but master, why?

What’s the point of classes and objects?

Encapsulating data and functions

- A class can be a module that groups some data with related functions
  - A convenient unit of reuse and change
  - Often a reasonable brain-size chunk (if designed well)

But classes aren’t the only kind of module ... and that still doesn’t explain the wacky backward syntax object.method(arg, ...)

why not function(object, arg, ...)?

We don’t want this:

```javascript
total = 0;
for shape in shapelist:
    if is_square(shape) :
        total = total + square_area(shape)
    elif is_triangle(shape) :
        total = total + tri_area(shape)
    elif is_circle(shape):
        total = total + circ_area(shape)
```

Imagine we have a lot of code like this, and then we need to add support for ellipses. Yuck.

Polymorphism in method dispatch

- poly = many, morph = form
- same method call can work for several different kinds of data
- Example: suppose we have a list of shapes, including triangles, squares, and circles. We want to add up their areas.
Better:

total = 0;
for shape in shapelist:
    total = total + shape.area()

There is still an area function for squares, and an area function for
triangles, and one for circles ... but each one is a method in the
corresponding class.

If we need to add a class for ellipses,
no change needed here!

We’ve localized some kinds of program changes, which is a GOOD
THING (even worth putting up with wacky syntax)

Why backward you must write?

```
total = 0;
for shape in shapelist:
    total = total + shape.area()
```

Object-oriented = Good ?

• No!
  – Often useful, not always. Helps with certain kinds
    of program evolution
  – Not the only way to modularize programs. Not
    always the best way

• Widely used. You need to learn it, but you
  don’t always have to choose it.

A bit of history ...

• “Object-oriented” features (classes and objects) were
developed in SmallTalk and spread to many other
languages
• Originally for simulation (Simula 67), then for
  modularity
OO and Java

• Java programs are collections of classes
  – Methods only.
    No functions outside of classes.
    Closest thing to Python function is a “static” method (like “class” methods in Python)

Summary

• Classes and Objects provide a way to divide programs into modules
  • Grouping data and related functions into meaningful, potentially reusable chunks

• Polymorphic, dynamic dispatch in method calls helps localize changes
  • Adding a new, related class is simpler than changing a lot of code to call the appropriate functions

• A widely used approach
  • Often useful, worth learning, but not always best