Classes and Objects

Christopher Pitts

CSE 210

Overview

What Problem Do Classes Solve?

What Is A Class?

Python Classes

What Problem Do Classes Solve?

Picture This

You have data structures that contain data (naturally). You also have functions that operate on those data structures. Problematically, those functions and data structures can be updated independently, and it is very easy to update a structure and forget to update the corresponding class. They might not even be in the same file, which compounds the problem.

Picture This

Wouldn't it be nice if you could reason about the data structures and the functions that operate on them as a logical unit? And furthermore, wouldn't it be nice if some parts of that data could *only* be modified by the functions that were supposed to be modifying them, so you couldn't even do it accidentally?

What Is A Class?

Definition

A *class* is the unification of data structures and the methods that operate on them, usually with some level of access control that prevents unintended (or well-intentioned but misguided) modification of the data.

Solving A Problem

Classes provide a solution to the problem statement from previous slides. Because a class describes both data structures and the functions that operate on them, you can reason about them together. And, if your language is statically typed/compiled, you can know when you've forgotten to update the functions and/or the data structures!

Python Classes

Why Do Python Classes Behave The Way They Do?

Python classes have some interesting semantics:

1. Required use of “self” as the first parameter[1]
2. On-the-fly attribute definition
3. Complete lack of privacy

[1] There are exceptions to this, and it doesn’t have to be called “self”.

Why Do Python Classes Behave The Way They Do?

To really understand this behavior, we need to start with C programming.

The Old Days: C Programming

C is an old programming language (but still very useful and important, you should spend the time to learn it). Notably, it predates the dominance of object-oriented programming languages, and does not have classes at all.

Now C Here

Some important characteristics of C for this discussion:

1. Statically typed
2. Compiled
3. Structs: collections of data structures
4. Functions

Solving A Problem, The C Way

C programmers were aware of the problem addressed earlier[2].
So, to solve this problem, they defined a convention.

[2] Being C programmers, many of them pretended the problem didn't exist as long as it wasn't affecting them.

Meet Struct, The C Class

```
struct Foo
{
  int bar;
  float baz;
};
```

A *struct* is a collection of data structures. The first part of the convention is that structs contain all of the data you want to keep together.

One Struct, Please

```c
#include "foo.h"

void fooBazTheBar(struct Foo* f)
{
  f->bar *= f->baz;
}

void fooUpTheBar(struct Foo* f, int u)
{
  f->bar += u;
}

void fooInit(struct Foo* f, int bar, float baz)
{
  f->bar = bar;
  f->baz = baz;
}
```

The second part of the convention is that functions that are supposed to operate on the data structures are named using the name of the struct type (usually), and take as the first parameter an instance of the struct.

One Struct, Coming Up

```
void fooInit(struct Foo* f, int bar, float baz)
{
  f->bar = bar;
  f->baz = baz;
}
```

This function is worth a second look; it's analogous to a constructor in other programming languages, taking in the initial values and setting them.

From One Good Thing Comes Another

Python is implemented in C[3], and therefore takes a lot of design cues and implementation details from that language.

[3] In fact, the reference implementation is called CPython; that's the version you install when you get it from python.org

Look In The Mirror

```c
void fooInit(struct Foo* f, int bar, float baz)
{
  f->bar = bar;
  f->baz = baz;
}
```

```python
class Foo:
    def __init__(self, bar, baz):
        self.bar = bar
        self.baz = baz
```

Do these two methods look similar to you?

Python Classes Are Just Fancy C Structs

Python classes work exactly as C structs work.

1. That “self” parameter is just an instance of the class type
2. You set all the values in the init method because that’s how you would do it in C
3. Python made a conscious choice to do this, but it’s very C to say “all the data is public, and we’re all consenting adults”[4]

[4] Literally how one person explained it on Stack Overflow

But Wait...

You're probably thinking about the fact that Python doesn't look quite like this:

```
int main(int argc, char* argv[])
{
  struct Foo f;
  initFoo(f, 1, 0.5);
  fooBazTheBar(f);

  return 0;
}
```

Pour Some Sugar On That Struct

Python uses some *syntactic sugar* to simplify the process of writing classes this way. These are equivalent:

```
class Foo:
    def __init__(self, bar, baz):
        self.bar = bar
        self.baz = baz

    def baz_the_bar(self):
        self.bar *= self.baz

f = Foo(1, 0.5)
f.baz_the_bar()
Foo.baz_the_bar(f)
```

QED: Python Classes Are Just Fancy C Structs

Man, Kipling Was A Visionary

This convention happens to be explicitly codified in a modern programming language that debuted to much acclaim: Golang does this type of method resolution as well. And in fact compels you to write out code exactly how you would in C, because Golang doesn't have classes[5].

[5] This was apparently on purpose.

Man, Kipling Was A Visionary

```
type Foo struct {
        foo int
        bar float
}

func (Foo f) bazTheBar() {
        f.foo *= f.baz
}
```

After fifty years, we went from C to… C with garbage collection[6][7]?

[6] Yes, yes, I know, Go has other nice features.

[7] And mandatory *tab* indentation.

Coda: What About Being Able To Create New Fields At Runtime?

```
class Foo:
    def __init__(self, bar, baz):
        self.bar = bar
        self.baz = baz

    def lets_add_buf(self, buf):
        self.buf = buf
```

Yeah, this is kind of weird. Python uses what's called the "metaobject protocol", which is a system for creating and modifying classes and objects on the fly. While you *can* do this, you really don't want to, it will cause you problems later on down the road.