Part II
Object-Oriented Programming
in Python

## Python and Objects

Everything in Python is an object (under the hood)

Every datatype (dict, String,..) is based on a Class that is hidden

Let's look at what you already know how to do in Python, thru an object lens

## All Python data types are objects

```
• my_dict  = {}    # where's the object?
```

name of the object instance of the Class **dict**

Python shorthand that constructs a dict object based on the Class **dict**

| dict |
|---|
| value:  None |
| All the methods defined in the class |

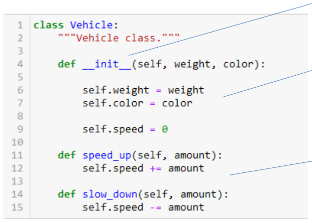
To execute an object's methods we use the dot notation.

**my_dict.clear()**

executes the clear method of the object my_dict

## Defining Your Own Classes

## Vehicle Class in Python



```
class Vehicle:
    """Vehicle class."""

    def __init__(self, weight, color):

        self.weight = weight
        self.color = color

        self.speed = 0

    def speed_up(self, amount):
        self.speed += amount

    def slow_down(self, amount):
        self.speed -= amount
```

**constructor** method used to initialize an object. All method definitions start with self – to distinguish them from regular functions

**data (attributes)** of our object defined with prefix **self**. Note that speed is by default set to 0

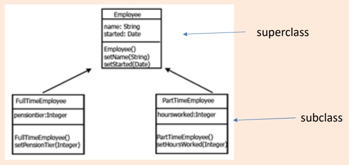
**methods** of our object defined with prefix **self**. Methods can modify attributes using prefix **self**.

## Review Account Class from Deitel Textbook

## Hacking Semi-Private Data

## Inheritance

see: https://www.programiz.com/python-programming/inheritance



Inheritance allows to define is-a relationships between classes

## Summary

- The reason for OOP is to reduce the complexity of software
- OOP provides
  - Encapsulation
  - Polymorphism
  - Inheritance
- Key concepts:
  - defining classes – using keyword self
  - public and private data
  - overriding methods in a superclass