Let's Learn Python!

No, it's not a snake!

Meet your teacher:

Chris Bradfield

introduction, talk about what I do with programming

What is programming?

Who knows what a computer program is?

- A **program** is a detailed set of **instructions** telling a computer exactly what to do.

Instructions for people:

"Clean your room."
- my mom, 1982

"Sit quietly and listen."
- your teacher, every day

"I'll have a burger with cheese, pickles and onions."
- me, at the drive-thru

Instructions to people are simple, because people know what you mean.
How many of you think computers are smart? (show of hands)
Nope, computers are actually really dumb – they'll only do exactly what you tell them to do. So you have to be the smart one and give them good instructions. That's what we're going to learn about today.

Algorithms

Computer programmers like to use fancy words for simple things.
An algorithm is just a recipe, with specific steps to follow.

97 Simple Steps to a PB&J

Is making a PB&J difficult?

How many steps does it feel like?

PB & J Demonstration
Kids call out directions – write on whiteboard/computer

Let's talk Python!

"Python is just one of the languages we use to talk to computers, but there are many others."
- Open Idle
- Briefly talk about arranging the window, etc

Python is a language

There are many different programming languages

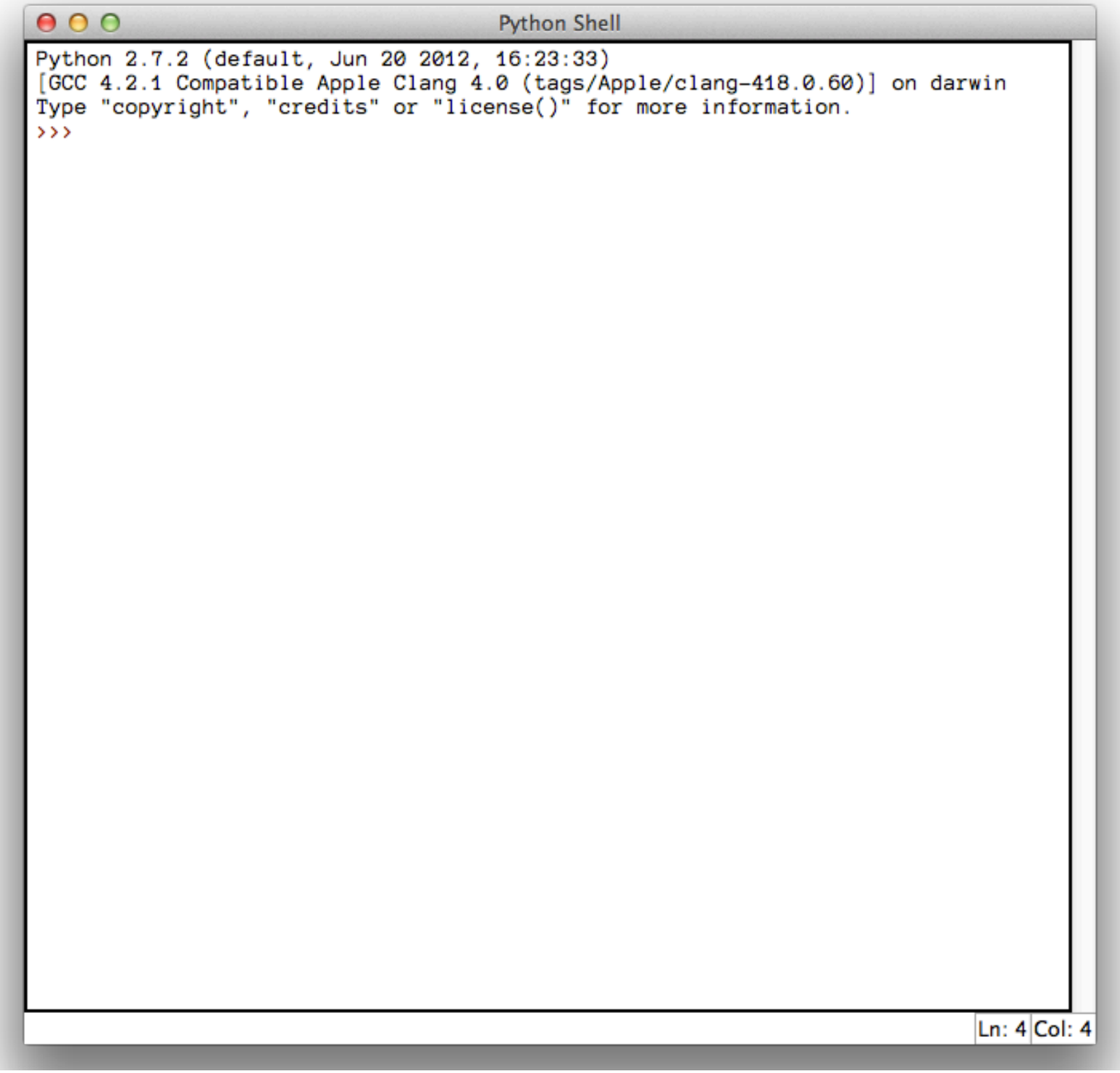
Python can be used to make:

- Games
- Websites
- Robots
- Science Projects

Starting Python

Double-click on “IDLE”

Python Shell



What's a shell?
Make sure everyone sees this on the screen

Python

This is called the “prompt”: >>>

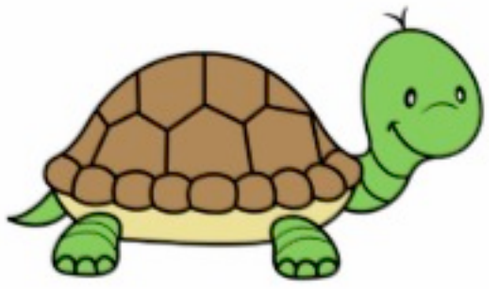
It means the computer is waiting for you to tell it to do something.

Let’s try this together:

```
>>> print(“Hello, World!”)
```

There’s a tradition in programming – your first program says “Hello, World”
You’ve made your first program!

TURTLES!



Turtles

Let's make a turtle:

```
>>> import turtle
>>> fred = turtle.Pen()
```

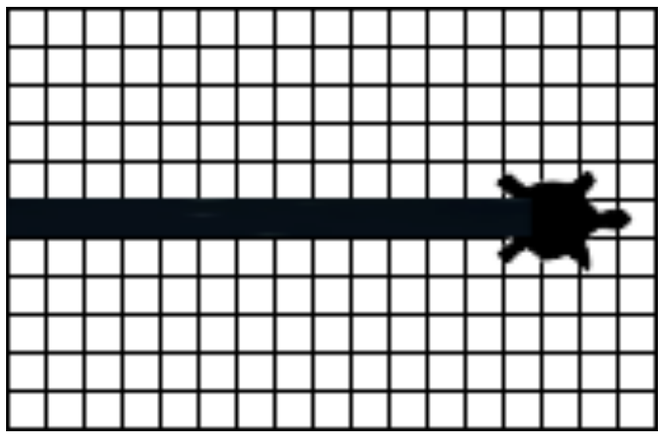
Do you see your turtle?

Confirm everyone has a turtle window on screen
Pause for moving windows around

Turtles

Turtles can draw!

```
>>> fred.forward(50)
```



```
>>> fred.left(90)

>>> fred.forward(50)
```

What's a pixel?

Turtles

Errors

```
>>> fred.forwad(50)
```

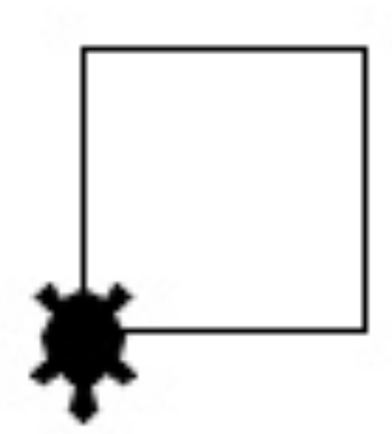
```
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    fred.forwad(50)
AttributeError: 'Turtle' object has no attribute 'forwad'
```

What happens when we make a mistake?
Don’t be scared of errors, the computer doesn’t mind.
Don’t worry about all the confusing stuff in red, we’ll talk about that later

Turtles

What do you think this will draw?

```
>>> fred.forward(50)
>>> fred.left(90)
>>> fred.forward(50)
>>> fred.left(90)
>>> fred.forward(50)
>>> fred.left(90)
>>> fred.forward(50)
```



Wait for guesses and trying

Turtles

Changing color:

```
>>> fred.reset()
```

```
>>> fred.color(“red”)
```

```
>>> fred.forward(50)
```

Pause on each and ask what happened

Turtles

Picking up and putting down the pen:

```
>>> fred.reset()
>>> fred.up()
>>> fred.forward(50)
>>> fred.down()
>>> fred.forward(50)
```

up() and down() tell the turtle whether to draw while he moves

Pause on each and ask what happened

Turtles

Circles:

```
>>> fred.reset()
>>> fred.circle(100)
>>> fred.circle(-100)
>>> fred.left(90)
>>> fred.circle(100)
>>> fred.circle(-100)
```

Pause on each and ask what happened

Turtles

Now let's use everything we've learned:

```
forward(?)
left(90), right(90)
circle(?)
color(?)
up()/down()
```

And one more bonus command:

```
>>>fred.width(5)
```

What can you draw?

Let the kids try their own commands – encourage them to change the numbers, etc.

Math

Math

Arithmetic operators:

addition: +

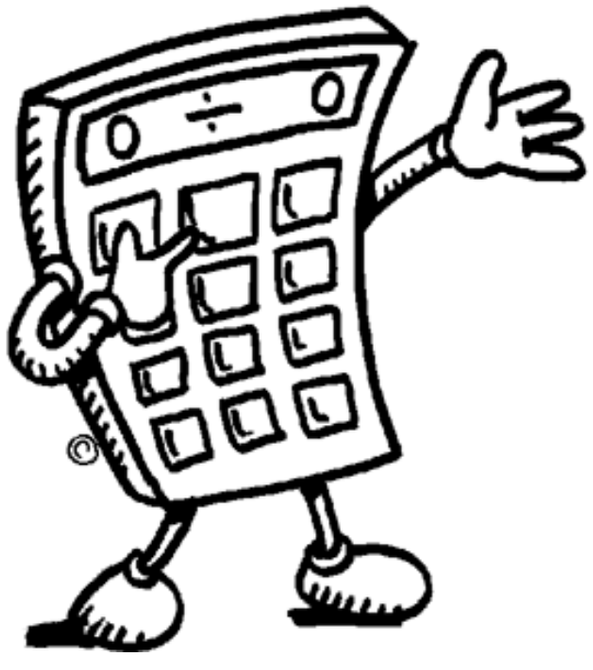
subtraction: -

multiplication: *

division: /

Try doing some math:

```
>>> 1 + 2
>>> 12 - 3
>>> 6 * 5
>>> 10 / 2
```



Term to introduce: expression
Is the last result what you expected?

Math

Comparison operators:

==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

“The exclamation point is also sometimes called a ‘bang’.”

Math

Comparison practice:

```
>>> 1 + 1 == 2
>>> 5 < 4 + 3
>>> 12 + 1 >= 12
>>> 8 * 2 == 10
>>> 5 >= 6
```

Guess the answer, then try in the Python shell.

Math

Comparison practice:

```
>>> 1 + 1 == 2          True
>>> 5 < 4 + 3           True
>>> 12 + 1 >= 12        True
>>> 8 * 2 == 10         False
>>> 5 >= 6              False
```

Strings

Strings

```
>>> “garlic breath”
>>> “Thanks for coming!”
```

Try typing one without quotes:

```
>>> apple
```

What’s the result?

If it’s a string, it must be in quotes.

```
>>> “How do dinosaurs pay
their bills?”
>>> “With tyrannosaurus
checks!
```

A string is a character, or a sequence of characters
A number can be a string, if it’s wrapped in quotes

Strings

String operators:

concatenation (adding words together): +

multiplication: *

Try concatenating:

```
>>> “Hi” + “there!”
'Hithere!'
```

Try multiplying:

```
>>> “HAHA” * 250
```

Strings: Indexes

Strings are made up of **characters**:

```
>>> “H” + “e” + “l” + “l” + “o”
‘Hello’
```

Each character has a position called an *index*:

```
H e l l o
0 1 2 3 4
```

In computers, indexes start at **0**

Spend some time on counting from 0 versus from 1

Strings: Indexes

```
>>> “Hello”[0]
‘H’
>>> “Hello”[4]
‘o’
```

```
>>> “Hey, Bob!”[6]
‘o’
>>> “Hey, Bob!”[6 - 1]
‘B’
```

Strings: Indexes

```
>>> "Hey, Bob!"[4]
```

What did Python print?

```
H e y ,   B o b !
0 1 2 3 4 5 6 7 8
```

Rules:

- ★ Each character's position is called its *index*.
- ★ Indexes start at 0.
- ★ Spaces inside the string are counted.

Questions?

Variables

Variables

Calculate a value:

```
>>> 12 * 12
144
```

How can you save that value, `144`?

Assign a name to a value:

```
>>> donuts = 12 * 12
>>> color = "yellow"
```

A **variable** is a way to *store a value*.

Variables

```
>>> donuts = 12 * 12
>>> donuts
>>> color = “yellow”
```

Assign a <u>new</u> value:

```
>>> color = “red”
>>> donuts = 143

>>> color = “fish”
>>> color
>>> color = 12
>>> color
```

Once you have a variable, you can change its value to anything else

Variables

Some other things we can do with variables:

```
>>> coins = 20
>>> magic_coins = 10
>>> coins + magic_coins
30
>>> magic_coins = 5
>>> coins + magic_coins
25

>>> fred = turtle.Pen()  (remember that?)
```

Demonstrates using a variable as a value in mathematical operations
save the value for later
keep the name, but change the value

Lists

Lists

List: a sequence of objects

```
>>> fruit = [“apple”, “banana”, “grape”]
>>> numbers = [3, 17, -4, 8.8, 1]
```

Guess what this will output:

```
>>> wizardlist = [“slug slime”, “bat
wing”, “spider leg”, “eye of newt”]
>>> wizardlist[1]
```

Lists

Lists have indexes just like strings.

```
>>> wizardlist[1]
'bat wing'
```

```
>>> wizardlist
['slug slime', 'bat wing', 'spider
leg', 'eye of newt']
```

Lists

Make a **list** of the three primary colors.

```
>>> colors = [‘blue’, ‘red’, ‘yellow’]
```

Use an **index** to print your favorite color’s name.

```
>>> colors[1]
```

Logic

`if` Statements

When we talk about logic, we're talking about making decisions about what to do next in our code.

One of the ways we do that is with "if statements".

`if` Statements

Making decisions:

"**If** the bell rings, it's time for class to end."
"**If** the trash is full, go empty it."

If the condition is met, perform the action that follows:

```
>>> name = "Jesse"
>>> if name == "Jesse":
        print("Hi Jesse!")

Hi Jesse!
```

Mention indentation here

`if` Statements

Adding more choices:

"**If** you're not busy, let's eat lunch now.
Or **else** we can eat in an hour."

"**If** there's mint ice cream, I'll have a scoop.
Or **else** I'll take butter pecan."

The `else` clause:

```
>>> if name == “Jesse”:
        print(“Hi Jesse!”)
    else:
        print(“Impostor!”)
```

blocks

A **block** is a group of statements.

```
>>> age = 20
>>> if age > 10:
        print(“You are too old!”)
    print(“Why are you here?”)
```

You tell the computer about a **block** by indenting (usually 4 spaces). The 2 `print` statements are a block.

If age > 10, then the computer does what’s in the block.

Another example from the demo:

IF the lid is on the peanut butter jar:
    take it off

Try it

What do you think this code will do?

```
>>> money = 2000
>>> if money > 1000:
        print(“I’m rich!”)
    else:
        print(“I’m not rich.”)
        print(“But maybe someday.”)
```

Encourage thinking about the answer before trying to type it in.

Loops

Loops

Loops are chunks of code
that repeat a task over and over again.

★*Counting* loops repeat a
certain number of times.

★*Conditional* loops keep
going until a certain thing happens
(or as long as some condition is True).

Programmers hate doing the same thing over and over again.

Loops

Counting loops repeat a certain number of times.

```
>>> for number in range(5):
        print("Hello", number)

Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```

Remember, computers like to start counting at 0.

Break down the "for" statement – for each element in the list

Loops

Remember when we drew a square with the turtle?

```
>>> fred.forward(50)
>>> fred.left(90)
```
1

```
>>> fred.forward(50)
>>> fred.left(90)
```
2

```
>>> fred.forward(50)
>>> fred.left(90)
```
3

```
>>> fred.forward(50)
>>> fred.left(90)
```
4

8 lines of code and 4 times repeating!

Very repetitive! Make sure they get it: doing the same thing 4x

Loops

How do we make the square with a loop?

```
>>> for i in range(4):
        fred.forward(50)
        fred.left(90)
```

Do it 4 times

Block

We changed 8 lines of code into only 3!!

Break it down – this is much easier to change (bigger square,etc)
'i' is usually used for loops

Loops

Experimenting - any guesses what this will do?

```
>>> for number in range(6):
        fred.forward(50)
        fred.left(60)
```

Try it!

Break it down - this is much easier to change (bigger square,etc)
Questions?

Loops

Conditional loops repeat until something happens.

```
>>> count = 0
>>> while (count < 4):
        print 'The count is:', count
        count = count + 1

The count is: 0
The count is: 1
The count is: 2
The count is: 3
```

The *while* keyword is used to create this kind of loop, so it is usually just called a *while loop*.

Point out how the counter is increasing each time we go through the loop

Talk about infinite loops

Save your work

Saving a Program

What if we want to save our square program so we don't have to type it in again?

File > New Window

<u>Type this:</u>

```
import turtle
fred = turtle.Pen()

for number in range(4):
    fred.forward(50)
    fred.left(90)
```

File > Save

Just like saving a document in Word

Saving a Program

Now we can open and run our program whenever we want.

Run > Run Module