Clases y objetos

Clases

La definición mas sencilla tiene la forma:

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Atributos y métodos de clase

```
class MyClass:
    """A simple example class"""
    i = 12345
    def f(self):
        return 'hello world'
```

MyClass.i

MyClass.f

MyClass.__doc__

X=new MyClass()

Crea una instancia de la clase MyClass y la guarda en la variable X del tipo objeto

Inicialización de objetos

Constructor de objetos en python

def __init__(self):

self.data = []

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

Atributos

Los atributos, al igual que las variables, no necesitan ser declarados y simplemente son creados al ser utilizados

```
x = new MyClass()
x.counter = 1

while x.counter < 10:
    x.counter = x.counter * 2
    print(x.counter)
```

Métodos

```
def Persona():
    def __init__(self, nombre=None):
        self.nombre = nombre
    def saludo(self):
        If self.nombre:
            print('Hola, me llamo {}'.format(self.nombre))
            tu_nombre = raw_input('Cómo te llamas? ')
            print('Hola,', tu_nombre)

x = new Persona()
x.saludo()
```

Herencia

Una clase se define como una especialización de otra:

```
class DerivedClassName(BaseClassName):
    <statement-1>
    .
    .
    .
    <statement-N>
```

```
class DerivedClassName(Base1, Base2, Base3):
    <statement-1>
    .
    .
    .
    <statement-N>
```

Funciones y atributos de clase

- **objeto.__class__**
- **isinstance()**

  isinstance(obj, int) es verdadero si obj.__class__ es int o una clase derivada de int.

- **issubclass()**
  - issubclass(bool, int) es verdadero porque bool es una subclase de int.