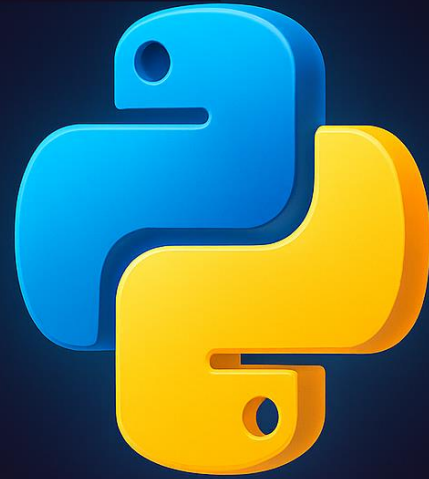
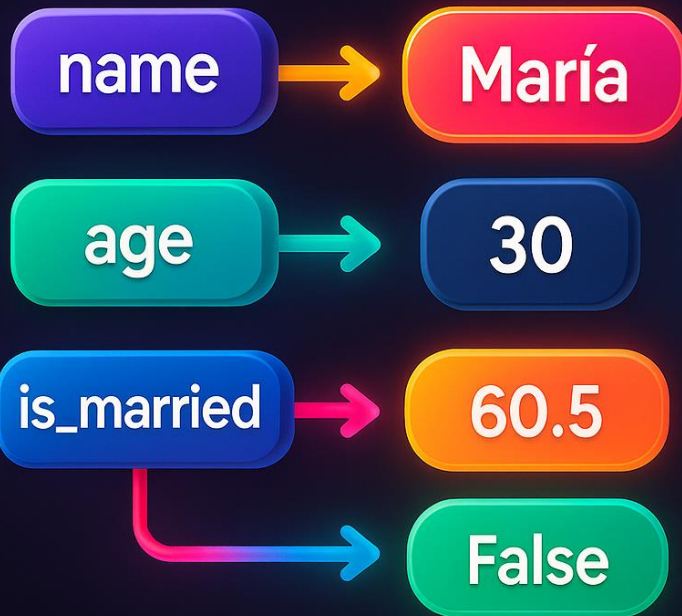


# VARIABLES



## Ejemplo de Reglas y Convenciones para Nombres de Variables en Python

### Introducción

En esta lección vamos a aplicar de forma práctica las reglas y buenas prácticas para nombrar variables en Python. Veremos ejemplos claros de qué hacer y qué evitar, incluyendo errores comunes como comenzar con dígitos, usar palabras reservadas o ignorar la sensibilidad a mayúsculas y minúsculas. También aprenderás a aplicar la convención `snake_case`, muy usada en Python, para que tu código sea más legible y profesional.

#### 1. Crear el archivo de trabajo

📄 Vamos a trabajar dentro del proyecto de variables, creando un nuevo archivo Python llamado:

**Ruta y nombre del archivo:**

Variables/reglas\_convenciones.py

---

## 📌 2. Declarar variables siguiendo las reglas correctas

📄 Primero vamos a declarar una variable que sigue correctamente las reglas de Python.

**Descripción breve**

Creamos una variable con letras minúsculas y separada con guión bajo (`snake_case`), iniciando con una letra.

```
nombre_usuario = 'Juan Pérez'
```

✓ Esta declaración es válida porque:

- Comienza con una letra.
  - Usa solo letras, guiones bajos y dígitos (permitidos).
  - Sigue la convención `snake_case`.
- 

## ✗ 3. Error al comenzar con un dígito

**Descripción breve**

Declaramos una variable que empieza con un número, lo cual no está permitido en Python.

```
1nombre_usuario = 'Carla Gómez'
```

🔴 Esto genera un error: `SyntaxError: invalid decimal literal`

⚠ Python no permite que los nombres de variables empiecen con números.

---

## ⊖ 4. Error al usar palabras reservadas

**Descripción breve**

Intentamos declarar una variable con una palabra reservada del lenguaje (`class`).

```
class = 'MiClase'
```

🛑 Esto también lanza un `SyntaxError`.

Las palabras clave como `if`, `for`, `class`, `try` no se pueden usar como nombres de variables.

---

## 🔧 5. Solución alternativa con modificación leve

### Descripción breve

Si realmente necesitas una palabra similar, puedes modificarla para evitar el conflicto.

```
klass = 'MiClase'
```

✓ Aquí modificamos `class` a `klass`, evitando el error.

---



## 6. Sensibilidad a mayúsculas y minúsculas

### Descripción breve

Python distingue entre mayúsculas y minúsculas. Veamos cómo:

```
nombre = 'Juan'  
Nombre = 'Carla'
```

```
print(nombre)  
print(Nombre)
```

✂ Resultado:

```
Juan  
Carla
```

Si usas una variable como `NOMBRE`, sin haberla definido antes, obtendrás:

```
print(NOMBRE) # ✖ Error: NameError
```

---



## 7. Convención Snake Case

## Descripción breve

Se recomienda nombrar variables usando letras minúsculas y separando las palabras con guiones bajos.

```
nombre_completo = 'Ricardo Esparza'
```

! Evita nombres demasiado largos o de una sola letra. Usa nombres descriptivos y claros.

---



## 8. Uso de prefijos y sufijos descriptivos

### Descripción breve

A veces es útil agregar prefijos o sufijos para hacer el significado de la variable más claro.

```
es_casado = True
nombre_txt = 'archivo.txt'
```

- `es_casado`: el prefijo `es_` indica que se trata de una condición booleana.
  - `nombre_txt`: el sufijo `_txt` indica que el contenido se refiere a un archivo de texto.
- 



## Código completo del archivo trabajado

```
# Regla y convenciones en nombres de variables

# Ejemplos de reglas estrictas
nombre_usuario = 'Juan Perez'
# 1nombre_usuario = 'Karla Gomez'

# No podemos usar palabras reservadas
#class = 'Mi clase'
klass = 'Mi clase'

# Sensibles a mayúsculas y minúsculas
nombre = 'Juan'
Nombre = 'Karla'
print(nombre)
print(Nombre)
# print(NOMBRE) esta variable no ha sido definida


# snake case
```

```
nombre_completo = 'Ricardo Esparza'  
  
# prefijos y sufijos  
es_casado = False  
nombre_txt = 'archivo.txt'
```


## Conclusión

En esta lección aprendiste:

- Las reglas estrictas para declarar nombres válidos en Python.
- Qué errores comunes debes evitar.
- Cómo aplicar la convención `snake_case` correctamente.
- La importancia de usar nombres descriptivos para mantener tu código claro y profesional.

Estas buenas prácticas son la base para desarrollar programas legibles, mantenibles y con estándares profesionales en Python .

---

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

¡Saludos! 

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)