1/4/23 - **Declarando un método constructor en Python**

|  |  |
| --- | --- |
| Puntos clave | Notas |
|  | En Python encontrarás un concepto denominado Métodos Mágicos, estos métodos son llamados automática y estrictamente bajo ciertas reglas. El método constructor en Python forma parte de esta familia de métodos y como aprendimos en la clase anterior lo declaramos usando \_\_init\_\_, aunque si nos ponemos estrictos este método no construye el objeto en sí. El encargado de hacer esto es \_\_new\_\_ y el método \_\_init\_\_ se encargará de personalizar la instanciación de la clase, esto significa que lo que esté dentro de \_\_init\_\_ será lo primero que se ejecute cuando se cree un objeto de esta clase.  Para nuestro proyecto tenemos la necesidad de que algunas variables se inicialicen obligatoriamente cuando ocurra la instanciación. Así que declaremos el método \_\_init\_\_ en las clases de nuestro proyecto con las propiedades obligatorias.  Para la clase Account quedaría algo así, notarás que usamos la palabra clave self, esta es muy parecida a lo que venimos trabajando a otros lenguajes con this. Y como su nombre lo dice hace referencia a los datos que componen la clase, en este caso self.name está llamando al atributo name que se encuentra en la línea 3 de la clase y, le está asignando el dato que se pasa en el método \_\_init\_\_ de la línea 8.  Lo que notarás de diferente es que cambiamos el tipo de dato de driver, ahora es de tipo Account y como ves está solicitando los dos datos obligatorios para instanciar un objeto de este tipo. Esto lo verás más en acción en el próximo fragmento de código del archivo main.py. Además, mucho ojo, en la primera línea observamos que es importante importar la clase para poderla usar. |
| Sumario: | |

1/4/23

|  |  |
| --- | --- |
| Puntos clave | Notas |
|  | Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated |
| Sumario: | |

1/4/23

|  |  |
| --- | --- |
| Puntos clave | Notas |
|  | En PHP  La palabra Polimorfismo puede que cause miedo al escucharla por primera vez, a mi me pasó, pero realmente no es nada del otro mundo, simplemente es darle a una clase diferentes comportamientos, prácticamente lo que hacemos al heredar la clase Base en la clase Admin o User es sobreescribir su constructor para que se comporte de una manera distinta, pero que el resto de la clase Base siga igual, prácticamente, si yo heredo una clase, puedo cambiar el comportamiento de sus métodos sobreescribiéndolos, y si alguien hereda mi clase, se comportará de la nueva forma en la que yo la definí, pero si alguien hereda la clase padre, se comportará de la forma que tiene por defecto, de esa forma no tenemos que cambiar el código base y podemos conseguir que una clase se comporte de maneras diferentes dependiendo de la situación. |
| Sumario: | |