Python, Day 6.5: Classes and Objects

Andrew Bydlon

January 13, 2019

Procedural vs Object Oriented programming

So far, we have been programming **procedurally**; we tell the computer what to do in several steps. This is how things are always done in small code.

However, when code becomes larger, it is better to manage our data in larger chunks, called **objects**. Every type of data in python is an object; strings, lists, numbers, etc.

Previously, our programs manipulated data. Objects can contain data (or **attributes**) and functions (or **methods**) all in one package.

Classes

First, we must develop the idea of a **class**. Classes essentially define a new data type (blueprint) for python to interpret.

We can then define objects of this type to actually hold our data/functions.

Syntax (Classes)

```
class ClassName:   # adjoining (ParentClass) adds a parent class
    method1
    method2
    etc
```

We will discuss parent classes more next class.

Explaining with example

Example (Bank Account)

```
class BankCustomer:
# Define the properties of a customer
    def __init __ (self, name, balance=0.0)
        self.name = name
        self.balance = balance
# Allow the withdrawal function to be applied to customers
    def withdrawal(self, amount):
        if amount > self.balance:
            raise RuntimeError('Amount greater than available balance.')
        self.balance -= amount
        return self.balance
# Allow the deposit function to be applied to customers
    def deposit(self, amount):
        self.balance += amount
        return self.balance
```

Special components of a class

- __init__: This function declares how an object of type MyClass (i.e. BankCustomer) is defined. In the case above, we need to assign both a name and a balance to define someone's bank account.
- self: Representative of the object name later. For example, if we make Rob an object of classification BankCustomer, we will define Rob.name to be Rob, and Rob.balance to be Rob's balance at the moment of opening the account.

  NOTE: Python assumes that once init is satisfied, the data contains everything it needs to *minimally* satisfy the class.

Example (Possible Error)

```
class BankCustomer:
    def __init __ (self, name)
        self.name = name
    def balance (self, balance)
        self.balance = balance
    def withdrawal(self, amount):
        if amount > self.balance:
            raise RuntimeError('Amount greater than available balance.')
        self.balance -= amount
        return self.balance
    def deposit(self, amount):
        self.balance += amount
        return self.balance
```

What's wrong?

If someone called withdrawal/deposit before calling balance, they would get an error!

Creating objects in your class paradigm

Classes should be thought of as a blueprint. They contain no data themselves, but only a framework for how an **object** of type MyClass should be defined.

Objects are the data holding and function running items of a given class type. They are defined as follows:

Syntax (Objects)

MyObject = MyClass(InitVar1, InitVar2, . . . , InitVarN)

Note: You need to assign each of the init variables when declaring the variable.

Continuing with BankCustomer

Example (Making our first object: Rob!)

```
> > > Rob = BankCustomer("Robert",10.0)
> > > # I've assigned the name Robert, and balance 10.0
> > > Rob.balance
10.0
> > > Rob.withdrawal(11)
Traceback (most recent call last):
RuntimeError: Amount greater than available balance.
> > > Rob.withdrawal(9)
1.0
> > > Rob.deposit(100000000) # Robert won the lottery!
100000001.0
```

Accessing objects

As in the previous slide, you can access the functions of an object by typing the following:

Syntax

MyObject.MyClassMethod(vars)

We can also access the attributes in a similar way:

Syntax

MyObject.MyAttribute

MyObject.MyAttribute = SomeNewValue

Passing your objects to functions

You can pass an entire object to a function! This is advantageous especially since you could have **100 attributes** and **100 methods** on a given class, and pass it all with a single entry!

Example (Continuation with BankCustomer)

```
Rob = BankCustomer("Robert Langland", 10000.0)
Karen = BankCustomer("Karen Smith", 20000)
Kendrick= BankCustomer("Kendrick Lamar", 10E10)

def ApplyInterest(BankGoer):
    BankGoer.balance = BankGoer.balance * 1.03

BankMembers = [Rob, Karen, Kendrick]

for i in BankMembers:
    ApplyInterest(i)
```

Assignment 11

Create a class for an ellipse. It should have the following two init-class properties:

1. Major axis (MA)
2. Minor axis (ma)

It should also have a function to compute the focus length, area, & approximation for the perimeter of such an object. Here are some formulae:

$$FL = \sqrt{MA^2 - ma^2}$$

$$Area = \pi \cdot MA \cdot ma$$

$$Perimeter \approx 2 \cdot \pi \cdot \sqrt{\frac{MA^2 + ma^2}{2}}$$

Ask the user for the MA and ma, and display out this info. Upload when you finish!