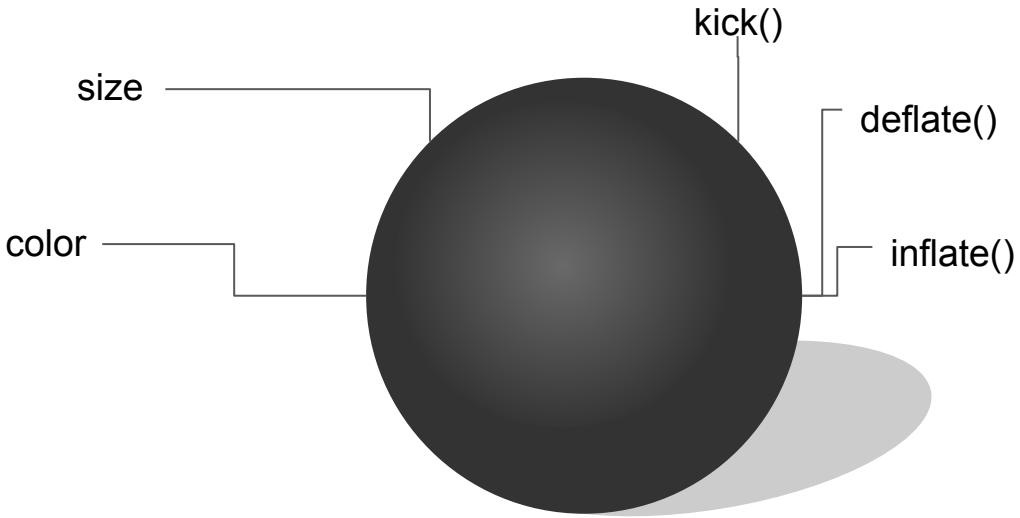
OOP in Python

An introduction

What is an object?

- *A thing you can interact with that has some properties*



- In Python, *everything is an object*

What's a class?

- Prescription of an object

```
class Vehicle: # Class names are PascalCase
    def __init__(self, speed=0, direction='N'):  # Special method __init__
        self.speed = speed
        self.direction = direction

    def change_speed(self, by): # Methods get extra parameter self
        self.speed += by

    def turn(self, direction):
        self.direction = direction
```

Things to note

- Functions of classes and objects are their *methods*
- Methods get passed *self*
  - Refers to the object on which the method is called
  - Always the first argument of function definition

```
>>> vehicle1.change_speed(5)
```

Is the same as

```
>>> Vehicle.change_speed(vehicle1, 5)
```

Special methods

- __init__
- __str__
- __repr__
- Operator overloading
  - Arithmetic (__add__, __sub__, __mul__, __mod__, …)
  - Accessor (__getitem__, __getattr__, __setattr__, …)

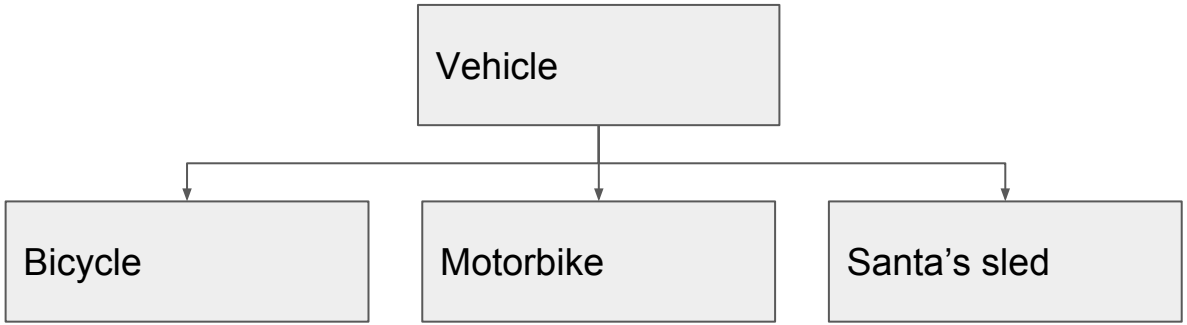
Playtime!

- Create two classes:
  - Vehicle - represents a road vehicle (has number of passengers)
  - Accident - represents a terrible accident (has a list of Vehicles involved)
- Make it so that Vehicle + Vehicle = Accident
- Make accident display number of people involved

Solution https://repl.it/IrbL/1

Inheritance

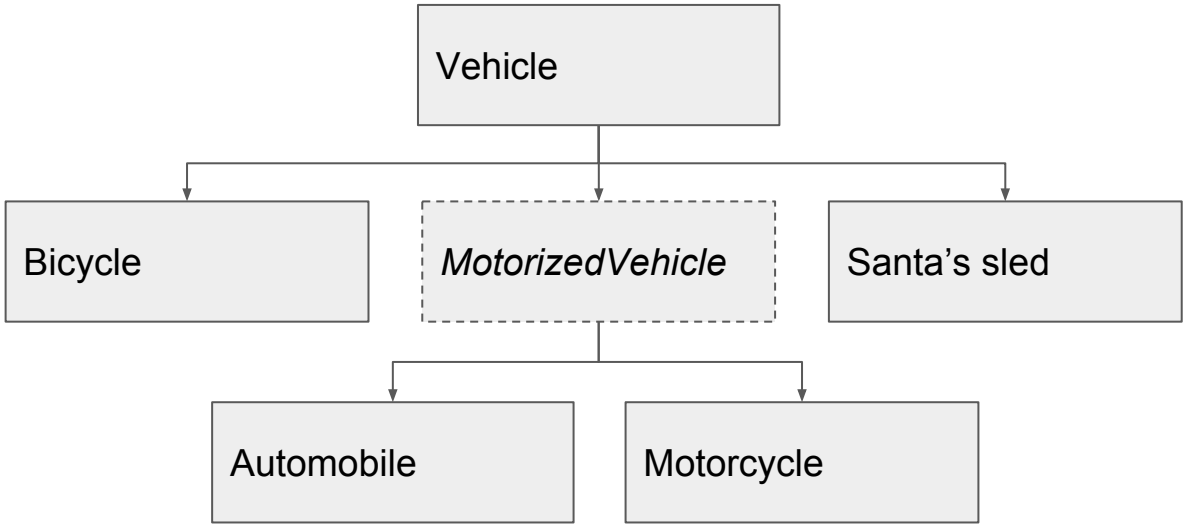
- For specialization, building structures



https://docs.python.org/3.1/tutorial/classes.html#inheritance

Inheritance

- For specialization, building structures



https://docs.python.org/3.1/tutorial/classes.html#inheritance