Programación orientada a objetos en Python

Clases

Las clases proveen una forma de empaquetar datos y funcionalidad juntos. Al crear una nueva clase, se crea un nuevo *tipo* de objeto, permitiendo crear nuevas *instancias* de ese tipo. Cada instancia de clase puede tener atributos adjuntos para mantener su estado. Las instancias de clase también pueden tener métodos (definidos por su clase) para modificar su estado.

Comparado con otros lenguajes de programación, el mecanismo de clases de Python agrega clases con un mínimo de nuevas sintaxis y semánticas. Es una mezcla de los mecanismos de clases encontrados en C++ y Modula-3. Las clases de Python proveen todas las características normales de la Programación Orientada a Objetos: el mecanismo de la herencia de clases permite múltiples clases base, una clase derivada puede sobre escribir cualquier método de su(s) clase(s) base, y un método puede llamar al método de la clase base con el mismo nombre. Los objetos pueden tener una cantidad arbitraria de datos de cualquier tipo. Igual que con los módulos, las clases participan de la naturaleza dinámica de Python: se crean en tiempo de ejecución, y pueden modificarse luego de la creación.

```
class MiClase:
    """Simple clase de ejemplo"""
    i = 12345
    def f(self):
        return 'hola mundo'
```

Instancias

La instanciación de clases usa la notación de funciones. Hacé de cuenta que el objeto de clase es una función sin parámetros que devuelve una nueva instancia de la clase. Por ejemplo (para la clase de más arriba):

```
x = MiClase()
```

La operación de instanciación (“llamar” a un objeto clase) crea un objeto vacío. Muchas clases necesitan crear objetos con instancias en un estado inicial particular. Por lo tanto una clase puede definir un método especial llamado **__init__()**, de esta forma:

```
def __init__(self):
```

```
    self.datos = []
```

Por supuesto, el método **__init__()** puede tener argumentos para mayor flexibilidad. En ese caso, los argumentos que se pasen al operador de instanciación de la clase van a parar al método **__init__()**. Por ejemplo,

```
>>> class Complejo:
...     def __init__(self, partereal, parteimaginaria):
...         self.r = partereal
...         self.i = parteimaginaria
...
>>> x = Complejo(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

Bibliografía

http://docs.python.org.ar/tutorial/3/classes.html