LECTURE 5

OBJECT-ORIENTED PROGRAMMING

SUBCLASSES AND INHERITANCE

MCS 275 Spring 2022
Emily Dumas

LECTURE 5: SUBCLASSES AND INHERITANCE

Course bulletins:

- Worksheet 2 solutions available.
- Homework 2 due at Noon on Tuesday (25 Jan).

THE BIG NEWS

UIC is returning to in person instruction on Monday.

- Come to Lecture Center A002 for lectures
- Wear a well-fitting mask that fully covers your nose and mouth
- Install Acadly on a smartphone or tablet and bring to class

ZOOM REMAINS AVAILABLE

Come to lecture via zoom if:

- You're sick (even mild non-covid symptoms)
- You don't yet meet requirements to return to campus
- Your UIC daily pass is not green
- A travel disruption or other unpredictable event makes it difficult for you attend in person

But in the long term, joining via zoom needs to be an occasional workaround, not a habit.

IMPROVED POINT2 AND VECTOR2

I added new features to our `plane` module between lectures. Let's take a tour of the changes:

- Can multiply `Vector2` by integer or float
- `abs(Vector2)` gives length
- `Point2` and `Vector2` support equality testing

(There are other features we might want in a real-world application, but this will suffice for now.)

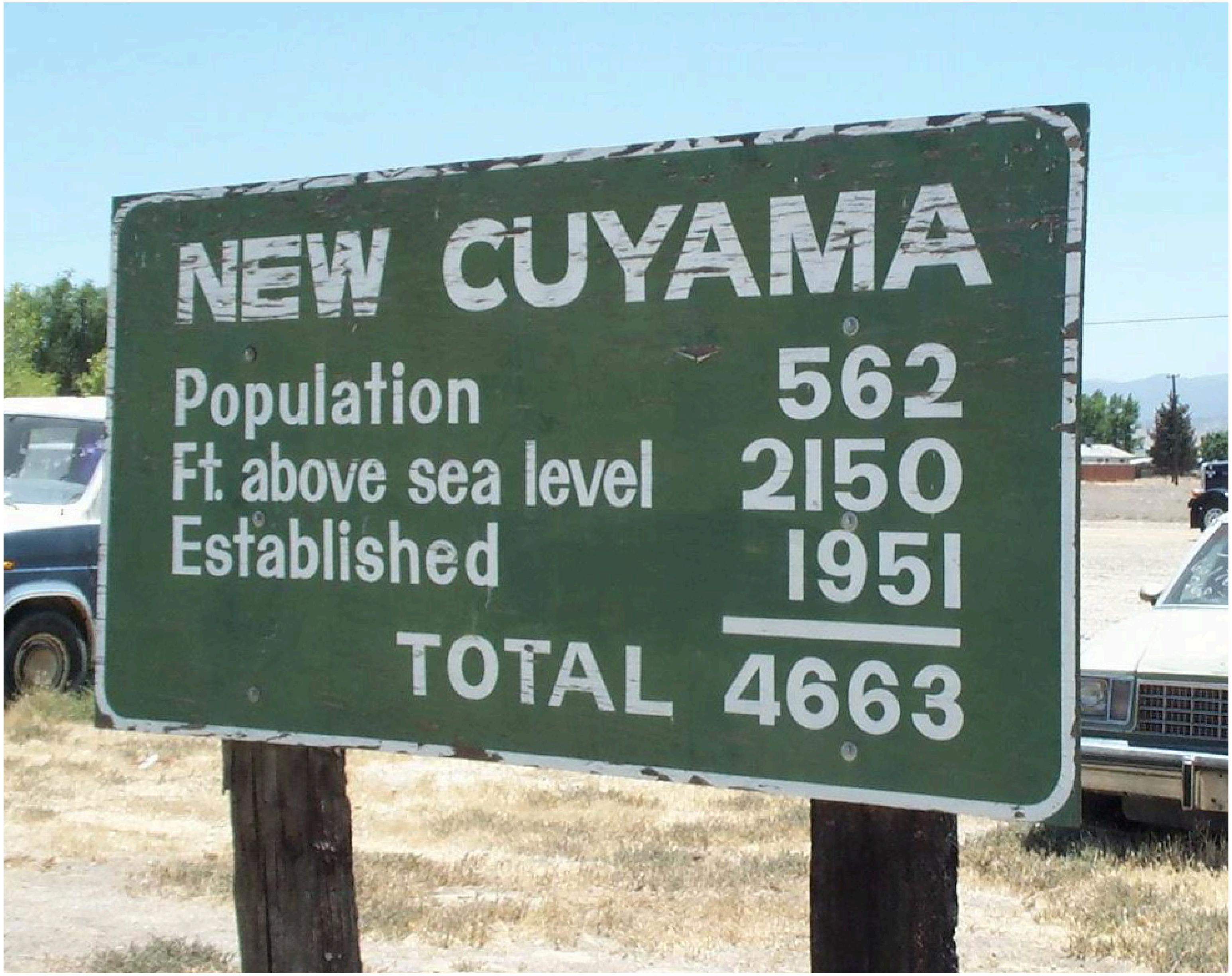


Photo by Mike Gogulski (CC-BY-SA)

INHERITANCE

It is possible to build a class that is derived from an existing one, so that the new class **inherits** all the methods and behavior of the existing class, but can add new features, too.

If new class `B` is derived from existing class `A` in this way, we say:

- `B` is a **subclass** of `A` (or child of `A` or inherits from `A`)
- `A` is a **superclass** of `B` (or parent of `B`)

WHY SUBCLASS?

Some common reasons:

- To add custom behavior to an existing class (e.g. a dict that only allows certain kinds of keys)
- To avoid code duplication when multiple classes share some behavior
- To formalize relationships between classes

Subclassing should express an "is-a" relationship. Dog and Cat might be subclasses of Pet.

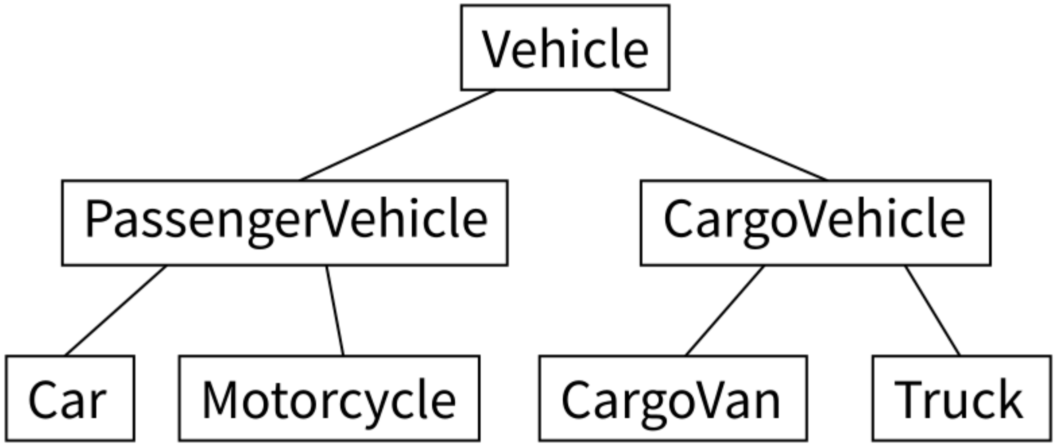
PYTHON SUBCLASS SYNTAX

Specify a class name to inherit from in the class definition:

```
class ClassName(SuperClassName):
    """Docstring of the subclass"""
    # ... subclass contents go here ...
```

There is a built-in class `object` that every class inherits from, even if you don't specify it explicitly.

CLASS HIERARCHIES



Inheritance patterns are often shown in diagrams. Lines represent inheritance, with the superclass appearing above the subclass (usually).

LIVE CODING

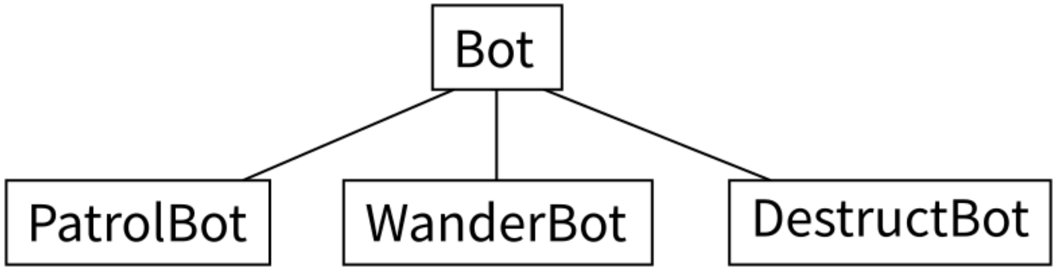
Let's build a class hierarchy for a simple robot simulation.

Every type of robot will be a subclass of `Bot`.

`Bot` has a `position` (a `Point`), boolean attribute `alive`, and method `update()` to advance one time step.

Subclasses give the robot behavior (e.g. movement).

PLANNED BOT HIERARCHY



- `PatrolBot` walks back and forth.
- `WanderBot` walks about randomly.
- `DestructBot` sits in one place for a while and then self-destructs.

ROBOT SIMULATION TEMPLATE

We haven't built any of the `Bot` subclasses yet, but I have already created:

- A barebones module `bots` containing a class `Bot`. This robot sits in one place. In `bots.py` in the sample code repository.
- A script `botsimulation.py` to run the simulation and show it with simple text-based graphics.

SUPER()

In methods of a subclass, `super()` returns a version of `self` that behaves like an instance of the superclass.

`super()` allows the subclass to call methods of the superclass even if the subclass overrides them.

FROM

The `from` keyword can be used to import individual symbols from a module into the global scope.

So

```
import mymodule
# ...
mymodule.useful_function()  # module name needed
```

is equivalent to

```
from mymodule import useful_function
# ...
useful_function()  # no module name needed
```

Please use `from` very sparingly!

REFERENCES

- I discussed inheritance in MCS 260 Fall 2021 Lecture 27
- See Lutz, Chapter 31 for more discussion of inheritance.
- Lutz, Chapters 26-32 discuss object-oriented programming.

REVISION HISTORY

- 2022-01-21 Initial publication