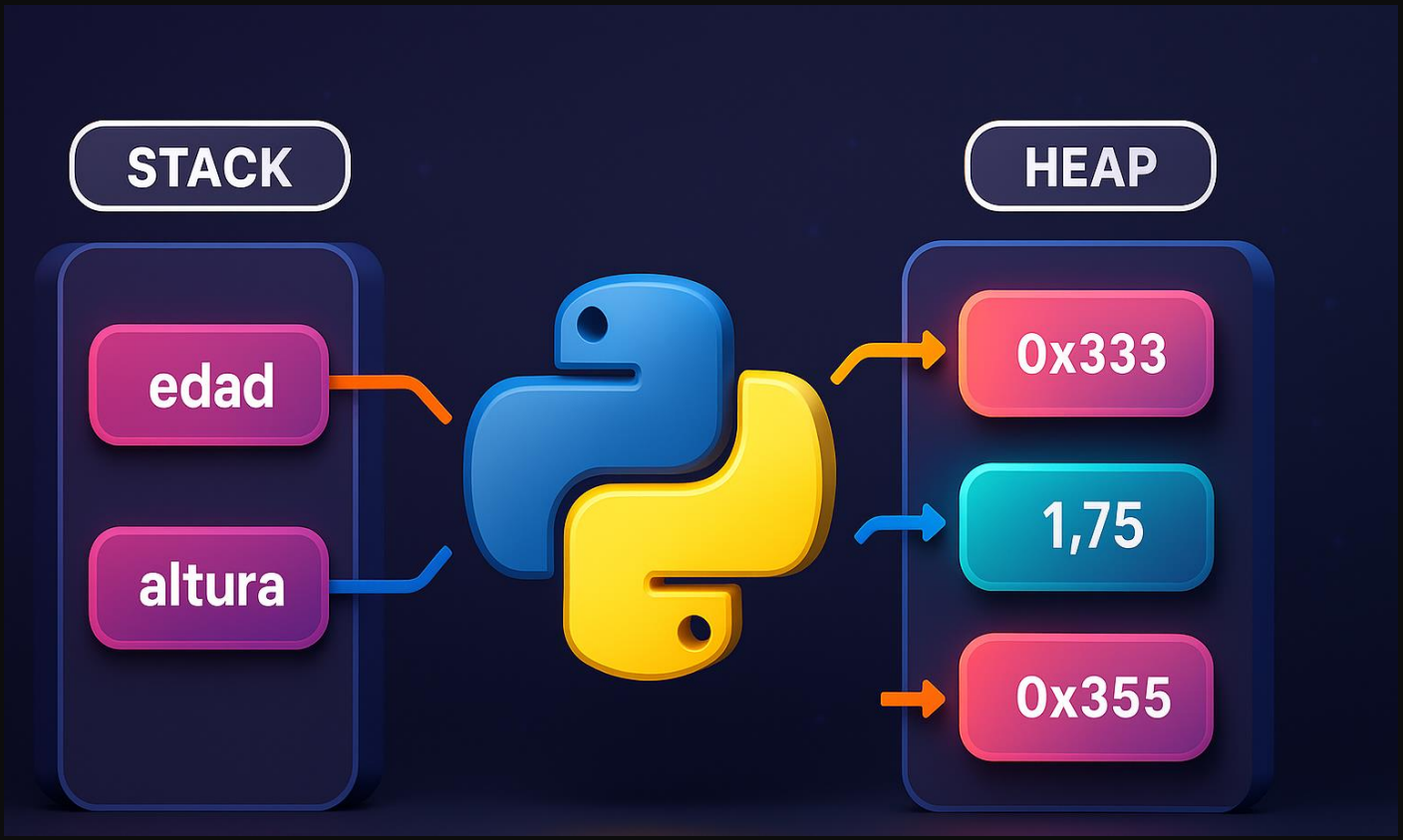

🧠 Variables y la Memoria RAM en Python

Introducción

En esta lección aprenderás cómo se almacenan las variables en la memoria RAM cuando trabajamos con Python. Exploraremos conceptos como memoria stack y heap, el comportamiento de los objetos, las referencias en memoria, y cómo Python maneja internamente los valores que asignamos a nuestras variables. También veremos una versión simplificada para facilitar la comprensión en futuras lecciones.

🔹 Paso 1: ¿Qué ocurre al crear una variable?

Cada vez que creamos una variable en Python y le asignamos un valor, **se reserva un espacio en la memoria RAM** de nuestra computadora.

Por ejemplo:

```
edad = 30
```

📌 Esto **no significa** que la variable almacena directamente el valor `30`. En realidad, en Python **todos los valores son objetos**. La variable `edad` **apunta** a un objeto que contiene el valor `30`.

🔹 Paso 2: Tipos de memoria involucrados

Python utiliza dos áreas principales en la memoria RAM:

- **Stack (pila):** donde se almacenan las **referencias** de las variables.
- **Heap (montón):** donde se almacenan los **objetos** como los valores reales.

🔁 Por lo tanto, cuando creamos una variable, se crea **una referencia en el stack** que apunta a un **objeto en el heap**.

🔹 Paso 3: Creación de distintos tipos de objetos

Veamos otro ejemplo:

```
altura = 1.75
```

- Se crea una variable `altura`.
- Esta variable apunta a un **objeto flotante** (tipo `float`) con valor `1.75`.

💡 Importante: aunque sea otro tipo de dato, sigue siendo un objeto almacenado en el heap, diferente al de `edad`.

🔹 Paso 4: ¿Qué pasa si se modifica el valor?

Si después de varias líneas de código cambiamos el valor de `edad`, por ejemplo:

```
edad = 32
```

Lo que ocurre es:

- Ya no se apunta al objeto con valor `30`.
- Se crea un **nuevo objeto** con el valor `32`.
- La variable `edad` ahora apunta a esta **nueva referencia**.

🧠 El objeto anterior con valor `30` puede ser eliminado por el recolector de basura si ya no se utiliza.

🔹 Paso 5: Direcciones de memoria

Cada objeto creado en el heap tiene una **dirección de memoria** única, generalmente expresada en formato hexadecimal, por ejemplo:

```
0x333   → valor 30
0x444   → valor 1.75
0x555   → valor 32
```

Python maneja internamente estas referencias sin que el programador tenga que gestionarlas manualmente.

🔹 Paso 6: Versión simplificada para continuar aprendiendo

Para simplificar los siguientes temas, asumiremos que una variable **almacena directamente** su valor.

Ejemplo:

```
edad = 30
altura = 1.68
```

🎯 Aunque sabemos que en realidad se guarda una referencia al objeto, vamos a tratarlo como si el valor estuviera almacenado directamente en la variable. Esto facilitará el aprendizaje sin perder precisión.

✅ Conclusión

En esta lección comprendiste cómo Python administra las variables en la memoria RAM utilizando referencias, objetos y dos tipos de áreas de memoria: stack y heap. También aprendiste que al modificar una variable, se crea un nuevo objeto en memoria, y que usaremos una versión simplificada para continuar aprendiendo de forma clara y progresiva. ¡Ahora estás listo para seguir con ejercicios prácticos! 💡💪

Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 👋

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de GlobalMentoring.com.mx