

1000 PROGRAMADORES

Introducción a la Programación en Python



Ministerio de Educación
Cultura, Ciencia y Tecnología
Gobierno de Salta



Ministerio de Economía
y Servicios Públicos
Gobierno de Salta



Universidad
Nacional de Salta

MÓDULO 6

**Programación estructurada vs POO.
Clases, objetos y atributos.**





Python - Multiparadigma



admite PROGRAMACIÓN

Estructurada

Orientada a Objetos

Funcional



Programación Estructurada

- Los programas son más fáciles de entender.
- Un programa estructurado puede ser leído en secuencia, de arriba hacia abajo, sin necesidad de estar saltando de un sitio a otro en la lógica.
- La estructura del programa es más clara, las instrucciones están más relacionadas entre sí, y es más fácil comprender lo que hace cada función.
- Favorece la reducción del esfuerzo en las pruebas.
- Aumenta la productividad del programador.



Programación Orientada a Objetos (P00)

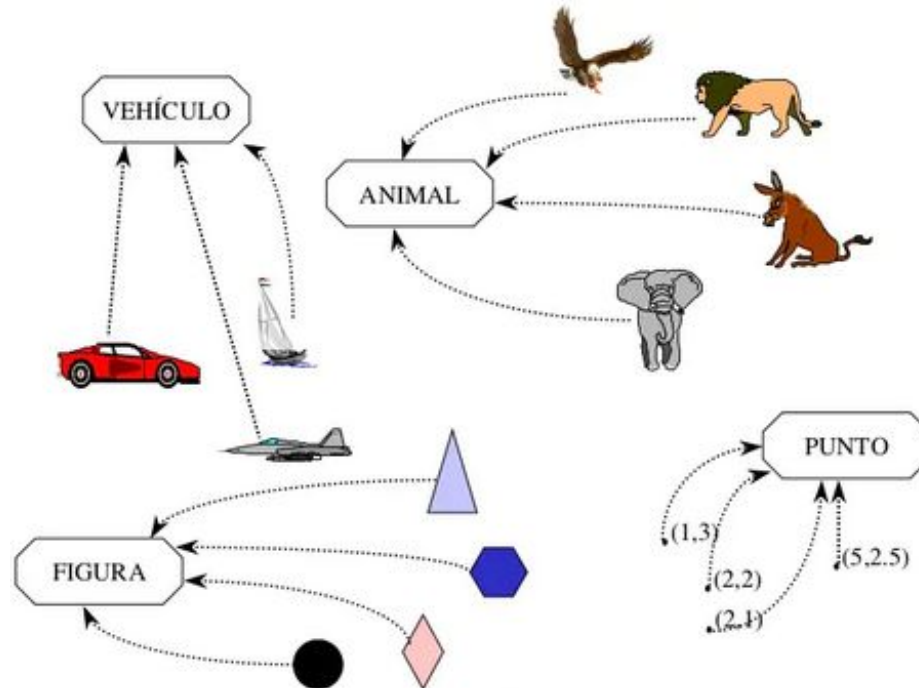
- Es una forma especial de programar, más cercana a la forma de expresar las cosas en la vida real que otros tipos de programación.
- Hay que pensar de una manera distinta, para escribir programas en términos de objetos, propiedades, métodos y otros conceptos nuevos.
- El adecuado diseño de clases favorece la reusabilidad.
- Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender y mantener, permiten ocultar detalles de implementación dejando visibles sólo los detalles más relevantes.
- La facilidad de añadir, suprimir o modificar nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.



Tratamos de establecer una equivalencia entre un objeto del mundo real con un componente software.

Todos los objetos presentan dos componentes principales:

- un **conjunto de características**
- un **comportamiento determinado**



Un auto, una moto, un velero y un avión tienen características como **color, marca y modelo**.

Su comportamiento puede ser descrito con operaciones como **frenar, acelerar o girar**.

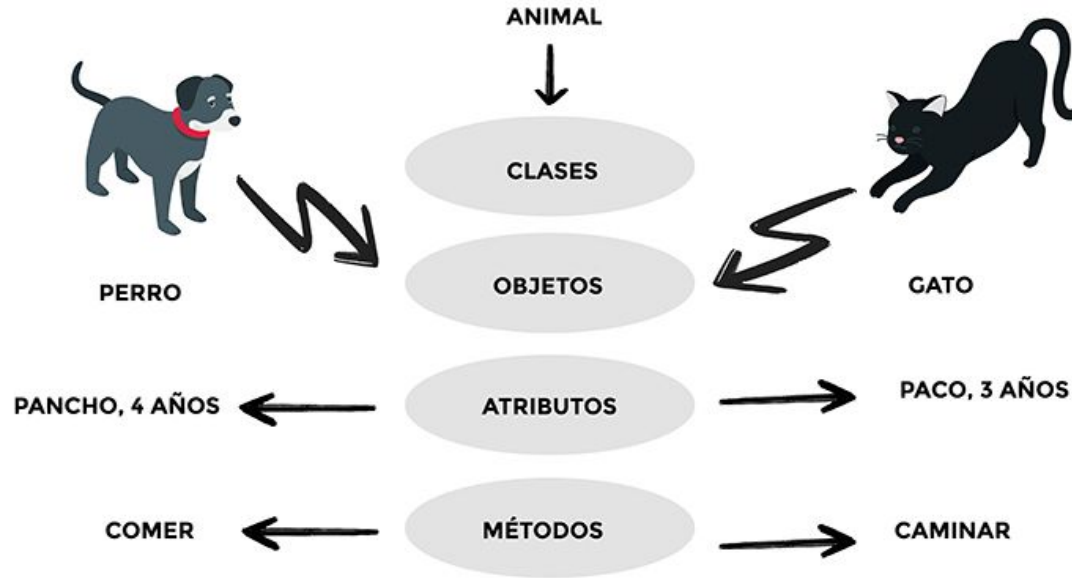


Características = Atributos

Lo que puede hacer = Métodos



Clases y Objetos



La definición de los atributos y operaciones de un objeto se lleva a cabo a través de una clase.

La instanciación es un mecanismo que nos permite crear un objeto que pertenece a una determinada clase.

Así podemos tener diferentes objetos que pertenezcan a la misma clase.



Entonces, una clase es un “molde” o una “plantilla” para generar un objeto.



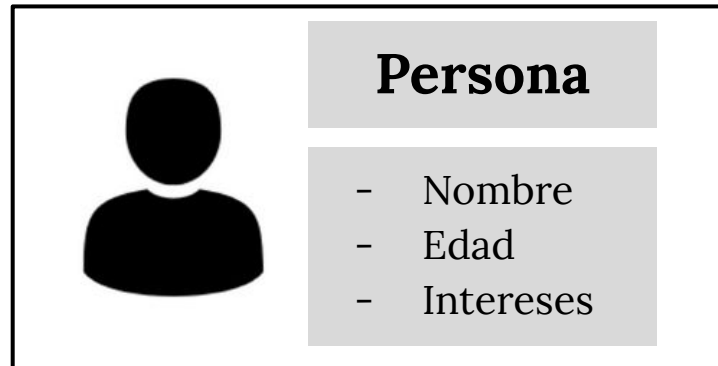


Ejemplo:

Imaginemos que tenemos una **persona**, donde nos interesa :

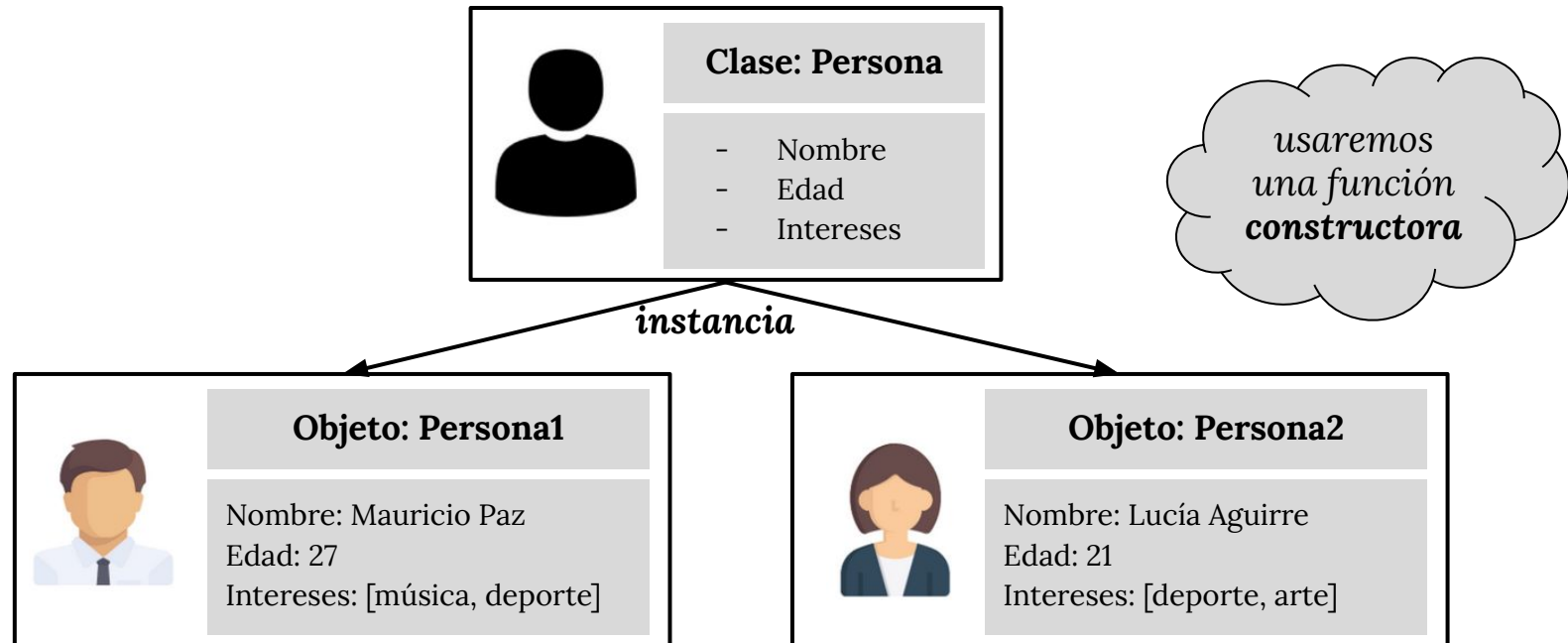
- Nombre (nombre, apellido)
- Edad
- Intereses.

Veamos esto en forma de ficha:





La **clase** **Persona** tiene **características** (nombre, edad, intereses). Teniendo nuestra **clase**, ya podemos generar **objetos** (personas) con características específicas.





La clase **Persona** se hace una **instancia** de **objeto** a Persona1 y otra **instancia** de **objeto** a Persona2.

Podemos crear cuantas instancias de objeto necesitemos.

Cuando creamos una clase debemos tener en cuenta:

- Atributos
- Métodos
- Constructor



Creación de Clase e instanciación de Objeto

```
class Persona:  
    pass  
  
Personal = Persona()
```

Es una clase vacía y sin mucha utilidad práctica, pero es la mínima clase que podemos crear.

Teniendo la **clase**, podemos crear un **objeto** de la misma como si se tratase de una variable normal. Nombre de la variable igual a la clase con los ().
Dentro de los paréntesis irían los parámetros de entrada si los hubiera.



Atributos

Atributos de instancia

Pertenecen a la instancia de la clase o al objeto.
Son atributos particulares de cada instancia.

Atributos de clase

Se trata de atributos que pertenecen a la clase, por lo tanto serán comunes para todos los objetos.



Atributos de Instancia

Crearemos un par de **atributos de instancia** para nuestra Persona, el nombre, la edad y sus intereses.

Creamos un método `__init__` que será llamado automáticamente cuando creamos un objeto.

Se trata del **constructor**.

```
class Persona:
# El método __init__ es llamado al crear el objeto
    def __init__(self, nombre, edad, intereses):
        print("Creando persona", nombre, edad, intereses)
        # Atributos de instancia
        self.nombre = nombre
        self.edad = edad
        self.intereses = intereses
```

El **self** que se pasa como parámetro de entrada del método es una variable que representa la instancia de la clase, y deberá estar siempre ahí.



Atributos de Instancia

Ahora que hemos definido el método *init* con tres parámetros de entrada, podemos crear el objeto pasando el valor de los atributos.

Usando `type()` podemos ver cómo efectivamente el objeto es de la clase `Persona`.

```
Personal = Persona("Mauricio Paz", 27, ["deporte", "musica"])  
print(type(Personal))
```

```
print(Personal.nombre)  
print(Personal.edad)  
print(Personal.intereses)
```

} podemos acceder a los atributos
usando el objeto y punto (.)



Atributos de Clase

```
class Persona:
    especie = "mamifero"
    # El método __init__ es llamado al crear el objeto
    def __init__(self, nombre, edad, intereses):
        print("Creando persona", nombre, edad, intereses)
        # Atributos de instancia
        self.nombre = nombre
        self.edad = edad
        self.intereses = intereses

Personal = Persona("Mauricio Paz", 27, ["deporte", "musica"])

print(Persona.especie)
print(Personal.especie)
```

Dado que es un atributo de clase, no es necesario crear un objeto para acceder al atributo. Se puede acceder también al atributo de clase desde el objeto.





Metodos

Usando `__init__` ya hemos definido un método, solo que uno especial.

Ahora definiremos métodos que le den alguna funcionalidad interesante a nuestra clase, siguiendo con el ejemplo de persona.

Diseñaremos dos métodos, hablar y caminar. El primero no recibirá ningún parámetro y el segundo recibirá el número de pasos que queremos andar.

Definiremos un método con `def` y el nombre, y entre `()` los parámetros de entrada que recibe, donde siempre tendrá que estar `self` el primero.



Metodos

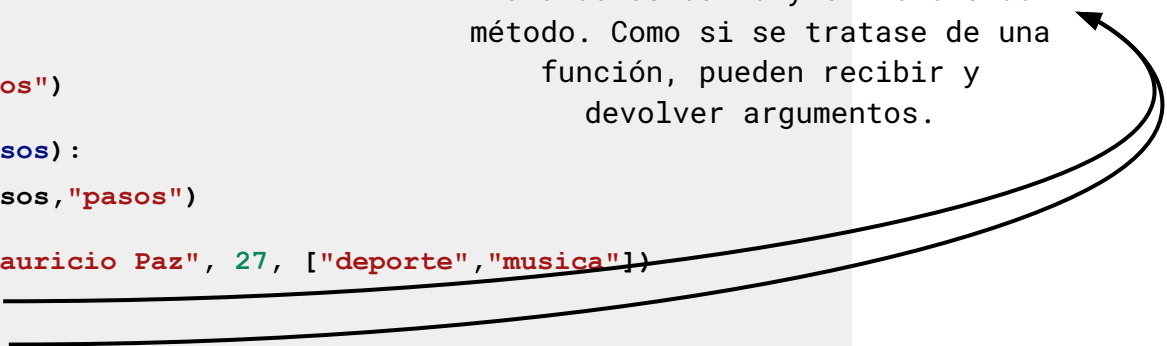
```
class Persona:
    especie = "mamifero"
    # El método __init__ es llamado al crear el objeto
    def __init__(self, nombre, edad, intereses):
        print("Creando persona", nombre, edad, intereses)
        # Atributos de instancia
        self.nombre = nombre
        self.edad = edad
        self.intereses = intereses

    def habla(self):
        print("Hola a todos")

    def camina(self, pasos):
        print("Camina", pasos, "pasos")

Personal = Persona("Mauricio Paz", 27, ["deporte", "musica"])
Personal.habla()
Personal.camina(10)
```

Si creamos un objeto, podremos hacer uso de sus métodos llamándolos con . y el nombre del método. Como si se tratase de una función, pueden recibir y devolver argumentos.





Ejemplo: Crear una clase llamada Cancion con los atributos: Nombre, Género y Duración (expresado en segundos). Construye los siguientes métodos para la clase:

- Un constructor;
- mostrar(): Muestra los datos de la canción;
- compara(cancion) indica si una canción es más larga que otra o son iguales

Escriba un programa que instancie 3 canciones y usando los métodos descritos antes, muestre los datos de la canción más larga y además muestre el nombre de la/s canciones de un género indicado por el usuario.





Definimos la clase

```
class Cancion:
    def __init__(self, nombre, genero, duracion):
        self.nombre = nombre
        self.genero = genero
        self.duracion = duracion
```



Definimos los métodos

```
def mostrar(self):  
    return "Nombre: " + self.nombre + "\nGénero: " + self.genero + "\nDuración: " +  
    str(self.duracion) + "\n"  
  
def retorna_nombre(self):  
    return self.nombre  
  
def retorna_genero(self):  
    return self.genero  
  
def compara(self, cancion):  
    if (self.duracion > cancion.duracion):  
        larga = 1  
    elif (self.duracion < cancion.duracion):  
        larga = -1  
    else:  
        larga = 0  
    return larga
```



Diseñamos el programa

```
Cancion1 = Cancion("jijiji", "Rock" , 180)
Cancion2 = Cancion("sol, arena y mar", "Balada" , 150)
Cancion3 = Cancion("mueve el toto", "Balada" , 100)

print("La cancion mas larga es: ")
if (Cancion1.compara(Cancion2)==1 and Cancion1.compara(Cancion3)==1):
    print(Cancion1.mostrar())
elif (Cancion2.compara(Cancion1)==1 and Cancion2.compara(Cancion3)==1):
    print(Cancion2.mostrar())
elif (Cancion3.compara(Cancion1)==1 and Cancion3.compara(Cancion2)==1):
    print(Cancion3.mostrar())
else:
    print("Tienen la misma duración")
```



Diseñamos el programa

```
generos=["Rock","Balada","Pop","Cumbia"]
gen=int(input("Ingrese género: \n1.Rock \n2.Balada \n3.Pop \n4.Cumbia\n"))

if (Cancion1.retorna_genero()==generos[gen-1]):
    print(Cancion1.retorna_nombre())
if (Cancion2.retorna_genero()==generos[gen-1]):
    print(Cancion2.retorna_nombre())
if (Cancion3.retorna_genero()==generos[gen-1]):
    print(Cancion3.retorna_nombre())
```