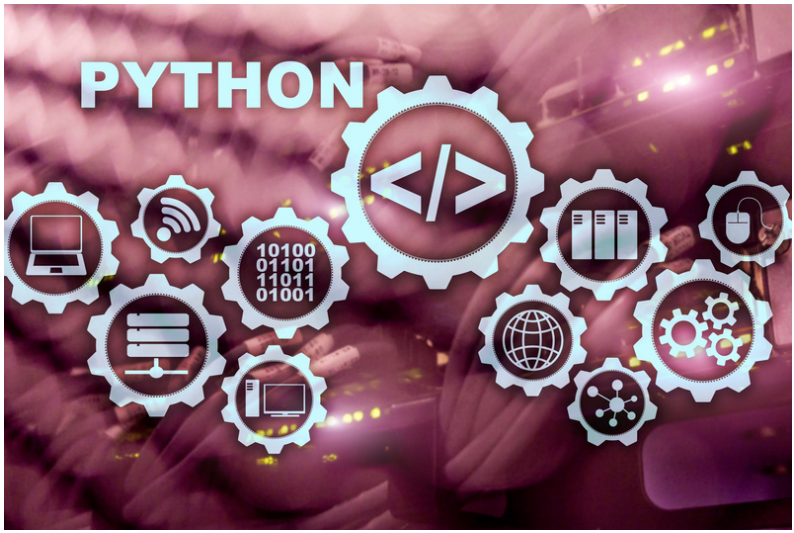

LENGUAJES DE
PROGRAMACIÓN
PYTHON

Clases y Objetos en Python

En la programación orientada a objetos (POO), las clases y los objetos son conceptos fundamentales. Permiten organizar y estructurar el código de manera más eficiente y modular, al permitirnos modelar entidades del mundo real y sus interacciones. En esta lección, exploraremos las clases y los objetos en Python, aprendiendo cómo definir clases, crear objetos y utilizarlos para crear programas más flexibles y reutilizables.



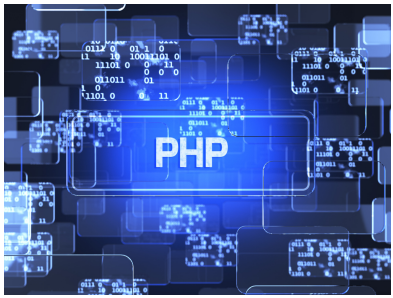
Clases en Python

Una clase en Python es una plantilla o un plano para crear objetos. Define las propiedades y los comportamientos comunes que los objetos de esa clase tendrán. Una clase se compone de atributos (variables) y métodos (funciones).

Para definir una clase en Python, utilizamos la palabra clave "class" seguida del nombre de la clase. A continuación, se muestra la sintaxis básica de una clase:

```
class NombreDeClase:
    # Definición de atributos
    # Definición de métodos
```

Los atributos son variables que almacenan datos relacionados a la clase, mientras que los métodos son funciones que definen el comportamiento de la clase.



A continuación, se muestra un ejemplo de una clase "Persona" que tiene atributos como nombre y edad, y métodos como saludar y obtener_edad:

```
class Persona:

    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def saludar(self):
                    print(f"Hola,  mi  nombre  es
{self.nombre}")

    def obtener_edad(self):
        return self.edad
```

En este ejemplo, la clase "Persona" tiene dos atributos: "nombre" y "edad". El método __init__ es un método especial llamado constructor, que se ejecuta automáticamente al crear un objeto de la clase. Los métodos "saludar" y "obtener_edad" definen el comportamiento de la clase.

Objetos en Python

Un objeto es una instancia de una clase. Representa una entidad del mundo real basada en la definición de la clase. Al crear un objeto, obtenemos una copia de la estructura y los comportamientos definidos en la clase.

Para crear un objeto en Python, utilizamos la sintaxis de llamada a la clase, seguida de paréntesis. A continuación, se muestra un ejemplo de creación de un objeto "persona" basado en la clase "Persona":

```
persona = Persona(”Ana”, 25)
```

En este caso, hemos creado un objeto "persona" utilizando la clase "Persona" y hemos pasado los valores "Ana" y 25 para los atributos "nombre" y "edad" respectivamente.

Podemos acceder a los atributos y métodos de un objeto utilizando la notación de punto. Por ejemplo:

```
print(persona.nombre) # Imprime ”Ana”
persona.saludar() # Imprime ”Hola, mi nombre
es Ana”
edad = persona.obtener_edad()
print(edad) # Imprime 25
```

Aquí, hemos accedido al atributo "nombre" y al método "saludar" del objeto "persona". También hemos utilizado el método "obtener_edad" para obtener el valor de la edad y asignarlo a la variable "edad".

Beneficios de las Clases y los Objetos

El uso de clases y objetos en Python ofrece varios beneficios:

Modularidad: Las clases nos permiten encapsular atributos y métodos relacionados en un solo lugar, lo que promueve la modularidad y la organización del código. Cada clase se centra en una entidad específica, lo que facilita la comprensión y el mantenimiento.

Reutilización de código: Al definir una clase, podemos crear múltiples objetos basados en ella. Esto nos permite reutilizar el código y los comportamientos definidos en la clase, evitando la duplicación y mejorando la eficiencia.

Herencia: La herencia nos permite crear nuevas clases basadas en clases existentes. Esto nos permite extender y personalizar la funcionalidad de una clase base, ahorrando tiempo y esfuerzo en el desarrollo.

Polimorfismo: El polimorfismo nos permite utilizar objetos de diferentes clases de manera intercambiable, siempre y cuando cumplan con una interfaz común. Esto nos brinda flexibilidad y nos permite escribir código más genérico y flexible.

Ejemplo de Uso de Clases y Objetos en Python

Supongamos que estamos desarrollando un programa para una tienda en línea que vende productos. Podemos utilizar clases y objetos para modelar los productos y sus características. A continuación, se muestra un ejemplo:

```
class Producto:

    def __init__(self, nombre, precio):
        self.nombre = nombre
        self.precio = precio

    def mostrar_informacion(self):
        print(f"Producto: {self.nombre}")
        print(f"Precio: ${self.precio}")

producto1 = Producto("Camiseta", 25.99)
producto2 = Producto("Pantalón", 39.99)

producto1.mostrar_informacion()
producto2.mostrar_informacion()
```

En este ejemplo, hemos creado una clase "Producto" con atributos "nombre" y "precio". El método "mostrar_informacion" muestra la información del producto en la pantalla.

Luego, hemos creado dos objetos "producto1" y "producto2" basados en la clase "Producto" y hemos asignado valores a los atributos correspondientes.

Finalmente, hemos invocado el método "mostrar_informacion" en cada objeto para mostrar la información del producto en la pantalla.

Las clases y los objetos son conceptos fundamentales en la programación orientada a objetos. Al utilizar clases, podemos modelar entidades del mundo real y definir sus atributos y comportamientos. Los objetos nos permiten crear instancias de estas clases y utilizar sus atributos y métodos para interactuar con el programa.

El uso de clases y objetos en Python promueve la modularidad, la reutilización de código y la organización estructurada del programa. Al dividir el código en clases y objetos, podemos mejorar la legibilidad, el mantenimiento y la escalabilidad del programa.

Recuerda seguir las convenciones de nomenclatura de clases, utilizar comentarios para documentar tus clases y utilizar métodos para encapsular la lógica y el comportamiento relacionado.