6.0001 Recitation 4 - Spring 2020

Friday, February 28th 2020

I. Administrivia

Monday 3/2:
- Microquiz 3 during lecture
- Pset 2 checkoff due at 9pm

Wednesday 3/4:
- Pset 4 due at 9pm

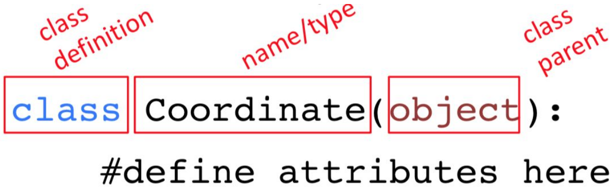
Next Monday 3/9:
- Pset 3 checkoff due at 9pm

II Object Orientation Programming (OOP)

- Objects allow you to store data in python
- Everything in python is an object
- Class defines a type of object
    - so far in class we have seen the following built-in classes: `int`, `float`, `string`, `list`, `tuples`, `dictionaries`
- Object is to a specific instance of a class
    - for example: 3, “hello”, [1,2,3] are all instances of a class
- Advantages of OOP:
    - Allows you to bundle data into packages
    - Reduces complexity of your code, making it easy to reuse code
    - Allows you to implement & test behaviour of each class separately

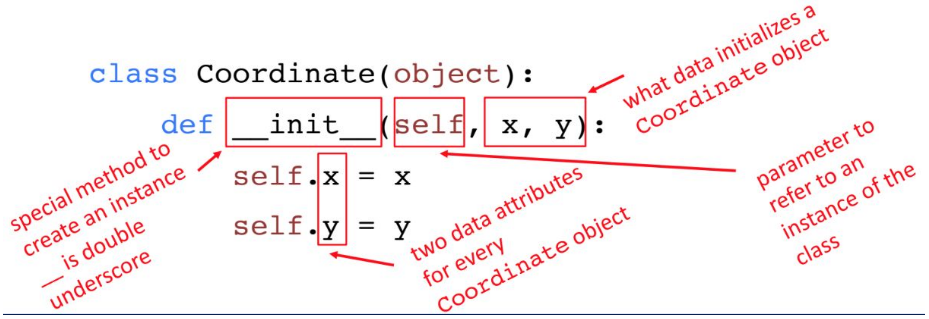
III. Classes

- a way to create your own data type using the built-in data types as building blocks
- attributes are data & procedures that belong to the class
    - Data attributes: objects that make up the class
    - Methods/procedural attributes: functions that only work with this class
- self:
    - Refers to the instance the method is called on
    - Always the 1st argument when defining a method
    - Not used outside the class definition
- Creating a class:
    1. Define class name



```
class Coordinate(object):
    #define attributes here
```

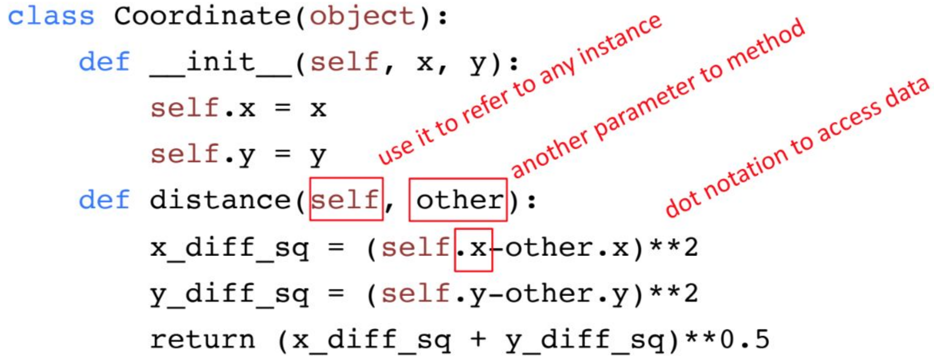
2. Define class attributes
    a. Define how to create an instance of a class using the `__init__` method



```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

what data initializes a Coordinate object
special method to create an instance
__ is double underscore
parameter to refer to an instance of the class
two data attributes for every Coordinate object

    b. Define other methods, these do not need to start with __



```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, other):
        x_diff_sq = (self.x-other.x)**2
        y_diff_sq = (self.y-other.y)**2
        return (x_diff_sq + y_diff_sq)**0.5
```

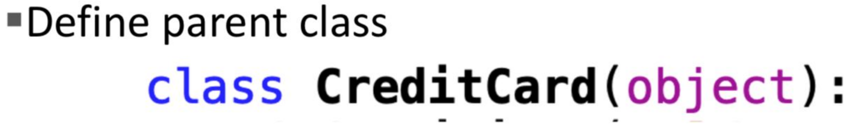
use it to refer to any instance
another parameter to method
dot notation to access data

- In order to be able to call `print` on an instance of your class you need to define the __str__ function.
    - there are a few more special operators (look at lecture slides for details on these)
- You can use `isinstance()` to check if an instance is an object of a class.
- In general the class defines the representation and methods common across all instances of the class; whereas an object is a SPECIFIC instance of the class
- In general you want to keep your internal representation of your class hidden, to prevent adversarial attacks.
    - the internal representation refers to what is in your `__init__` method
- class variables: shared across all members of a class
    - defined outside of the `__init__` method

IV. Inheritance

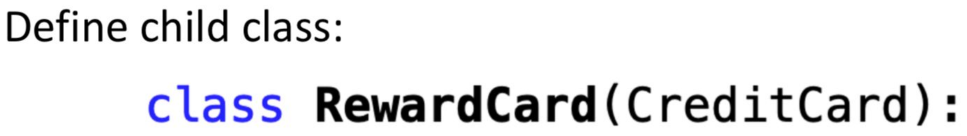
- Allows you to extend a class with new/different capability

- You create an inheritance relationship between two classes by defining what goes into the parameter in the class definition.



▪Define parent class

```
class CreditCard(object):
```



Define child class:

```
class RewardCard(CreditCard):
```

- The child class inherits all of the methods from the parent class.
- We can define new methods in the child class to **extend** behavior
- We can redefine methods defined in the parent class to **modify** behavior
- When you call a method on an instance of a class: the interpreter tries to find the method at the level of a class and then checks the parent.
    - This means that if two methods with the same name are defined, the method in the child class takes precedence.