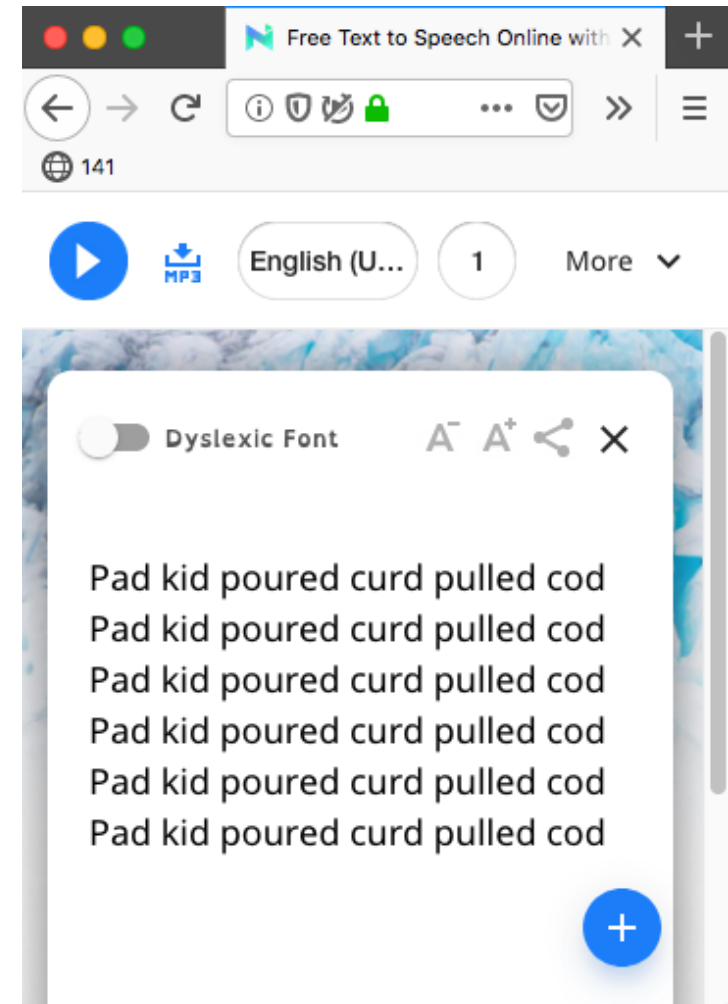
This is (according to MIT psychologists*) the hardest known tongue twister.

Fact: humans are **bad** (or at least slow) at performing repetitive tasks.

Motivation

Fact: humans are **bad** (or at least slow) at performing repetitive tasks.

<https://www.naturalreaders.com/online/>



Fact: computers are good (or at least fast) at performing repetitive tasks.

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance after five years?

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

balance = 100.00

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
```

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
```

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
balance = balance + (0.02 * balance)
print(balance) # year 3
```


Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
balance = balance + (0.02 * balance)
print(balance) # year 3
balance = balance + (0.02 * balance)
print(balance) # year 4
```

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
balance = balance + (0.02 * balance)
print(balance) # year 3
balance = balance + (0.02 * balance)
print(balance) # year 4
```

uh oh...
my font is
getting small

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
balance = balance + (0.02 * balance)
print(balance) # year 3
balance = balance + (0.02 * balance)
print(balance) # year 4
balance = balance + (0.02 * balance)
print(balance) # year 5
```

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for five years?

```
balance = 100.00
balance = balance + (0.02 * balance)
print(balance) # year 1
balance = balance + (0.02 * balance)
print(balance) # year 2
balance = balance + (0.02 * balance)
print(balance) # year 3
balance = balance + (0.02 * balance)
print(balance) # year 4
balance = balance + (0.02 * balance)
print(balance) # year 5
```

argh, ok, done.

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for **500** years?

An extremely common task: do the same thing over and over again, or do the same processing on many pieces of data.

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for **500** years?

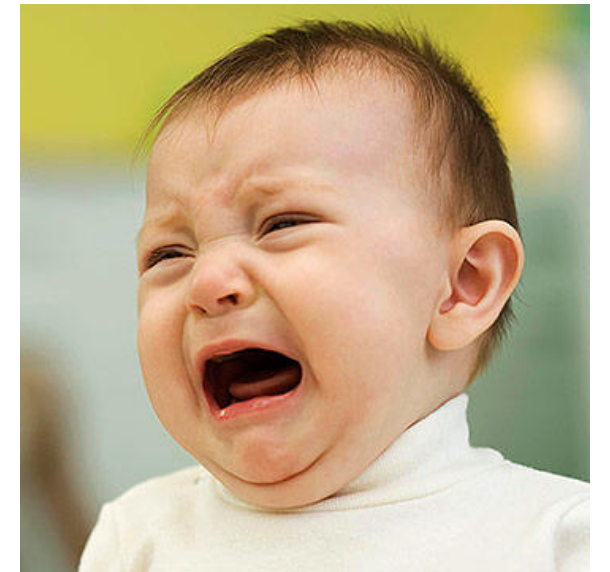
...

An extremely common task:
do the same thing over and
over again, or do the same
processing on many pieces
of data.

Motivation

Suppose you have a starting bank account balance of \$100.00, and your account earns 2% interest each year.

What is your balance each year for **500** years?

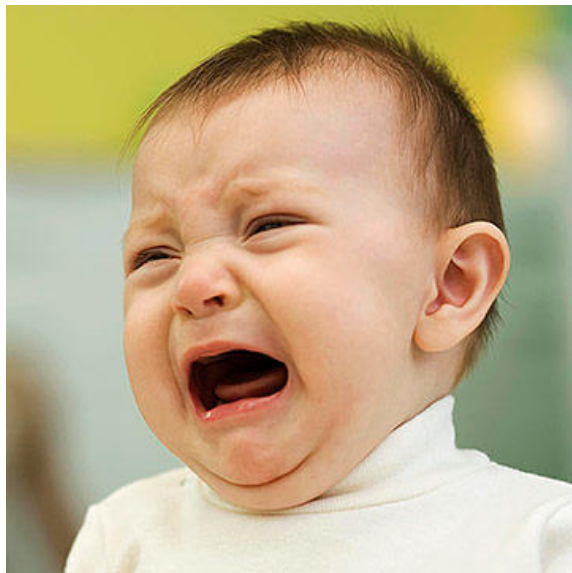


...

An extremely common task: do the same thing over and over again, or do the same processing on many pieces of data.

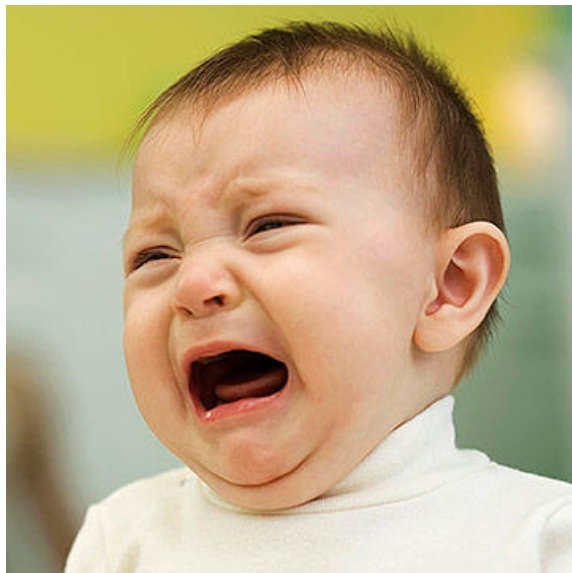
Motivation

Example: Convert this 100x100 pixel image to grayscale (“black-and-white”).



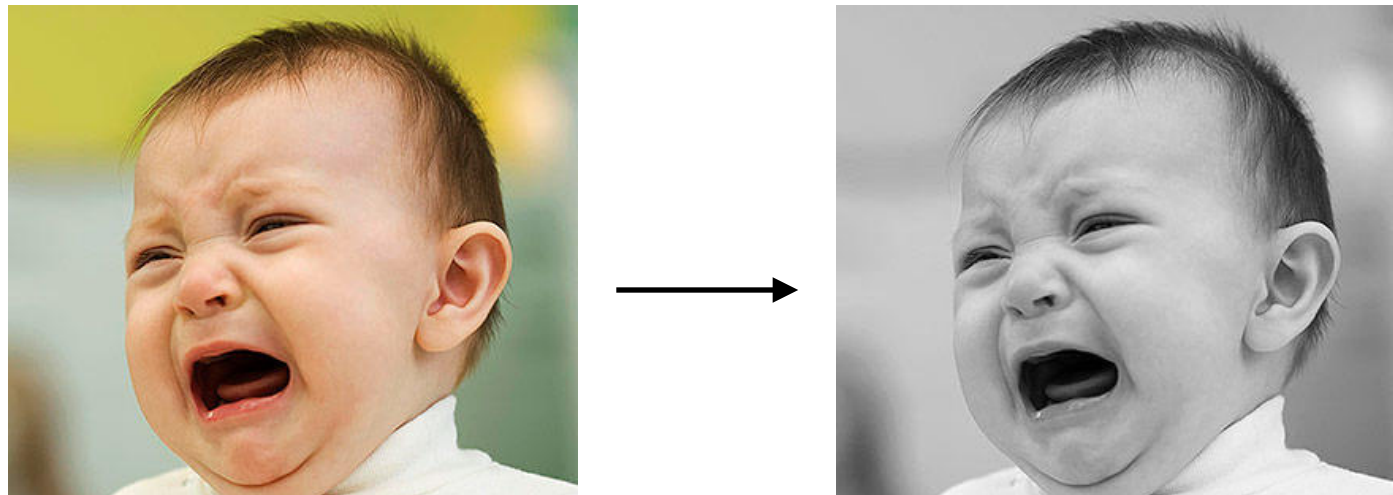
Motivation

Example: Convert this 100x100 pixel image to grayscale (“black-and-white”).



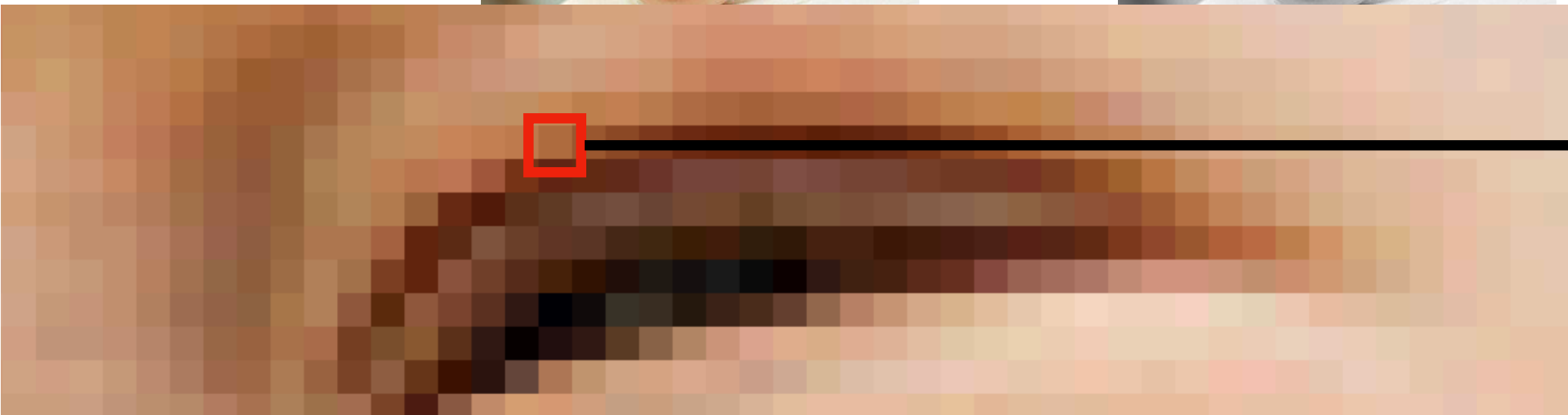
Motivation

Example: Convert this 100x100 pixel image to grayscale (“black-and-white”).



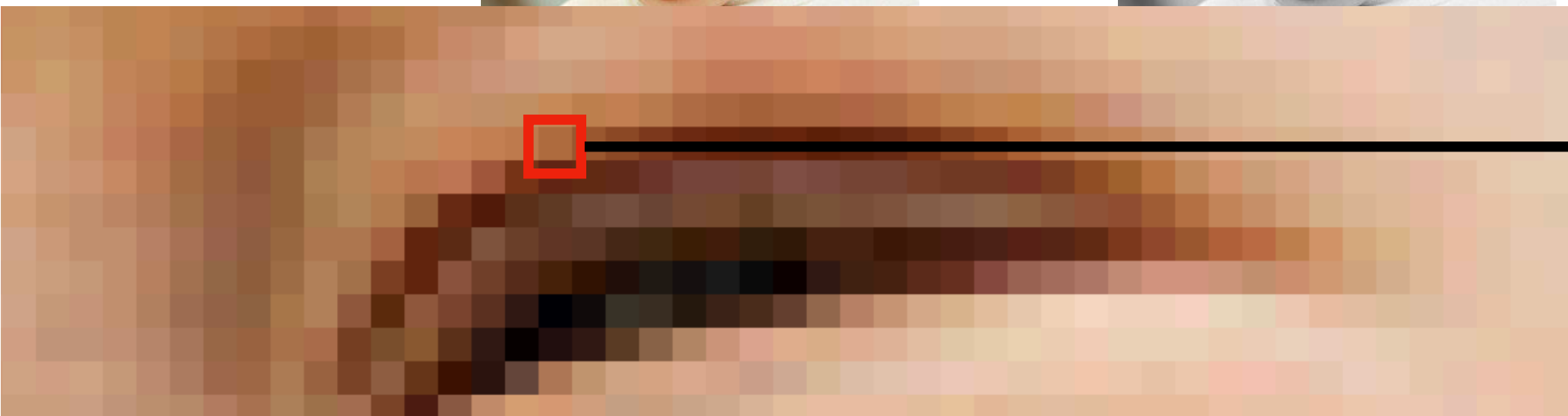
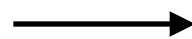
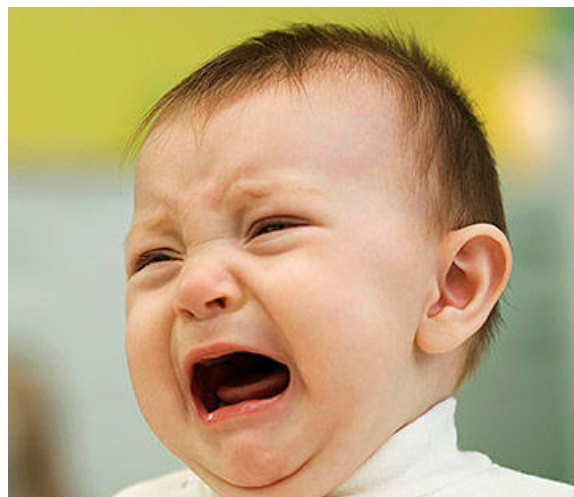
Motivation

Example: Convert this 100x100 pixel image to grayscale (“black-and-white”).



Motivation

Example: Convert this 100x100 pixel image to grayscale (“black-and-white”).



10,000 pixels, same calculation:

$$\text{grey} = 0.29 * \text{red} + 0.59 * \text{green} + 0.12 * \text{blue}$$

Python to the rescue: the `while` statement

Not so different from an `if` statement:

`if` keyword

a boolean expression (the `condition`)

a colon `:`

```
if year <= 5:  
    balance = balance + (0.02 * balance)  
    print(balance)
```

an indented `code block`: one or more statements to be executed if the boolean expression evaluates to **True**

Python to the rescue: the `while` statement

Not so different from an `if` statement:

`while` keyword

a boolean expression (the `condition`)

a colon `:`

```
while year <= 5:  
    balance = balance + (0.02 * balance)  
    print(balance)
```

an indented `code block`: one or more statements to be executed **while** the boolean expression evaluates to True

The `while` statement: A Working Example

```
# print account balance after each
# of five years:
balance = 100.0 # starting balance
year = 1
while year <= 5:
    balance = balance + (0.02 * balance)
    print(balance)
    year = year + 1
```

demo: interest

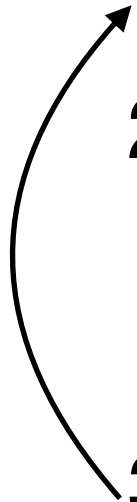
- `balance1.py`: the tedious way
- `balance2.py`: the loopy way

The `while` statement: Semantics (Behavior)

If statement:

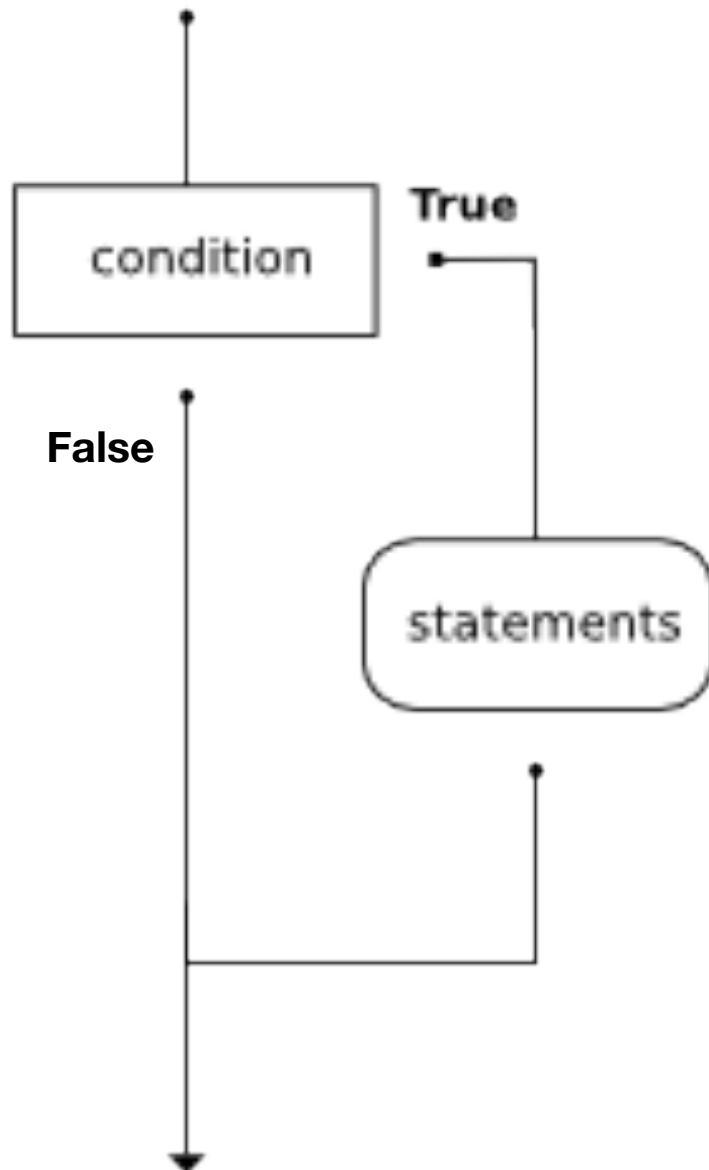
1. Evaluate the condition
2. If true, execute body (code block), then continue on.

While statement:

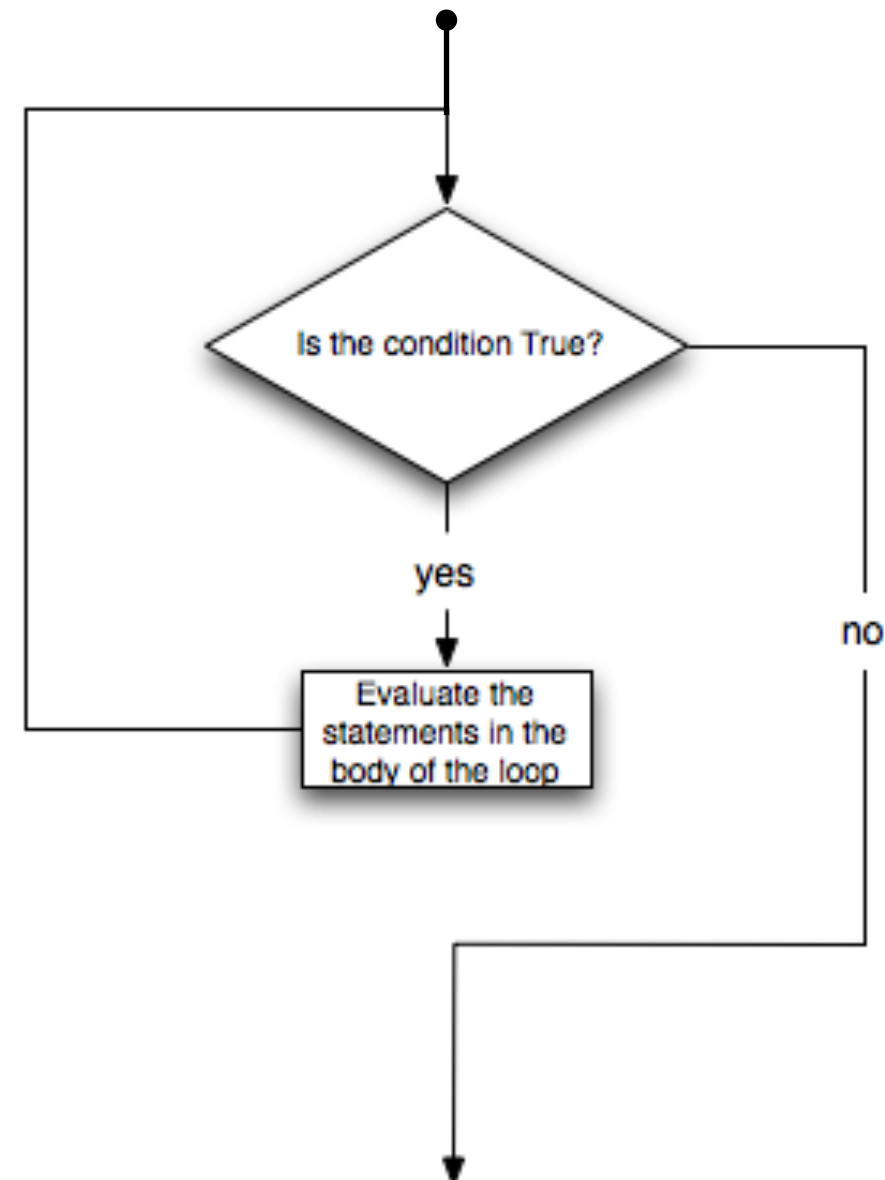
1. Evaluate the condition
 2. If true, execute body, otherwise skip step 3 and continue on.
 3. Go back to step 1
- 

The *while* statement: Semantics (Behavior)

If statement:



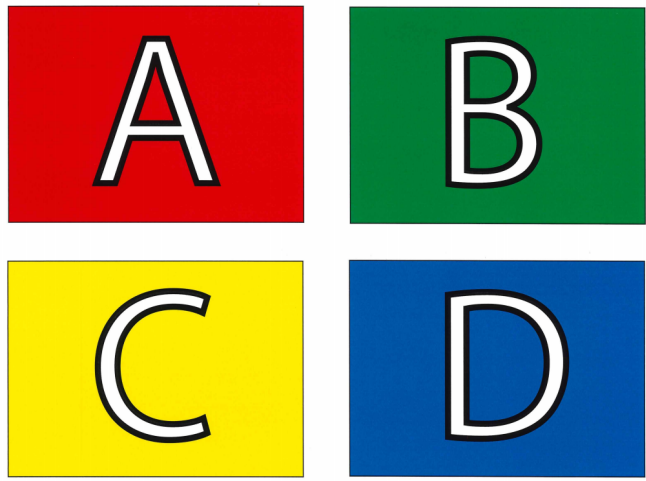
While statement:



Exercise

Task: Find how many times you can double the number 1 before it exceeds 1000.

```
times = 0
n = 1
while [condition here]:
    n = n * 2
    times = times + 1
print(times, "times!")
```



Exercise

Task: Find how many times you can double the number 1 before it exceeds 1000.

```
times = 0
n = 1
while [condition here]:
    n = n * 2
    times = times + 1
print(times, "times!")
```

Which of the following conditions is correct?

- A. `times < 1000`
- B. `times <= 1000`
- C. `n > 1000`
- D. `n <= 1000`

Aside: In-Place Operators

When writing loops (and code in general), you'll find yourself doing things like this often:

```
count = count - 1  
sum = sum + n
```

Python has a nice shorthand for this:

```
count -= 1  
sum += n
```

Many math operators have an in-place version:

```
+=, -=, /=, //=, %=
```

Aside: In-Place Operators

When writing loops (and code in general), you'll find yourself doing things like this often:

```
count = count - 1
sum = sum + n
```

Python has a nice shorthand for this:

```
count -= 1
sum += n
```

Many math operators have an in-place version:

`+=`, `-=`, `/=`, `//=`, `%=`

[**No**, Python doesn't have increment and decrement operators `++` and `--`]

Demo

Demo

- `count.py`:
 - Counting up, counting down by an interval
- `never.py`:
 - Condition never True
 - Condition never False
- `input.py`:
 - sum user-provided positive numbers until a negative number is entered