Python Básico

Programación Orientada a Objetos

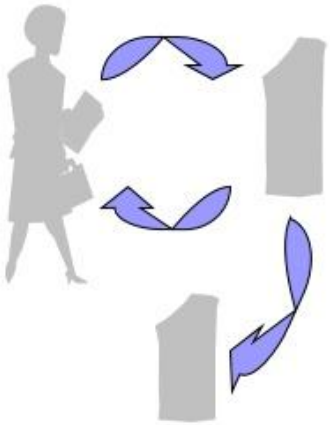
Octubre de 2017

PROTECO

Introducción

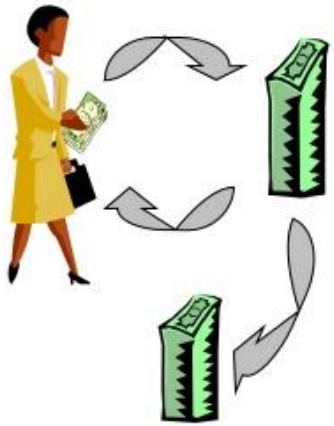
La programación orientada a objetos es un paradigma de programación basado en el concepto de "objetos", donde éstos interactúan entre sí para resolver un problema computacional.

PROCEDIMENTAL



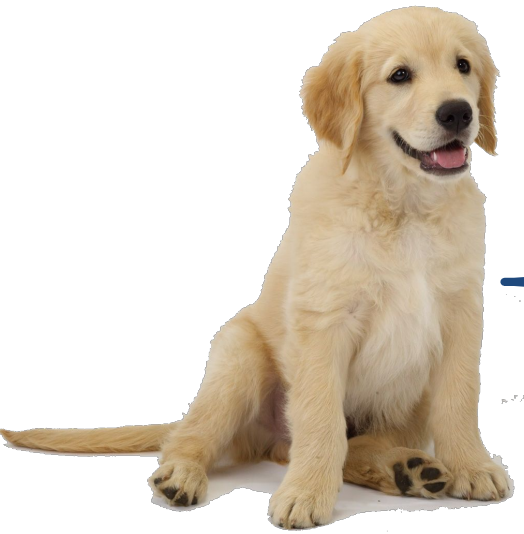
Retirar, depositar, transferir

OBJETOS



Cliente, dinero, cuenta

Objeto



Atributos:

- color
- raza
- edad
- energía

Métodos:

- jugar
- dormir
- comer

Un objeto es una entidad que tiene propiedades particulares, llamadas atributos, y formas de operar sobre sus propios atributos y los de otros objetos, llamadas métodos.

Clase

Una clase es un modelo definido por el usuario a partir del cual se crean objetos. En ellas se define cuáles serán los métodos y atributos que definirán a los objetos de su tipo.

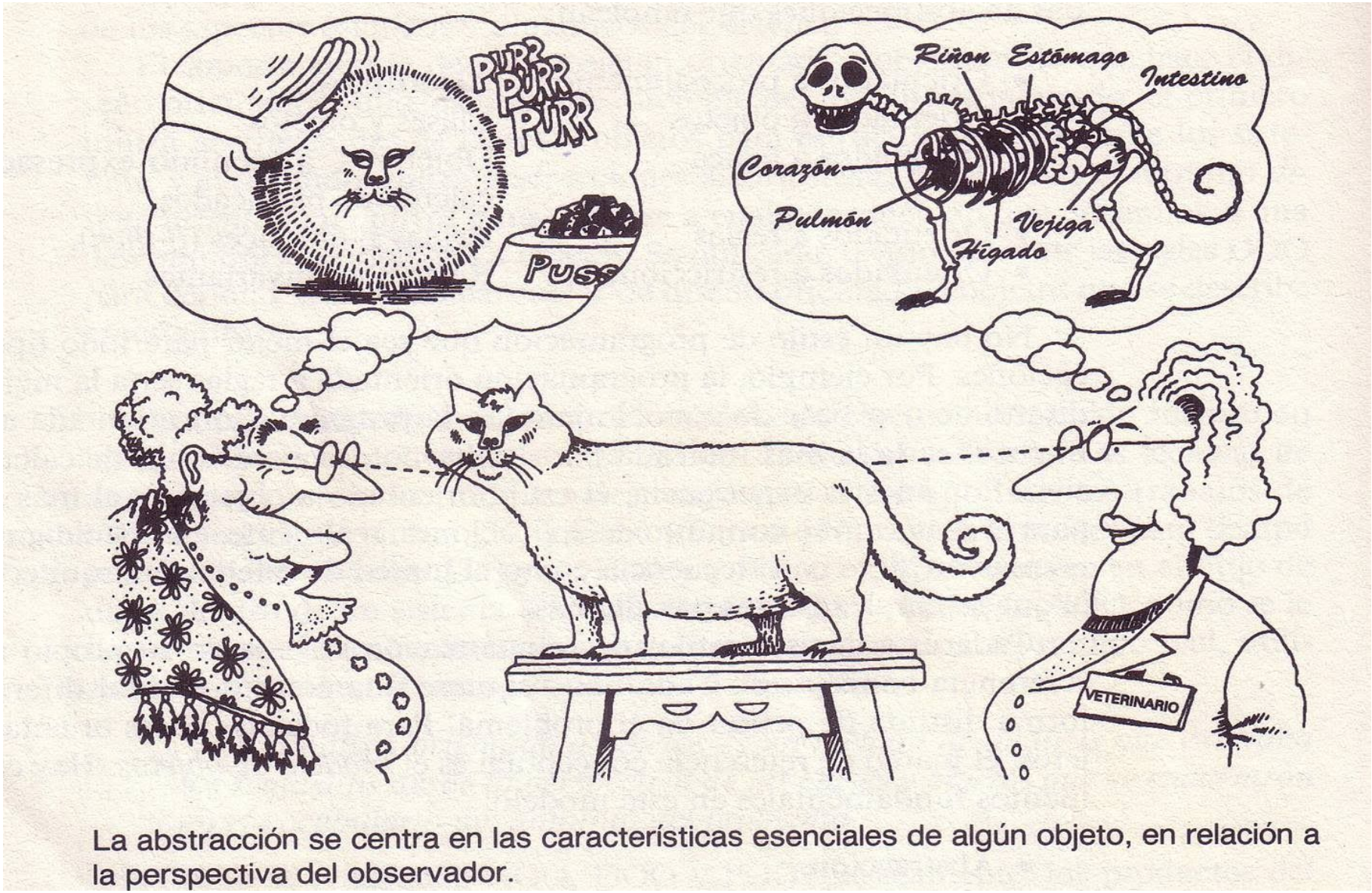
Cuando creamos un objeto, se dice que se "instancia" una clase. En ese momento, asignamos valores a los atributos de una clase para crear objetos diferentes.

Clase

Los 4 pilares de la POO

Abstracción

Consiste en aislar un elemento de su contexto como un individuo, y preguntarnos qué es lo que lo define.
En programación, el término se refiere al énfasis en el “¿qué tiene” y "¿qué hace?", más que en el "¿cómo lo hace?".



La abstracción se centra en las características esenciales de algún objeto, en relación a la perspectiva del observador.

Abstracción

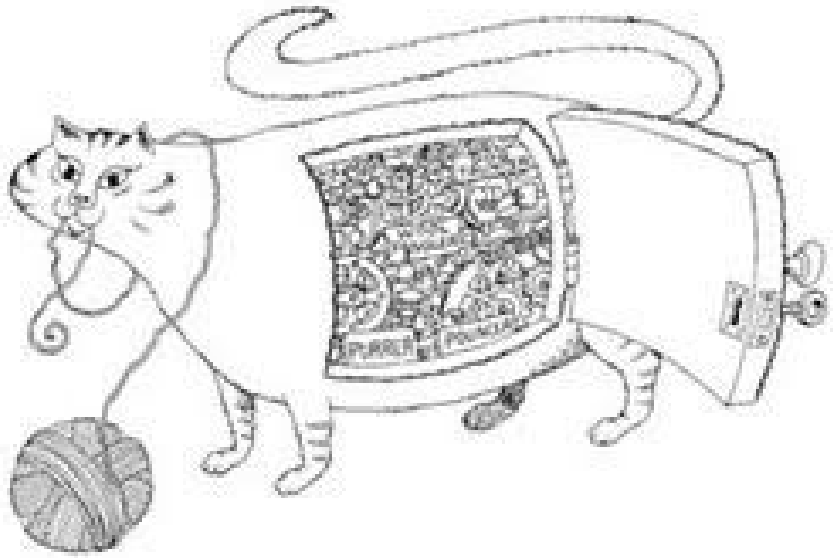
Sintaxis:

```
class NombreClase:
        def __init__ (self, at1, atn):
                self.at1 = at1
                self.atn = atn
        def metodo (self, arg):
                #instrucciones

objeto = NombreClase()
objeto.at1
objeto.metodo(arg)
```

Encapsulamiento

El encapsulamiento o ocultamiento de la información son dos puntos básicos en la abstracción de datos. Todo lenguaje permite definir objetos ocultando alguna parte de ellos (datos y/o métodos).



Encapsulamiento

Sintaxis:

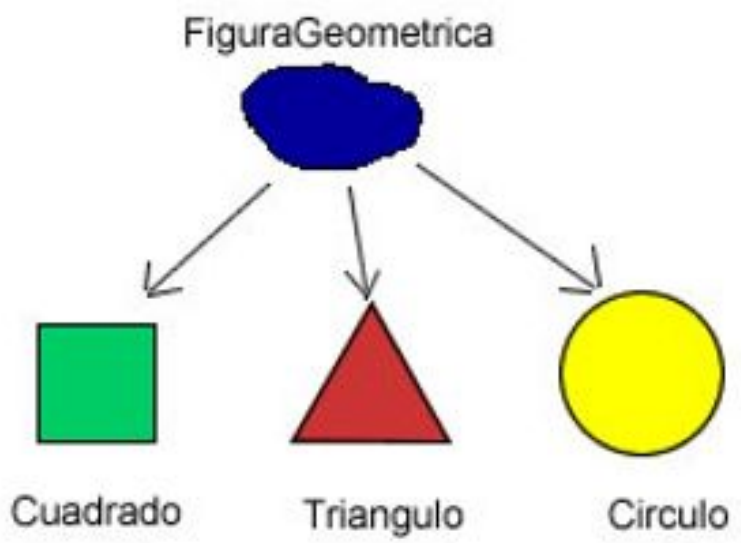
```
class NombreClase:
      def __init__ (self, at1, at2):
            self.at1 = at1
            self.__atPrivado = at2

      def __metodoPrivado (self):
            print(self.__atPrivado)

objeto = NombreClase()
objeto._NombreClase__metodoPrivado
```

Polimorfismo

Se denomina polimorfismo a la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.



Polimorfismo

Sintaxis:

```
class NombreClase:
        def metodo(self, objeto):
                #instrucciones

objeto1 = NombreClase()
objeto2 = NombreClase()
objeto1.metodo(objeto2)
```

Herencia

Es la facilidad mediante la cual una clase A (padre) hereda cada uno de sus atributos y métodos a una clase B (hija) de modo que pueda utilizarlos como si fueran suyos

Herencia

Sintaxis:

```
class ClasePadre():
        def __init__ (self, at1):
                self.at1 = at1

class ClaseHija(ClasePadre):
        def __init__ (self, at2):
                self.at1 = at2

objeto = ClaseHija():
```


Bien, ¿Algunas ideas para hacer que las mujeres se fijen en nosotros?
Java
C++
python
¿Hacer mas excepciones?
¿Redefinir nuestros métodos?
¿Y si las dejamos de tratar como objetos?
C++
python
ANSI C
szymi1000 / 9GAG
Java
ANSI C
C