datarebel

Bienvenido

Python Fundamentals

datarebel

Python Fundamentals

Programación orientada a objetos

datarebel

Objetivos del Módulo:

- El estudiante sabrá cuál es el paradigma de programación orientado a objetos.
- El estudiante conocerá los principios básicos de OOP.
- El estudiante podrá identificar los costos / beneficios de este paradigma, para determinar cuál de los dos parámetros es más funcional para el proyecto.

OOP

Modulo	Temas	Subtemas	Tipo de conocimiento	Duración (Horas)
OOP				4
	Diferencias entre el paradigma de programación estructurado y el orientado a objetos		Teoría	
	POO	POO en Python	Mixto	
		Clases	Mixto	
		Objetos	Mixto	
		Atributos	Mixto	
		Metodos	Mixto	
		Instancias	Mixto	
	Herencia	Clase Base	Mixto	
		Herencia Simple	Mixto	
		Herencia Multiple	Mixto	

datarebel

TEMA 7A: ¿QUE ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

Python Fundamentals: Sesión 5

Differences between structured and object-oriented language

- La OOP tiene sus raíces en la década del 60 con el lenguaje de programación Simula que en 1967, el cual fue el primer lenguaje que posee las características principales de un lenguaje orientado a objetos.
- Smalltalk (de 1972 a 1980) es posiblemente el ejemplo canónico, y con el que gran parte de la teoría de la OOP se ha desarrollado. Más su uso se popularizó a principios de la década de 1990.
- En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.
- Los objetivos de la OOP son:
    - Organizar el código fuente, y
    - Re-usar código fuente en similares contextos.

Differences between structured and object-oriented language

- **Programación orientada a objetos** es un estilo que trata los datos como objetos con atributos y métodos que pueden aplicarse a estos objetos y también ser heredados por otros objetos. Java es un gran ejemplo de un lenguaje que emplea este concepto. Pero Java es un lenguaje multi-paradigma y también utiliza algunos conceptos familiares para la Programación Procedimental.

- **La programación estructurada**, por otro lado, es un tipo de programación imperativa, donde las declaraciones se ponen en procedimientos, que se pueden volver a llamar cuando sea necesario. C usa programación procedimental.

Object-oriented programming

- Algunas particularidades de OOP en Python son las siguientes:
    - Todo es un objeto, incluyendo los tipos y clases.
    - Permite herencia múltiple.
    - No existen métodos ni atributos privados.
    - Los atributos pueden ser modificados directamente.
    - Permite «monkey patching».
    - Permite «duck typing».
    - Permite la sobrecarga de operadores.
    - Permite la creación de nuevos tipos de datos.

datarebel

datarebel

TEMA 7B: CLASES

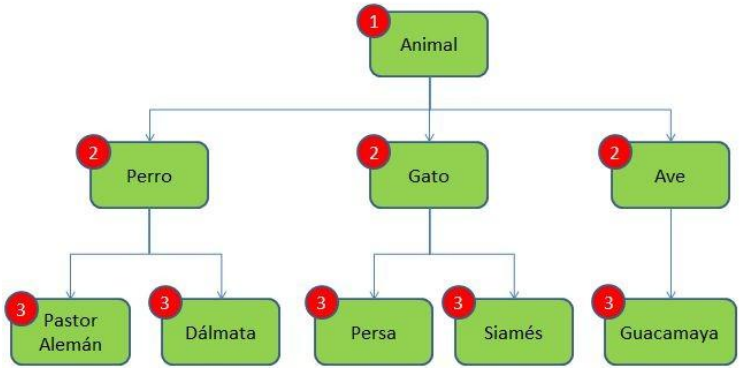
Python Fundamentals: Sesión 5

Clases

- Una clase es la descripción de un conjunto de objetos similares; consta de métodos y de datos que resumen las características comunes de dicho conjunto

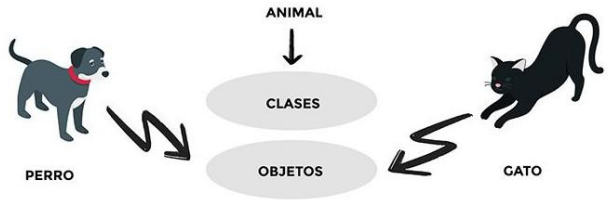
- Se pueden definir muchos objetos de la misma clase de la misma forma que, en la vida real, haríamos galletas (objeto) con el mismo molde (clase) solo que, para entenderlo mejor, cada galleta tendría igual forma pero es posible que tenga distinto sabor, textura, olor, color, etc.

```
class Persona:
    pass
```



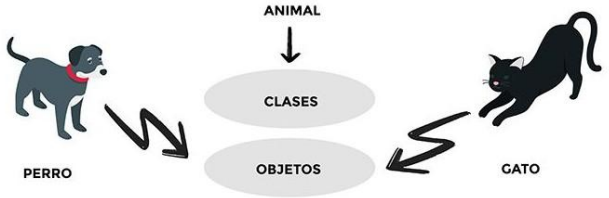
Objetos

- Los objetos son abstracción de Python para data. Toda la data en un programa Python es representado por objetos o por relaciones entre objetos.
- Cada objeto tiene una identidad, un tipo y un valor. Una identidad de objeto nunca cambia una vez es creada; usted puede pensar eso como la dirección de objeto en memoria.
- El tipo de un objeto también es inmutable. El tipo de un objeto determina las operaciones que admite el objeto y también define los valores posibles para los objetos de ese tipo.
- El conjunto de datos y objetos relacionados con un objeto en un momento dado, se le conoce como «estado». Un objeto puede tener múltiples estados a lo largo de su existencia conforme se relaciona con su entorno y otros objetos.



Objetos

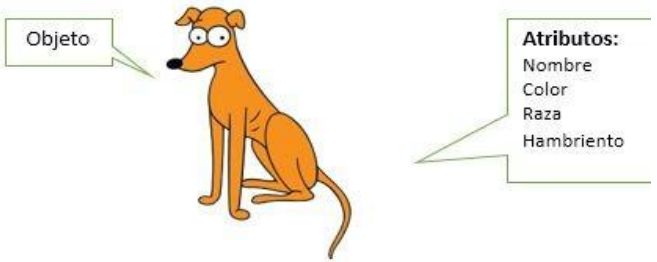
- Los objetos son abstracción de Python para data. Toda la data en un programa Python es representado por objetos o por relaciones entre objetos.
- Cada objeto tiene una identidad, un tipo y un valor. Una identidad de objeto nunca cambia una vez es creada; usted puede pensar eso como la dirección de objeto en memoria.
- El tipo de un objeto también es inmutable. El tipo de un objeto determina las operaciones que admite el objeto y también define los valores posibles para los objetos de ese tipo.
- El conjunto de datos y objetos relacionados con un objeto en un momento dado, se le conoce como «estado». Un objeto puede tener múltiples estados a lo largo de su existencia conforme se relaciona con su entorno y otros objetos.



Objetos

- Los atributos o propiedades de los objetos son las características que puede tener un objeto, como el color. Si el objeto es Persona, los atributos podrían ser: cedula, nombre, apellido, sexo, etc...

- Los atributos describen el estado de un objeto. Pueden ser de cualquier tipo de dato.



Objetos

```
class Persona:
    """Clase que representa una Persona"""
    cedula = "ROVA950203"
    nombre = "Africa"
    apellido = "Rodriguez"
    sexo = "F"
```

```
sahara= Persona
```

```
dir(sahara)
```

```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'apellido',
 'cedula',
 'nombre',
 'sexo']
```

```
sahara.cedula
```

```
'ROVA950203'
```

```
sahara.nombre
```

```
'Africa'
```

```
sahara.apellido
```

```
'Rodriguez'
```

```
sahara.sexo
```

```
'F'
```

```
print ("El objeto de la clase " + sahara.__name__ +"," + sahara.__doc__ + ".")
```

```
El objeto de la clase Persona,Clase que representa una Persona.
```

```
sahara.__name__
```

```
'Persona'
```

```
sahara.__doc__
```

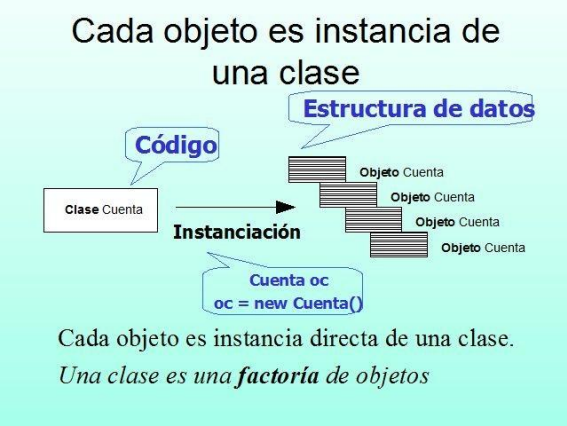
```
'Clase que representa una Persona'
```

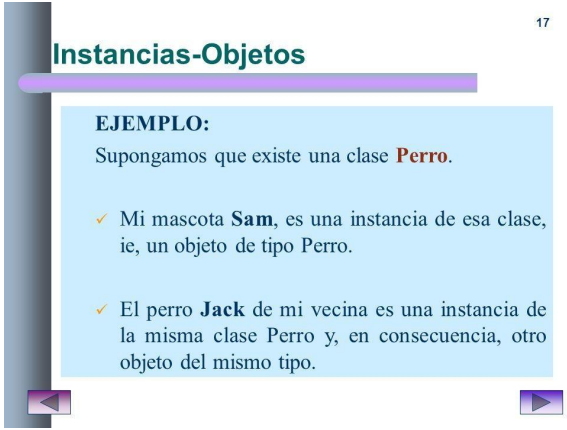
Métodos

- Los métodos describen el comportamiento de los objetos de una clase. Estos representan las operaciones que se pueden realizar con los objetos de la clase,
- La ejecución de un método puede conducir a cambiar el estado del objeto.
- Se definen de la misma forma que las funciones normales pero deben declararse dentro de la clase y su primer argumento siempre referencia a la instancia que la llama, de esta forma se afirma que los métodos son funciones, adjuntado a objetos.

Instancias

- Una instancia es una copia específica de la clase con todo su contenido.





Métodos

```
class Persona:
    """Clase que representa una Persona"""
    cedula = "ROVA950203"
    nombre = "Africa"
    apellido = "Rodriguez"
    sexo = "F"
    def hablar(self, mensaje):
        """Mostrar mensaje de saludo de Persona"""
        return mensaje
```

```
sahara=Persona
```

```
Persona().hablar("Hola, soy la clase {0}.".format(sahara.__name__))
```

```
'Hola, soy la clase Persona.'
```

```
type(Persona().hablar)
```

```
method
```

```
Persona().hablar.__doc__
```

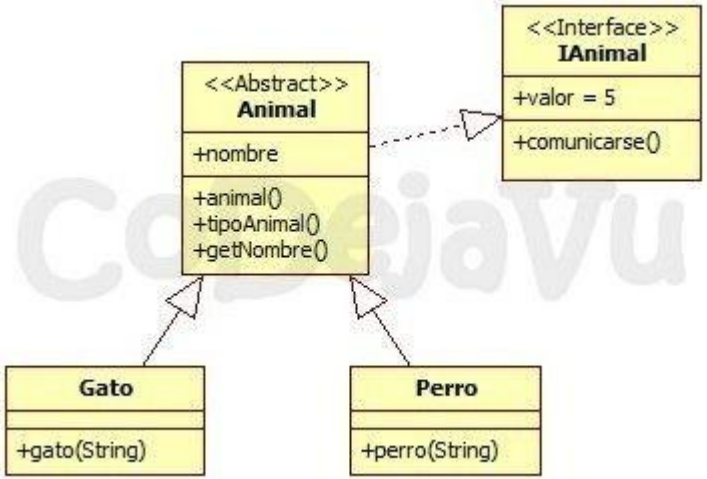
```
'Mostrar mensaje de saludo de Persona'
```

```
sahara.hablar
```

```
<function __main__.Persona.hablar(self, mensaje)>
```

Interface

- La forma en que los métodos de un objeto pueden ser accedidos por otros objetos se conoce como «interfaz». Una interfaz bien definida permite a objetos de distinta índole interactuar entre sí de forma modular. La interfaz define el modo en que los objetos intercambian información.



datarebel

¡Gracias!

Python Fundamentals