

Object-Oriented Programming

What is OOP?

- So far, we've worked with built-in types: integers, strings, lists, etc.
- Some things are too complex to model using just these types
- OOP lets us define our own types

Example

- Suppose we have a list of geographic coordinates. For each data point, we want to store a city name, latitude, and longitude.

Example

- We could choose to store them as tuples, like this:

```
[  
  ("Los Angeles", 34.05, -118.25),  
  ("Denver", 39.73, -104.98),  
  ("Seattle", 47.61, -122.33),  
  ...  
]
```

Example

- This starts to get less tractable if we want to store more information. What if we want to add elevation, area, average annual temperature, county and state...

```
("Seattle", "King", "WA", 47.61,  
-122.33, 520, 71 ... )
```

Example

- OOP lets us do this:

```
>> seattle = City("Seattle")
```

```
>> seattle.county = "King"
```

```
>> seattle.state = "WA"
```

```
>> seattle.lat = 47.61
```

```
>> seattle.lon = -122.33
```

```
...
```

```
>> seattle.print_location()
```

```
Seattle is in King County, WA at  
47.61 N, 122.33 W.
```

Terminology

- Classes and objects:
 - A *class* is like a blueprint.
 - After defining a class, you can make *objects* that are instances of that class.
 - `seattle = City()` ← `City` is a class, `seattle` is an object that is an instance of the `City` class

Terminology

- Objects have *members* and *methods*:
 - Members are like variables.
 - Methods are like functions.

```
>> seattle.lat = 47.61
```

```
>> seattle.lon = -122.33
```

lat and lon are members, like variables

```
>> seattle.print_location()
```

print_location is a method, like a function

Another Example

- Another example of OOP that you're already familiar with:

```
cursor = connection.cursor()
```

...

- Connection is an object of type Connection; it has a method, "cursor" that returns an object of type Cursor.
- Now you can use the cursor methods to do things.

Constructor

- The first method defined should be called `__init__`
- This method is called whenever you make a new object from your class:

```
class City:
    def __init__(self, name):
        self.name = name

seattle = City("Seattle")
```

How to make a class

```
class (name of class):  
    def __init__(self, arg1, arg2 ...):  
        ...  
    def method1(self, arg1, arg2 ...):  
        ...  
    def method2(self, arg1, arg2 ...):  
        ...
```

- Use "self" to refer to the object within methods