Programming fundamentals with Python

OOP - Inheritance

Pepe García

2020-04-20

Programming fundamentals with Python

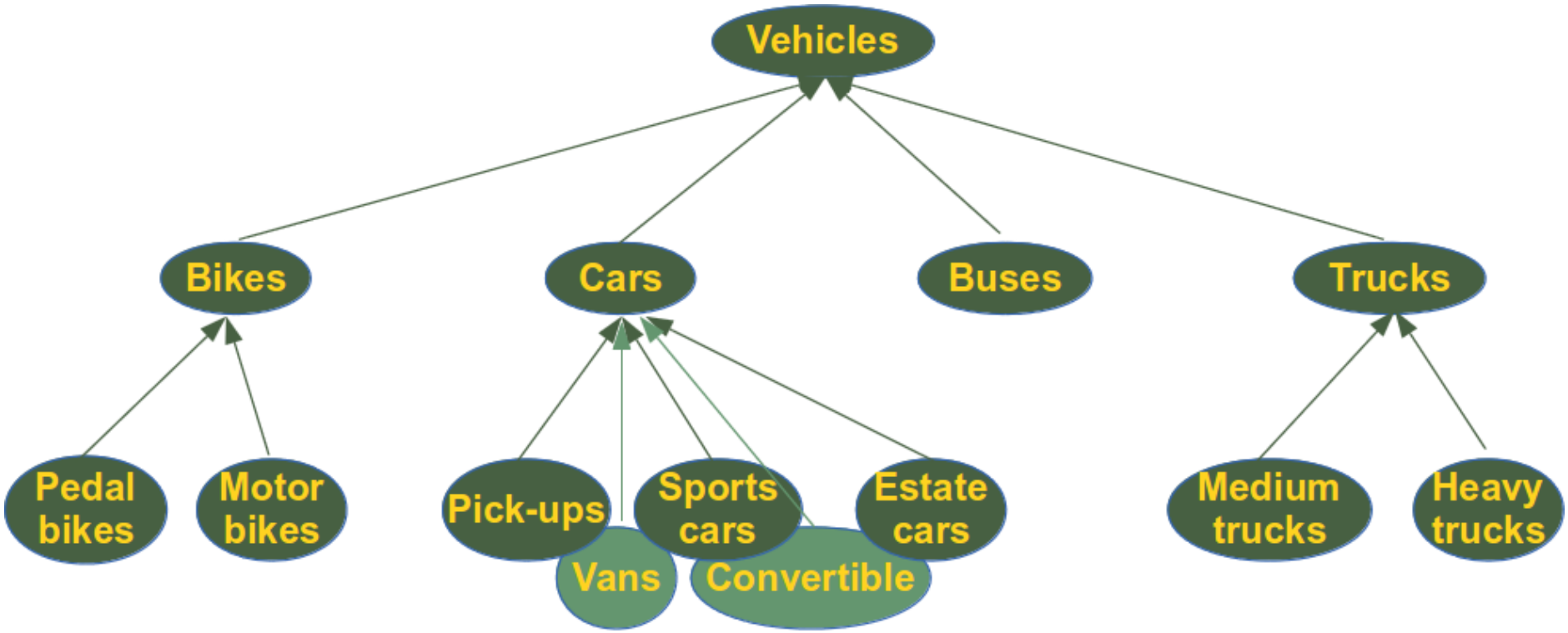
Plan for today

- OOP: Inheritance

Plan for today

- OOP: Inheritance
- Creating our own exceptions

Inheritance



Inheritance

Inheritance is a mechanism by which classes *extends* methods and attributes from other classes

Inheritance

classes and their subclasses form a **is a** relationship, and almost all things that have that relationship can be expressed using inheritance

Inheritance

How do we declare Inheritance?

```
class ClassName(ParentClass):
    pass
```

Inheritance

Example - polygons

Let's try to model some polygons using inheritance.

- Polygon
- Triangle
- Rectangle
- Square

Inheritance - constructor

Whenever we're using inheritance it's common that the parent's class constructor will take some arguments that we want to fulfill from the children classes. For solving that, Python provides the **`super()`** function.

Inheritance - constructor

Parent class

```
class Rectangle:
    def __init__(self, base, height):
        self.base = base
        self.height = height
```

Child class

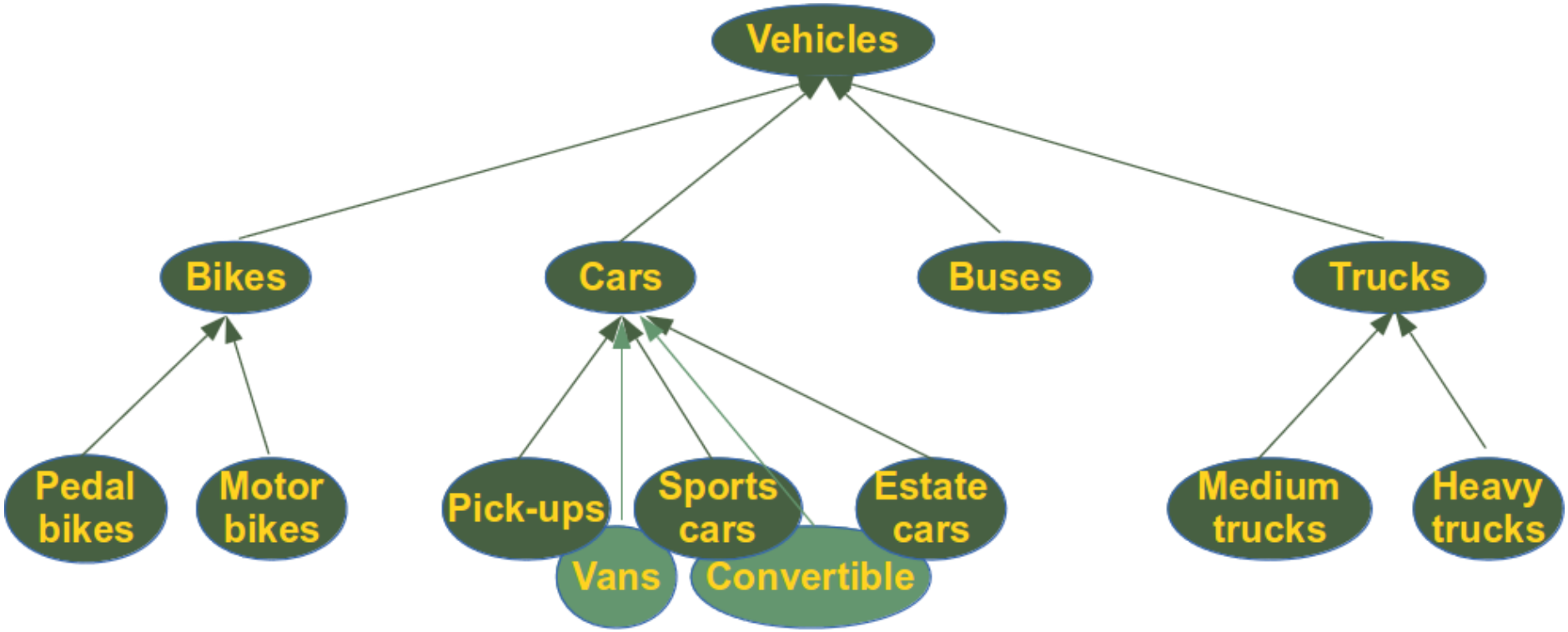
```
class Square(Rectangle):
    def __init__(self, side):
        super().__init__(side, side)
```

Method resolution

when calling a **method** in an object, python searches for the **method** in the object's class and, if not found, it goes up the class hierarchy

Method Resolution

In this example, trying to use the method **run** in a **SportsCar**, Python will first search it there, if it's not found, it will search it in **Car** class, and finally, in **Vehicle** otherwise.



Method resolution

Exercise

Create the following classes in Python. Implement the described methods in them.

Vehicle

- run()
- stop()

Car

- open_trunk()

SportsCar

- run_a_lot()

Creating exceptions

Creating our own exceptions is really simple, we just need to create a new class that **extends Exception**.

Creating exceptions

Exercise

Imagine you're doing the validation of a form. Create all the exceptions that come to your mind related to the validation of the fields

Exercise

Create a **Polygon** class, and two classes **Triangle** and **Circle** that extend from it.

Make **Triangle** and **Circle** have a **calculate_area** method that return their respective areas