UF3.8 Clases en Python

Centro Profesional
Universidad Europea de Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

Contenidos

- Clases en Python
- Sentencias class
- Propiedades de una clase
- Métodos de una clase
- Objetos en Python
- Accediendo a los métodos y propiedades de un objeto
- Constructores en Python
- Tipo de un objeto
- Herencia
- Mostrar objetos

Private & Confidential

Clases en Python

Las clases son los modelos sobre los cuáles se construirán objetos. En Python, una clase se define con la instrucción **class** seguida de un nombre genérico para el objeto.

```
class Objeto:
        pass
```

PEP 8: clases

El nombre de las clases se define en singular, utilizando CamenCase.

Sentencia pass

La sentencia **pass** no hace nada. Se puede usar cuando una sentencia es requerida por la sintaxis pero el programa no requiere ninguna acción. Se usa normalmente para crear clases en su mínima expresión.

```
class Objeto:
        pass
```

Private & Confidential

Propiedades de una clase

Las propiedades son las características intrínsecas del objeto. Éstas, se representan a modo de variables, solo que técnicamente, pasan a denominarse *propiedades*:

```
class Ojo():
    forma = ""
    color = ""
    tamanio = ""
```

PEP 8: propiedades

Las propiedades se definen de la misma forma que las variables (aplican las mismas reglas de estilo).

Private & Confidential

Métodos de una clase

Los métodos son *funciones* solo que técnicamente se denominan métodos, y representan acciones propias que puede realizar el objeto (y no otro):

```
class Objeto():
    color = "verde"
    tamanio = "grande"
    aspecto = "feo"
    ojos = Ojo()

    def flotar(self):
        pass
```

NOTA: **El primer parámetro de un método, siempre debe ser self**

Private & Confidential

Objetos en Python

Las clases por sí mismas, no son más que modelos que nos servirán para crear objetos en concreto. Podemos decir que una clase, es el razonamiento abstracto de un objeto, mientras que el objeto, es su materialización. A la acción de crear objetos, se la denomina *instanciar una clase* y dicha instancia, consiste en asignar la clase, como valor a una variable:

```
et = Objeto()
print et.color
print et.tamanio
print et.aspecto
```

Private & Confidential

Métodos y propiedades de un objeto

Una vez creado un objeto, es decir, una vez hecha la instancia de clase, es posible acceder a su métodos y propiedades. Para ello, Python utiliza una sintaxis muy simple: el nombre del objeto, seguido de punto y la propiedad o método al cuál se desea acceder:

```
objeto = MiClase()

print objeto.propiedad

objeto.otra_propiedad = "Nuevo valor"

variable = objeto.metodo()

print variable
```

Private & Confidential

Constructores en Python

En Python para crear un constructor se utiliza un método especial __init__.

```
class Punto(object):
    """ Representación de un punto en el plano, los atributos son x
e y que representan los valores de las coordenadas cartesianas."""
    def __init__(self, x=0, y=0):
        # Constructor de Punto, x e y deben ser numéricos
        self.x = x
        self.y = y
```

Este método se llama cada vez que se crea una nueva instancia de la clase.

Un constructor no puede retornar ningún valor.

Tipo de un objeto

Para saber de qué tipo es un objeto (o una variable), utilizamos la función type, y para saber qué métodos y atributos tiene ese objeto (o variable) utilizamos la función dir.

```
a = open("archivo.txt")
type(a) # 'file'
dir(a)
'''['__class__', '__delattr__', '__doc__', '__enter__', __exit__',
'__getattribute__', '__hash__', '__init__', '__iter__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__str__',
'close', 'closed', 'encoding', 'fileno', 'flush', 'isatty', 'mode', 'name',
'newlines', 'next', 'read', 'readinto', 'readline', 'readlines', 'seek',
'softspace', 'tell', 'truncate', 'write', 'writelines', 'xreadlines']'''
```

Private & Confidential

Herencia

Algunos objetos comparten las mismas propiedades y métodos que otro objeto, y además agregan nuevas propiedades y métodos. A esto se lo denomina herencia: una clase que hereda de otra. Vale aclarar, que en Python, **cuando una clase no hereda de ninguna otra, debe hacerse heredar de object**, que es la clase principal de Python, que define un objeto.

```
class Ojo(object):
    forma = ""
    color = ""
    tamanio = "“


class ClaseDerivada(ClaseBase):
```

Herencia

__**class**__ un método especial de Python que sirve para saber la clase a la que instancia un objeto.

Todo variable es un objeto, y por lo tanto tiene una clase (también llamado su tipo). Ésta se almacena como objeto.__class__.

Python tiene dos funciones integradas que funcionan con herencia:

- isinstance() para verificar el tipo de una instancia.

Ej: isinstance(obj, int) devuelve True solo si obj.__class__ es int o alguna clase derivada de int.

- issubclass() para comprobar herencia de clase.

Ej: issubclass(bool, int) da True ya que bool es una subclase de int.

Private & Confidential

Mostrar objetos

Para mostrar objetos, Python indica que hay que agregarle a la clase un método especial, llamado __**str**__ que debe devolver una cadena de caracteres con lo que queremos mostrar. Ese método se invoca cada vez que se llama a la función str.

El método __str__ tiene un solo parámetro, self.

```
def __str__(self):
    """ Muestra el punto como un par ordenado. """
    return "(" + str(self.x) + ", " + str(self.y) + ")"
```

Private & Confidential

Bibliografía y Webgrafía

http://docs.python.org.ar/tutorial/2/classes.html

http://librosweb.es/libro/algoritmos_python/capitulo_14/tipos.html