**INTRODUCCIÓN A PYTHON**

0 - INTRO TO PYTHON

1 - VARIABLES & OPERATORS

2 - DATA STRUCTURES

3 - CONDITIONAL STATEMENTS

4 - ITERATIONS ( FOR LOOPS, WHILE LOOPS & NESTED LOOPS)

5 - PSEUDOCODE

0 - INTRO TO PYTHON:

Python is described as:

* **Dynamically Typed**: No hay declaración de una variable, solo una declaración de asignación. Cuando **declaramos una variable** en C o lenguajes similares, esto **reserva un área de memoria** para almacenar **valores permitidos por el tipo de datos de la variable** (si es una variable entera, la memoria asignada se leerá como un entero y así sucesivamente), ¡no podemos mezclar tipos!.

Pero Python es un lenguaje escrito dinámicamente. **No conoce el tipo de variable hasta que se ejecuta el código** (Entonces la declaración no sirve de nada). Lo que hace es **almacenar ese valor** en alguna ubicación de memoria **en base al nombre de la variable**, no su formato (en vez de reservar, simplemente almacena) y hace que el contenido del contenedor sea accesible a través de ese nombre de variable (el **tipo de datos** no importa ya que **lo conoceremos en el momento de la ejecución**).

# This will store 6 in the memory and binds (une) the name x to it. After it runs, type of x will be “int”.

x = 6

print(type(x))

# This will store 'hello' at some location “int” the memory and binds name x to it. After it runs, the type of x will be “str2.

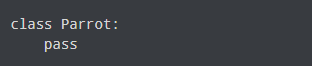
x = 'hello'

print(type(x))

* **Interpreted:** “Intérprete”  **→ ejecuta** directamente **instrucciones escritas en un lenguaje de programación** o de **scripting** (de secuencias de comandos [automatización]) **sin** necesidad de **que hayan sido compiladas** (traducidas al lenguaje de la máquina de golpe “de una”). ES CÓDIGO INDEPENDIENTE AL DE LA MÁQUINA (ya que interpreta las instrucciones en un código de bytes legible por máquina → traduce el código dado línea por línea y si encuentra un error detiene la traducción hasta que se corrija el error).
* **Interactive**: ya que es como si “hablases” con él. Cuando se ingresa una declaración de Python, seguida de la tecla Return, si corresponde, el resultado se imprimirá en la pantalla, inmediatamente, en la siguiente línea. Es por ello que es muy útil para la depuración.
* **Object-oriented**: OOP (Object-Oriented Programming). Un objeto tiene dos características:
  + atributos
  + comportamiento

PRINCIPIOS BÁSICOS:

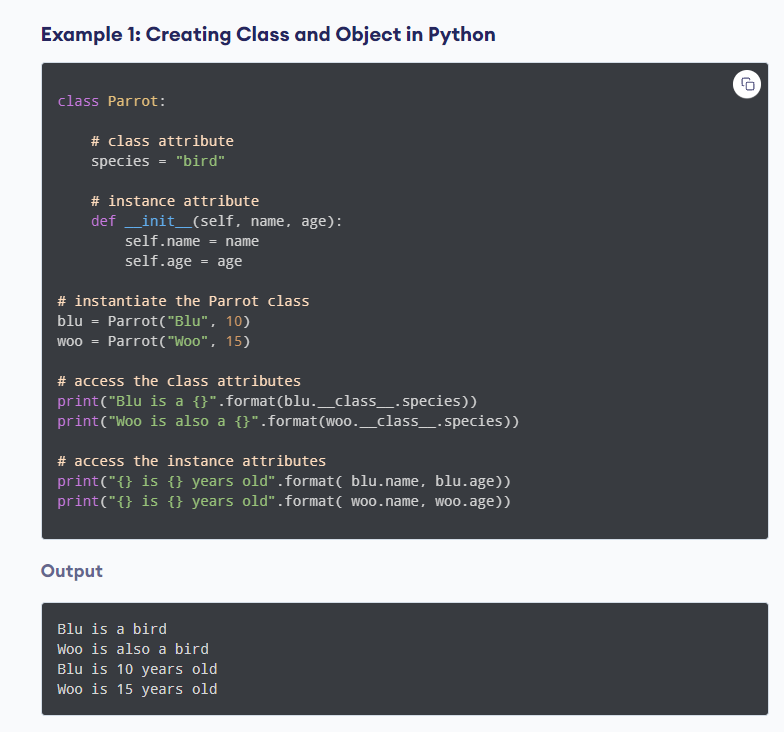
* CLASS: is a blueprint (etiqueta) for the object. From class, we construct **instances**. An instance is a **specific object** created from a particular class.



* OBJECT: is an instantiation of a class. When class is defined, only the description for the object is defined (por lo tanto, no se asigna memoria ni almacenamiento.



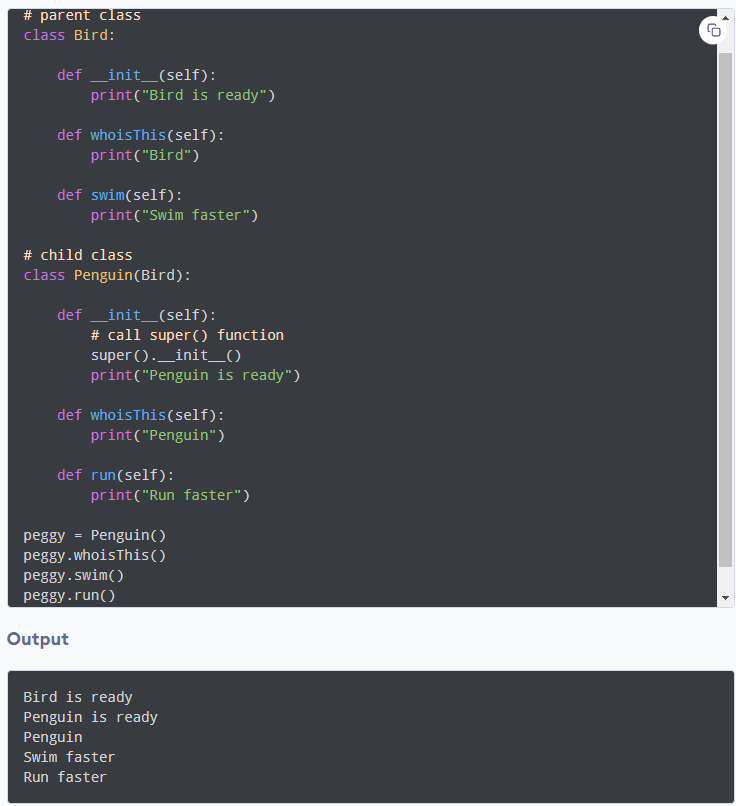
“obj” es un objeto de la class “parrot”



* METHODS: son funciones definidas dentro del cuerpo de una clase

****

* INHERITANCE: La herencia es una forma de crear una nueva clase para usar detalles de una clase existente sin modificarla. La clase recién formada es una clase derivada (o clase secundaria). De manera similar, la clase existente es una clase base (o clase principal).



The child class inherits the functions of parent class. We can see this from the swim() method.

* ENCAPSULATION: Usando OOP en Python, podemos restringir el acceso a métodos y variables. Esto evita que los datos se modifiquen directamente, lo que se denomina encapsulación. En Python, denotamos atributos privados usando un guión bajo como prefijo, es decir, simple \_o doble \_\_.



* POLYMORPHISM: capacidad (en OOP) de utilizar una **interfaz común para múltiples** formularios (**tipos de datos**).

Supongamos que necesitamos colorear una forma, hay múltiples opciones de forma (rectángulo, cuadrado, círculo). Sin embargo, podríamos usar el **mismo método** para colorear cualquier forma. Este concepto se llama polimorfismo.



**Puntos clave para recordar:**

* La programación orientada a objetos hace que el **programa sea fácil de entender** y **eficiente**.
* Dado que la clase se puede compartir, el **código se puede reutilizar** (ya que organiza el código de su programa de forma modular).
* Los **datos están seguros y protegido**s con la *abstracción de datos*.
* El *polimorfismo* permite la **misma interfaz para diferentes objetos,** por lo que los programadores pueden escribir código eficiente.