Introducción al Lenguaje Python y Sentencias Básicas Básicas

Python es un lenguaje de programación creado por Guido van Rossum a finales de los años 80, con su primera versión lanzada en 1991. Contrario a la creencia popular, su nombre no proviene de una serpiente, sino del programa de comedia británico "Monty Python's Flying Circus", que su creador admiraba.

La filosofía de Python es hacer que programar sea simple, legible y agradable para todos. Es un lenguaje multipropósito utilizado por grandes empresas como Google, Netflix, Facebook y la NASA, diseñado con el claro propósito de ser accesible y fácil de aprender.

R **por Kibernum Capacitación**

Preguntas de Activación

¿Qué habilidades personales crees que serán clave para aprender un lenguaje de programación como Python en este curso intensivo (bootcamp)?

¿Cómo se relaciona el aprendizaje de herramientas como Python con las expectativas laborales del perfil técnico que estás formando?

¿Qué importancia tiene para ti construir un portafolio de productos a lo largo del curso? ¿Cómo podrías usarlo al momento de postular a un empleo?

¿Qué es un lenguaje de programación?

Comunicación con la computadora

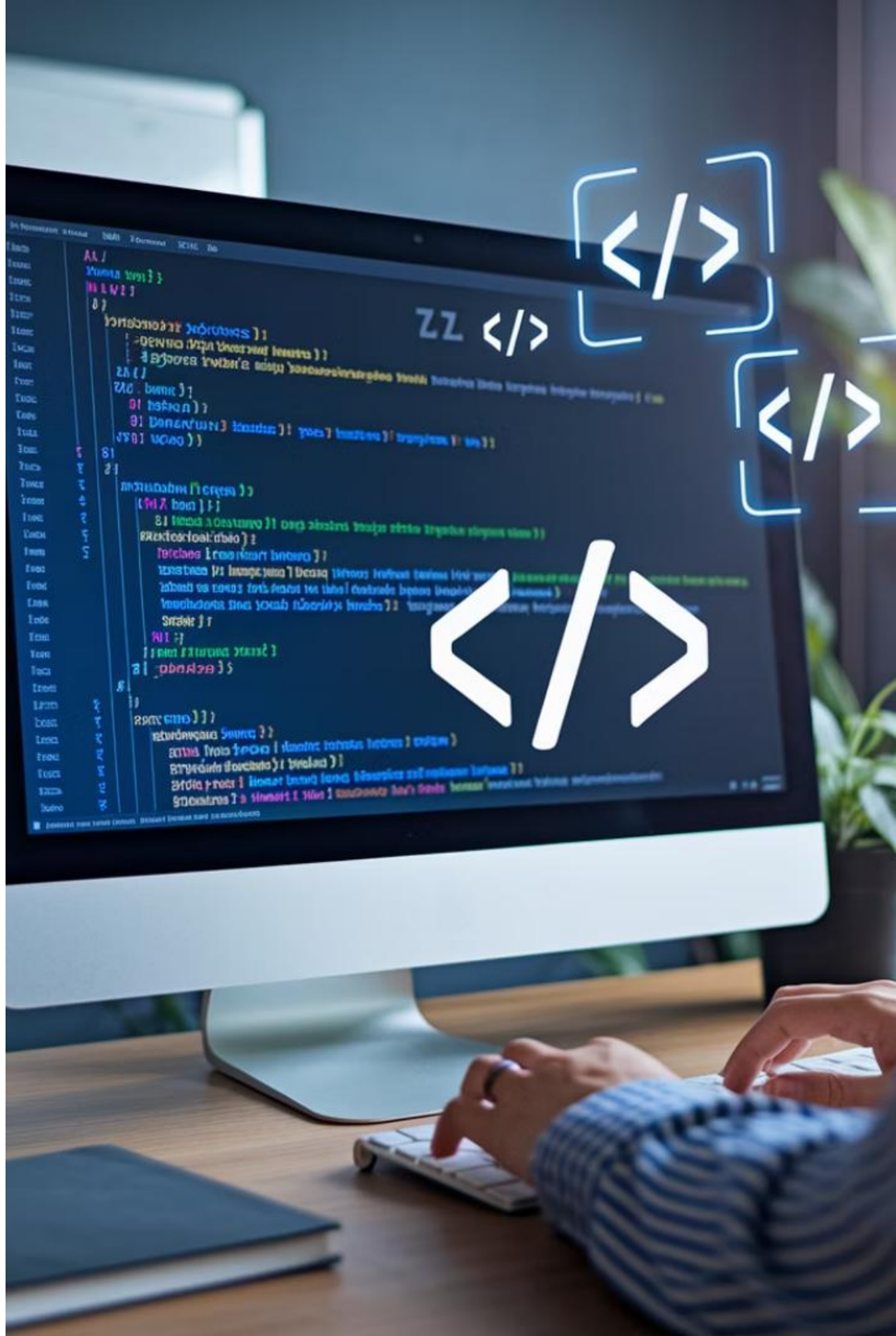
Es una forma de hablarle a la computadora para que haga lo que tú quieres, desde mostrar mensajes en pantalla hasta realizar cálculos complejos.

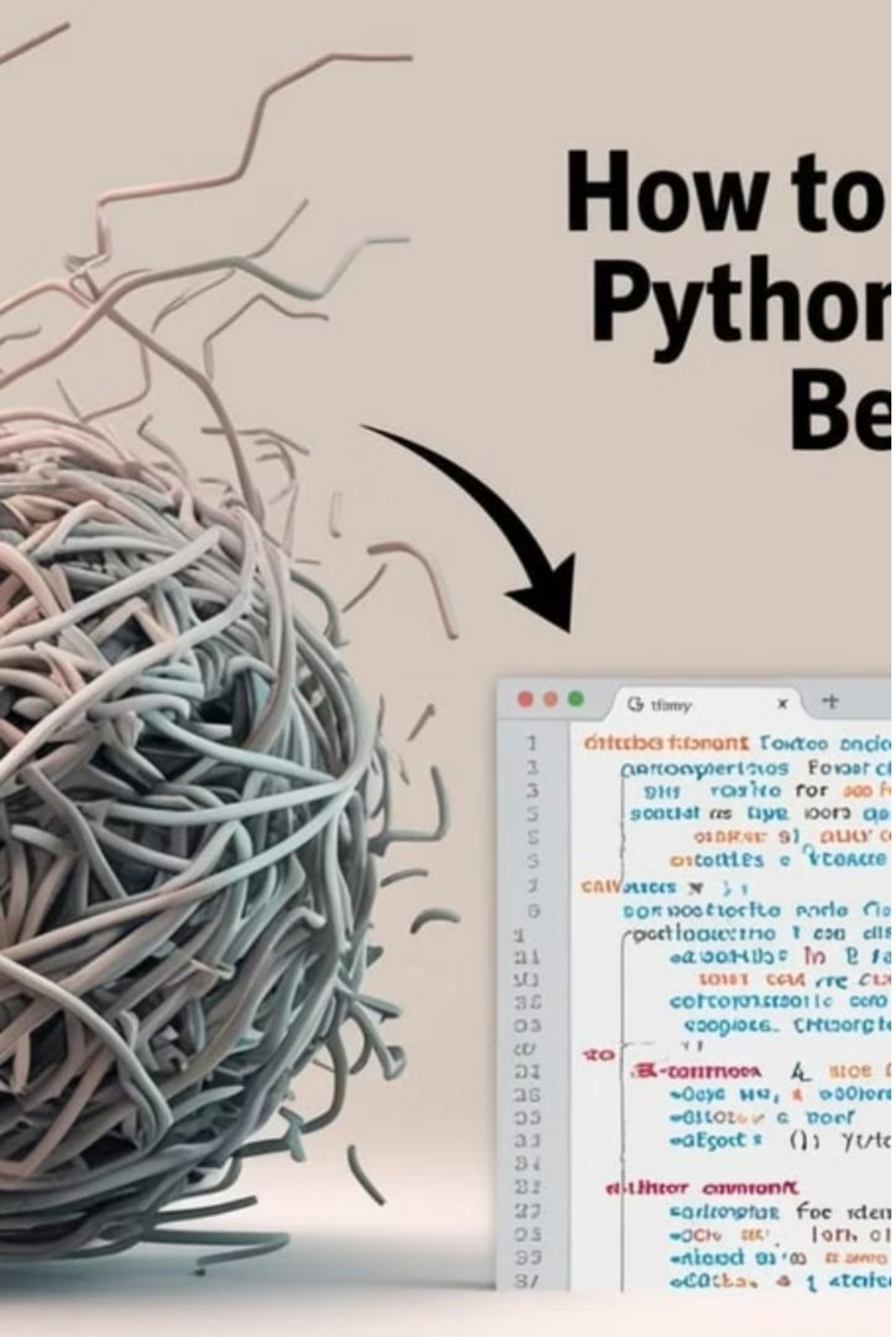
Automatización

Permite automatizar tareas repetitivas, ahorrando tiempo y reduciendo errores en procesos manuales.

Creación

Facilita la creación de páginas web, aplicaciones, juegos, análisis de datos y mucho más, expandiendo las posibilidades tecnológicas.





Principales características de de Python

Fácil de aprender

No necesitas poner llaves {}, ni punto y coma ;, ni elementos complicados para que funcione, lo que facilita su aprendizaje para principiantes.

Código limpio y legible legible

Usa espacios y sangrías en lugar de símbolos raros, obligando a escribir código ordenado, como una lista clara de instrucciones.

Muchas librerías

Cuenta con numerosas "cajas de herramientas" ya creadas para diferentes propósitos: gráficos, bases de datos, inteligencia artificial y más.

Ventajas adicionales de Python

Multiplataforma

Funciona en todos los sistemas operativos: Windows, Mac, Linux e incluso en la nube sin necesidad de instalación.

Gran comunidad

Millones de personas usan Python, lo que significa que hay abundante documentación y soporte disponible en internet.

Versatilidad

Se adapta a múltiples campos: desarrollo web, ciencia de datos, inteligencia artificial, automatización y muchos más.

Versiones de Python

Python 2

Es una versión antigua que ya no se usa ni se actualiza. Se recomienda evitarla para nuevos proyectos, aunque algunos sistemas legacy aún la utilizan.

Dejó de recibir soporte oficial en 2020, por lo que no recibe actualizaciones de seguridad ni nuevas funcionalidades.

Python 3

Es la versión actual que se sigue actualizando regularmente. Es la que debes usar siempre para nuevos proyectos y aprendizaje.

La versión más usada actualmente es Python 3.10 o superior, que incluye mejoras significativas en rendimiento y funcionalidades.

Propósito de Python

Python fue diseñado con un propósito claro:

“Hacer que programar sea fácil, legible y accesible para todos.”

Eso significa que:

•Puedes **leer un programa en Python casi como si leyeras una receta de cocina**.

•No necesitas saber cosas súper técnicas para empezar.

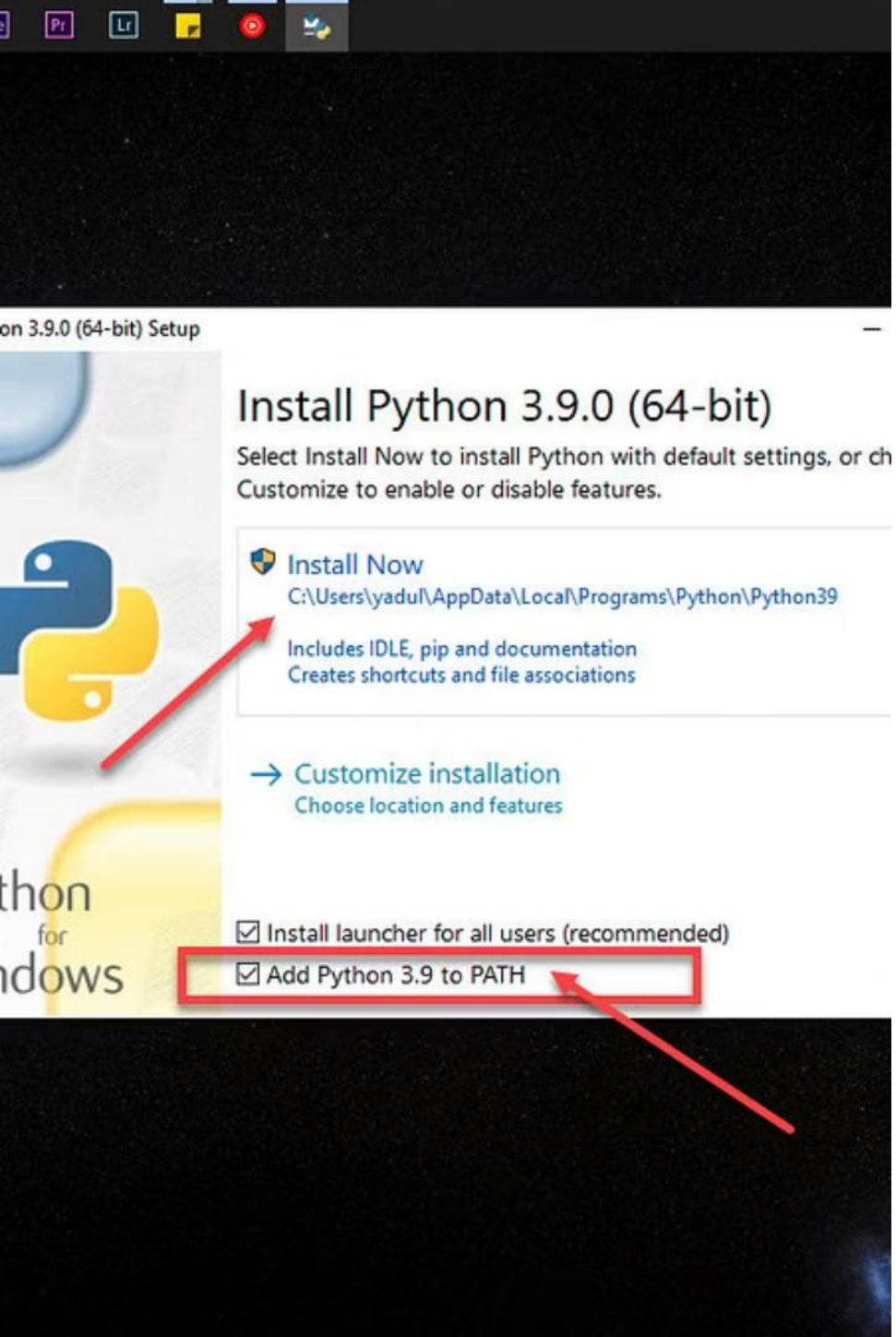
•Sirve para muchos usos distintos (es multipropósito).

🔧 **Ejemplos del mundo real:**

•Un robot en una fábrica podría estar programado en Python.

•Un bot que te responde en WhatsApp, puede usar Python.

•Un programa que analiza datos del clima, también puede estar hecho en Python.



Instalación de Python

Descargar Python

Visita el sitio oficial https://www.python.org/downloads/ y descarga la última versión estable de Python 3.

Instalar correctamente

En Windows, es crucial marcar la casilla "Add Python to PATH" durante la instalación para poder usar Python desde cualquier parte del sistema.

Verificar instalación

Abre una terminal o consola y escribe "python --version" para comprobar que Python se ha instalado correctamente. Deberías ver algo como "Python 3.10.11".

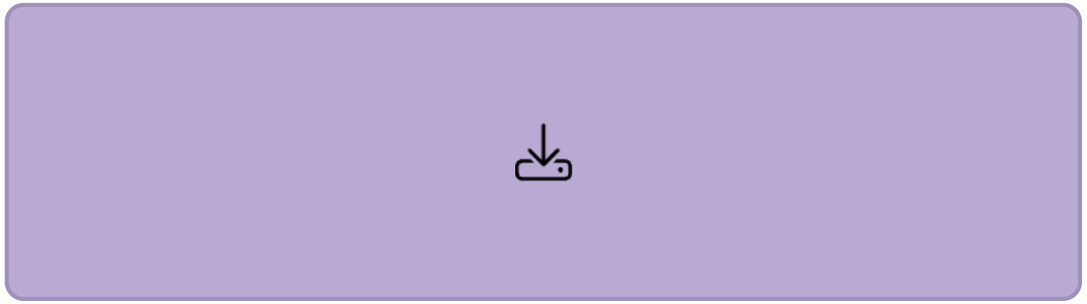
Entornos de trabajo: Anaconda

Todo en uno

Incluye Python, Jupyter Notebooks y librerías científicas

Ideal para ciencia de datos

Perfecto para machine learning y análisis



Fácil instalación

Disponible en

https://www.anaconda.com/products/distribution

Entornos de trabajo: Jupyter y Google Colab

Jupyter Notebooks

Es como un cuaderno digital interactivo donde puedes escribir código y texto explicativo en el mismo documento. Ideal para aprender paso a paso y visualizar resultados inmediatamente.

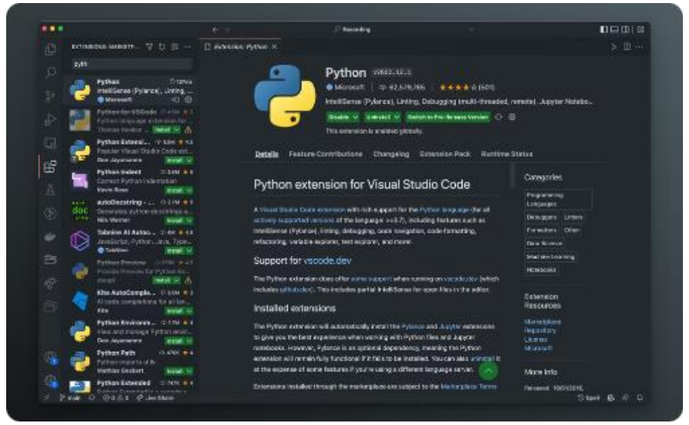
Puedes instalarlo con Anaconda o usarlo directamente desde su sitio web: **https://jupyter.org**

Google Colab

Es un Jupyter Notebook en la nube. No necesitas instalar nada, solo un navegador web y una cuenta de Google. Perfecto para principiantes y clases en línea.

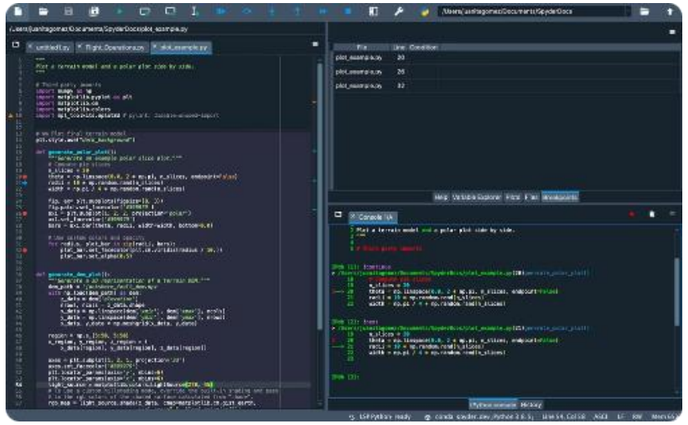
Accesible gratuitamente desde:
https://colab.research.google.com/

Entornos de trabajo: VS Code y Spyder



Visual Studio Code

Un editor de código liviano y potente que sirve para muchos lenguajes. Ideal para proyectos más grandes y para aprender a programar de forma profesional.
Necesita la instalación de Python y una extensión específica.



Spyder

Un entorno incluido en Anaconda, similar a MATLAB. Perfecto para personas que vienen de usar MATLAB o R, y para proyectos científicos o numéricos que requieren visualización de datos.

¿Cuál elegir?

La elección depende de tus necesidades: para empezar sin instalaciones, Google Colab; para tener todo listo con un clic, Anaconda; para programación profesional, VS Code; para análisis de datos, Spyder o Jupyter.

¿Qué es programar?

Entrada

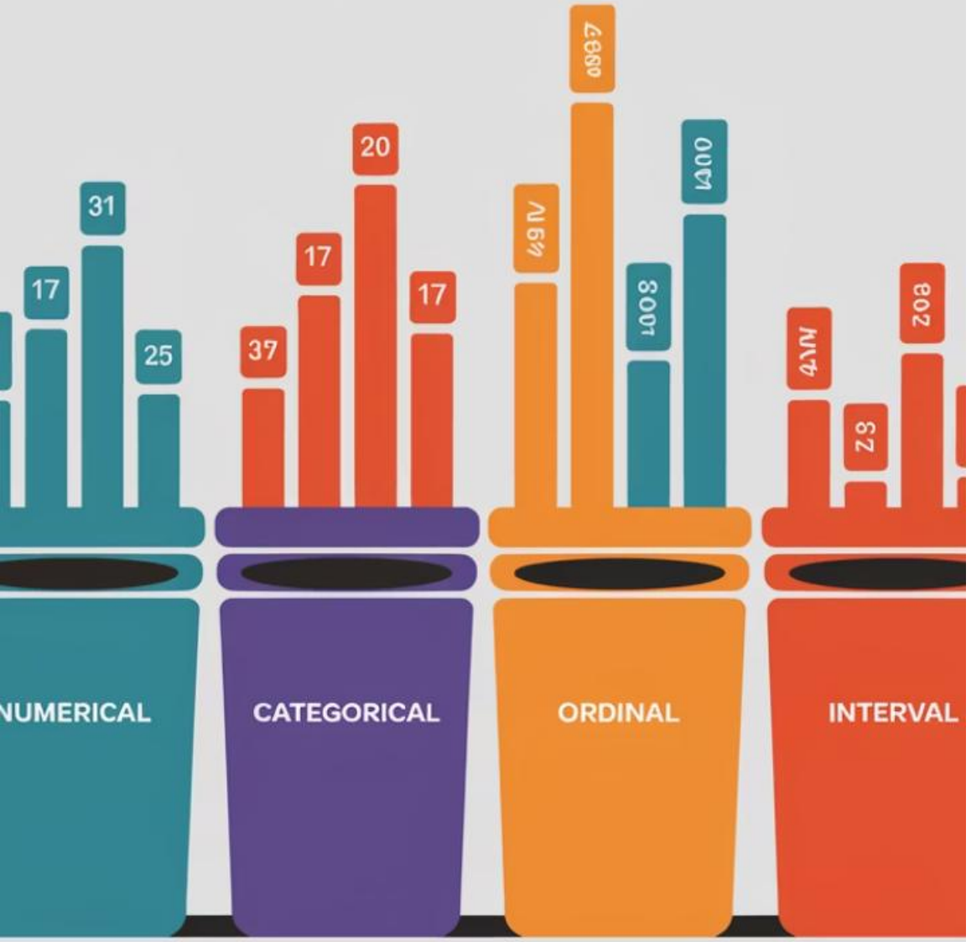
El usuario proporciona datos por teclado, archivos o sensores

Proceso

El programa realiza cálculos o toma decisiones con esos datos

Salida

Se muestra el resultado al usuario por pantalla o archivo



Variables y Tipos de Datos

Una variable es un espacio en la memoria que sirve para guardar información que usaremos en el programa. Es como una caja con nombre donde puedes almacenar diferentes tipos de datos.

Tipo de dato	Ejemplo	Uso común
Entero (int)	30	Contar, edades, cantidades
Decimal (float)	3.14	Medidas, porcentajes, precios
Cadena (str)	"Hola"	Nombres, mensajes, palabras
Booleano (bool)	True o False	Decisiones lógicas (sí o no)
Complejo (complex)	2 + 3j	Matemática avanzada, ingeniería

Variables y Tipos de Datos

```
nombre = "Sofía"
edad = 25
```

```
# Ejemplos de cada tipo
entero = 10
decimal = 3.14
texto = "Hola mundo"
es_estudiante = True
numero_complejo = 2 + 5j
```

Conversiones de Tipo

Texto (str)

Cadenas de caracteres como "Hola" o "42"

Conversión

Transformación mediante int(), float(), str()

Número (int/float)

Valores numéricos como 42 o 3.14

La conversión de tipo es útil para mostrar datos de forma legible, realizar operaciones cuando los tipos no coinciden y procesar entradas de usuario, que siempre vienen como texto. Por ejemplo, si necesitas sumar dos números ingresados por el usuario, deberás convertirlos de texto a números primero.

```python
# De string a entero
edad_texto = "30"
edad = int(edad_texto)

# De número a texto
numero = 10
texto = str(numero)

# De número a decimal
entero = 5
decimal = float(entero)
```

Operadores Aritméticos

Operadores básicos

- Suma: + (a + b)
- Resta: - (a - b)
- Multiplicación: * (a * b)
- División: / (a / b)

Operadores avanzados

- División entera: // (a // b)
- Módulo (resto): % (a % b)
- Potencia: ** (a ** b)
- Asignación: = (a = b)

Operadores combinados

- Suma y asigna: += (a += b)
- Resta y asigna: -= (a -= b)
- Multiplica y asigna: *= (a *= b)
- Divide y asigna: /= (a /= b)

```python
a = 10
b = 3

print(a + b)  # Suma → 13
print(a - b)  # Resta → 7
print(a * b)  # Multiplicación → 30
print(a / b)  # División → 3.333...
print(a % b)  # Módulo (resto) → 1
print(a ** b) # Potencia → 1000
print(a // b) # División entera → 3
```

Operadores y Expresiones Lógicas

Operadores de comparación

- Igual que: == (a == b)
- Distinto que: != (a != b)
- Mayor que: > (a > b)
- Menor que: < (a < b)
- Mayor o igual que: >= (a >= b)
- Menor o igual que: <= (a <= b)

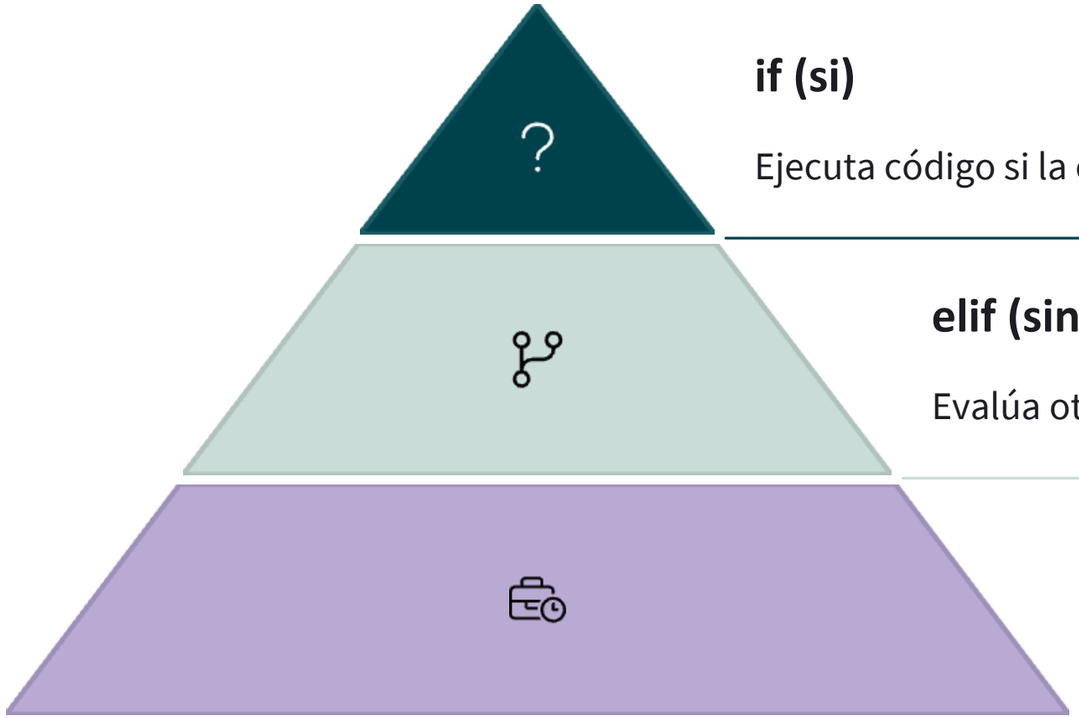
```
x = 5
y = 10

print(x > y)     # False
print(x < y)     # True
print(x == 5)    # True
```

Operadores lógicos

- AND: and (a and b) - Verdadero si ambas condiciones son verdaderas
- OR: or (a or b) - Verdadero si al menos una condición es verdadera
- NOT: not (not a) - Niega la condición, invirtiendo su valor

```
print(x > 3 and y < 15)   # True
print(x < 3 or y > 9)     # True
print(not(x == 5))        # False
```

Sentencias Condicionales



if (si)

Ejecuta código si la condición es verdadera

elif (sino si)

Evalúa otra condición si la anterior es falsa

else (sino)

Ejecuta código si todas las condiciones son falsas

El control de flujo permite que el programa tome decisiones y ejecute distintos caminos según las condiciones evaluadas. Una bifurcación ocurre cuando el código elige entre varias opciones basándose en una condición. Solo se ejecuta el bloque que cumple la condición especificada.

Sentencias Condicionales

```
edad = 20

if edad < 18:
    print("Eres menor de edad")
elif edad == 18:
    print("Tienes justo 18 años")
else:
    print("Eres mayor de edad")
```

Impresión y Entrada de Datos

```
nombre = "Miguel"
print("Hola", nombre)
```

```
nombre = input("¿Cómo te llamas? ")
edad = int(input("¿Qué edad tienes? "))

print("Hola", nombre, ", tienes", edad, "años.")
```

La función **print()** permite mostrar información en la pantalla, mientras que **input()** recibe datos desde el teclado. Es importante recordar que todo lo que entra por input() es texto (string), por lo que se debe convertir con int() o float() si se requieren números para operaciones matemáticas.

Creación y Ejecución de Scripts

¿Qué es un script?

Es un archivo de texto (.py) que **contiene código Python** que se puede ejecutar.

¿Cómo lo creo y ejecuto?

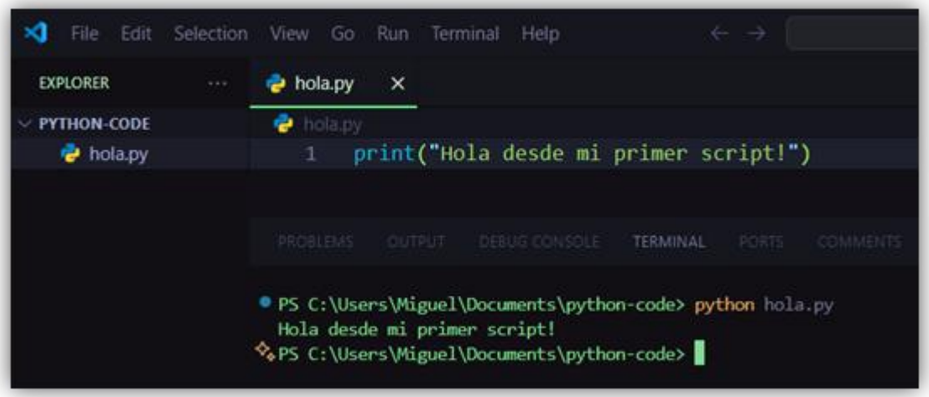
1.Abre un editor (VS Code, por ejemplo).
2.Crea un archivo llamado hola.py.
3.Escribe el siguiente código:

```
print("Hola desde mi primer script!")
```

Abre una terminal y ejecuta:

```
python hola.py
```

El script ejecuta todo lo que está escrito, línea por línea.



Actividad Práctica Guiada: Tu Primer Programa en Python

Objetivo:

Crear un programa en Python que solicite al usuario su nombre, edad y país, y luego imprima un mensaje personalizado utilizando los datos ingresados.

Paso a paso:

Paso 1: Crear el archivo

1.Abre tu editor de código favorito (VS Code o Google Colab).
2.Crea un archivo llamado: mi_primer_programa.py

Actividad Práctica Guiada: Tu Primer Programa en Python

Paso 2: Escribir el código

Escribe el siguiente código en el archivo:

```
print("¡Bienvenido al programa interactivo!")
nombre = input("¿Cómo te llamas? ")
edad = input("¿Cuántos años tienes? ")
pais = input("¿De qué país eres? ")

print("\nResumen de tu información:")
print(f"Nombre: {nombre}")
print(f"Edad: {edad}")
print(f"País: {pais}")

print(f"\nHola {nombre}, tienes {edad} años y vienes de {pais}. ¡Genial!")
```

Actividad Práctica Guiada: Tu Primer Programa en Python

Paso 3: Ejecutar el programa

1.Abre la terminal o consola.
2.Ejecuta el archivo:

```
python mi_primer_programa.py
```

3.Ingresa tus datos cuando el programa lo solicite.

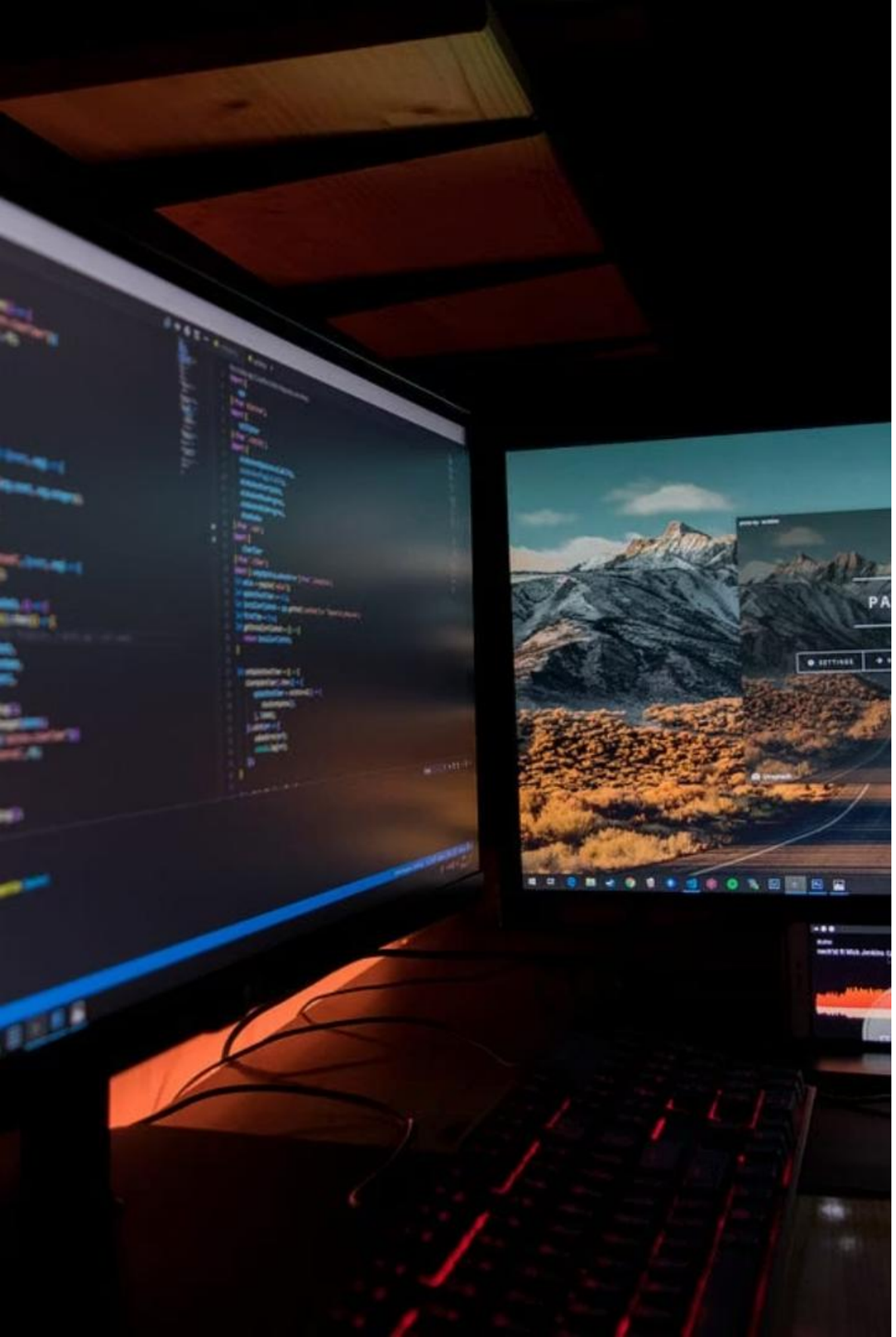
Paso 4: Reflexiona

•¿Qué sucede si ingresas un número en el campo de nombre?
•¿Y si dejas un campo vacío?
•¿Qué podrías agregarle al programa para que sea más interactivo?

Recursos Adicionales

🎥 https://www.youtube.com/watch?v=Kp4Mvapo5kc

Este video ofrece una visión práctica y accesible de los conceptos básicos de Python para quienes están comenzando a programar.



Reflexiones sobre Python y Python y Entornos de Desarrollo

1. ¿Qué ventajas encuentras en la sintaxis simple y legible de Python para comenzar a programar?

2. ¿Cómo podrías aplicar los conocimientos de variables, operadores y estructuras condicionales en un pequeño proyecto real o del día a día?

3. ¿Cuál de los entornos de desarrollo que exploraste (Anaconda, VS Code, Colab, etc.) te pareció más cómodo y por qué?