Classes and Objects

Namespaces are great. I want more!

More!

(And I don’t want to write a different module file for each one. I want to have a bunch of objects, each with their own namespace.)

A class is a lot like a module.

From one class, we make as many objects as we want. Each one has its own namespace ("attributes", including functions and variables)

You’re already using classes and objects

my_list = [5, 7, 9]
my_list.pop()

my_list = [5, 7, 9]
my_list.pop()
You can create your own ...

class Zombie:
    age = 128
    name = "Eton Z. Brains"
    wants = "I want to eat brains"
    def identify(self):
        return "I am " + self.name + " and " + self.wants

Why define classes and objects?

Grouping data with functions:
    Maybe each Boggle tile has a string and an "in use" flag. Maybe I want to access them together.

Same advantage as modules, but I can have a list of tiles.
    Especially useful if I have different kinds of tiles with similar behavior (same function names)
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, ...)?

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.
**We don’t want this:**

```python
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.*

---

**Better:**

```python
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?**

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

---

**Object-oriented = Good ?**

It’s not that simple.

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.