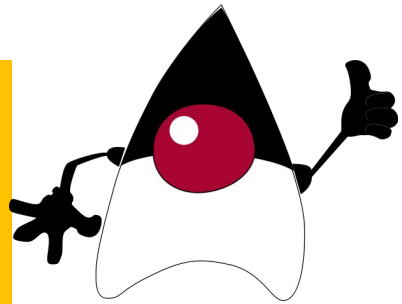
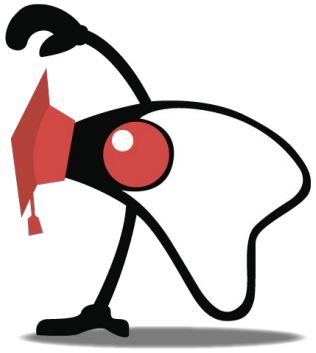
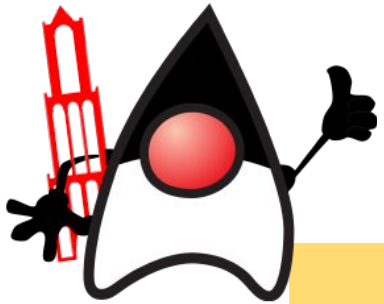
Curso FullStack Python

Codo a Codo 4.0







Clase 23

Python – Parte 5

¿Qué es Programación Orientada a Objetos?

- La programación orientada a objetos es un paradigma de programación, o una forma de pensar la programación, que se centra en objetos que pueden almacenar información y realizar acciones.
- Hace que el desarrollo de grandes proyectos de software sea más fácil y más intuitivo.
- Nos permite pensar sobre el software en términos de objetos del mundo real y sus relaciones.

POO

- Objetos
- Clases
- Abstracción
- Encapsulamiento
- Polimorfismo
- Herencia

Objetos

- Una entidad o “cosa” en tu programa, usualmente un sustantivo. Un ejemplo de un objeto sería una persona en particular.
  - Propiedades/Atributos
    - Nombre
    - Edad
    - Dirección
  - Comportamiento/Métodos
    - Caminar
    - Hablar
    - Respirar
  - Clase
    - Persona
  - Objeto
    - José Pérez de 23 años que vive en la calle Cucha cucha 123

Clases

- Las clases se usan para crear objetos.
- Podemos crear muchos objetos desde una sola clase.
- Las Clases definen el tipo de datos (type).
- El proceso de crear un objeto desde una clase se denomina instanciación.
- Ejemplo:
  - Crear una nueva variable denominada mi_nombre con el valor de “Matias”. Esta variable es en realidad una referencia a un objeto. El tipo de objeto es str porque para poder crearla, necesitamos instanciar desde la clase str.
  - mi_nombre = “Matias”

Clases y Objetos

- ¿Qué pasa si queremos crear nuestra propia clase?
- Una clase de Python es esencialmente una plantilla para un nuevo tipo de objeto que puede almacenar información y realizar acciones.
- Ejemplo:
  - Crear una clase Perro.
  - class Perro:
- Notación CamelCase para los nombres de las clases.

Atributos de Instancia

- Agreguemos propiedades que todos los perros deberían tener.
- Las propiedades de todos los objetos de tipo Perro deben definirse en un método denominado __init__()
  - __init__() define el estado inicial del objeto asignando valores a las propiedades del objeto.
  - Esto significa que __init__() inicializa a cada nueva instancia de la clase.
  - Le puedes pasar la cantidad de parámetros que quieras, siempre y cuando el primero sea self.
- Los métodos se declaran de manera similar a las funciones con def como primer elemento.

```
class Perro:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
```

Atributos de Clase

- Por otro lado, los atributos de clase son atributos que tienen el mismo valor para todas las instancias.
- Puedes definir atributos de clase declarándolos por fuera del método __init__().
- Ejemplo: en la misma clase Perro podemos asignar un atributo “Género” que sea siempre el mismo para todos.
- Los atributos de clase siempre se encontrarán directamente debajo de la definición del nombre de la clase.
- Debes recordar que cualquier modificación al valor de una variable de clase es arrastrada hacia las instancias.

```
class Perro:
    # Atributo de Clase
     genero= "Canis"
     def __init__(self, nombre, edad):
         self.nombre = nombre
         self.edad = edad
```

Instancias

- Puedes instanciar rápidamente una clase en el interprete de Python tipeando el nombre de la clase con paréntesis. Esto hace que la instancia de Perro se coloque en una posición de memoria. Para poder almacenarla en una variable hacemos lo siguiente:
  - miMascota = Perro()
  - otroPerro = Perro()
- Para instanciar el perro con atributos de nombre y edad hacemos lo siguiente:
  - miMascota = Perro("Popey", 8)
  - otroPerro = Perro("Bart", 5)
- Obviamos el parámetro self ya que es transparente al usuario en Python.
- Podemos acceder a cada atributo de cada instancia mediante el nombre de la variable y el nombre del atributo asociado:
  - miMascota.edad
  - otroPerro.nombre
- Si queremos cambiar un atributo podríamos hacer lo siguiente:
  - otroPerro.edad= 10
  - otroPerro.genero= “Felis” (*)

Métodos de instancias

- Los métodos de instancia son “funciones” definidas dentro de una clase que solo pueden ser llamadas desde la instancia de la clase.
- Como el método __init__(), en un método de instancia siempre el primer parámetro será self.

```
class Perro:
    # Atributo de Clase
    genero= "Canis"
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
    # Método de instancia
    def imprimir(self):
        return f'{self.nombre} tiene {self.edad} años.'
    # Otro método de instancia
    def ladrar(self, sonido):
        return f'{self.nombre} dice {sonido}.'
```

Llamada a un método

- Para llamar a cada método simplemente utilizamos el operador unario (el punto) y entre paréntesis pasamos los parámetros (que puede o no tener):
  - miMascota = Perro(“Paka”, 11)
  - miMascota.imprimir()
  - miMascota.ladrar(“Guau guau!”)