Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

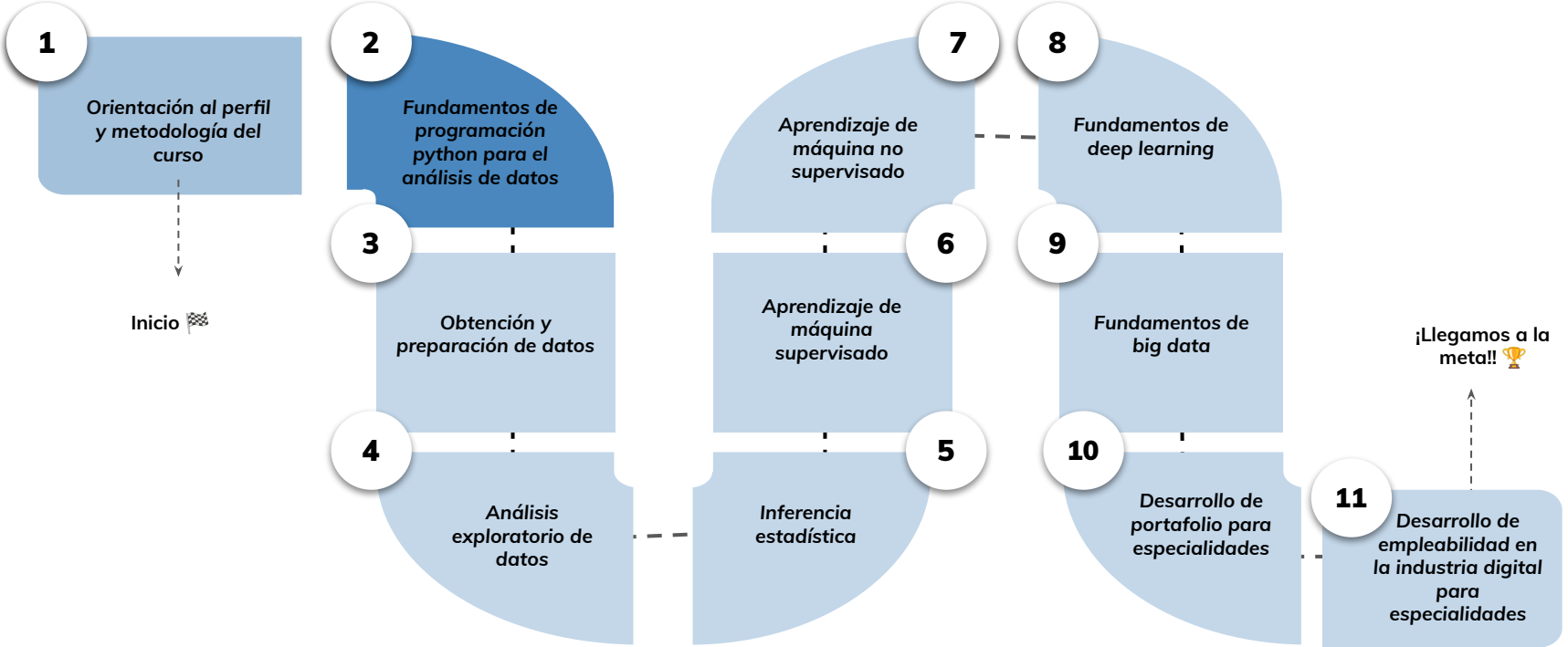
Sentencias básicas del lenguaje python- Parte 1

Aprendizaje Esperado 2: Aplicar el concepto de variable, tipos de dato fundamentales y expresiones aritméticas utilizando el lenguaje python para la creación de una rutina de baja complejidad.

alkemy

Hoja de ruta

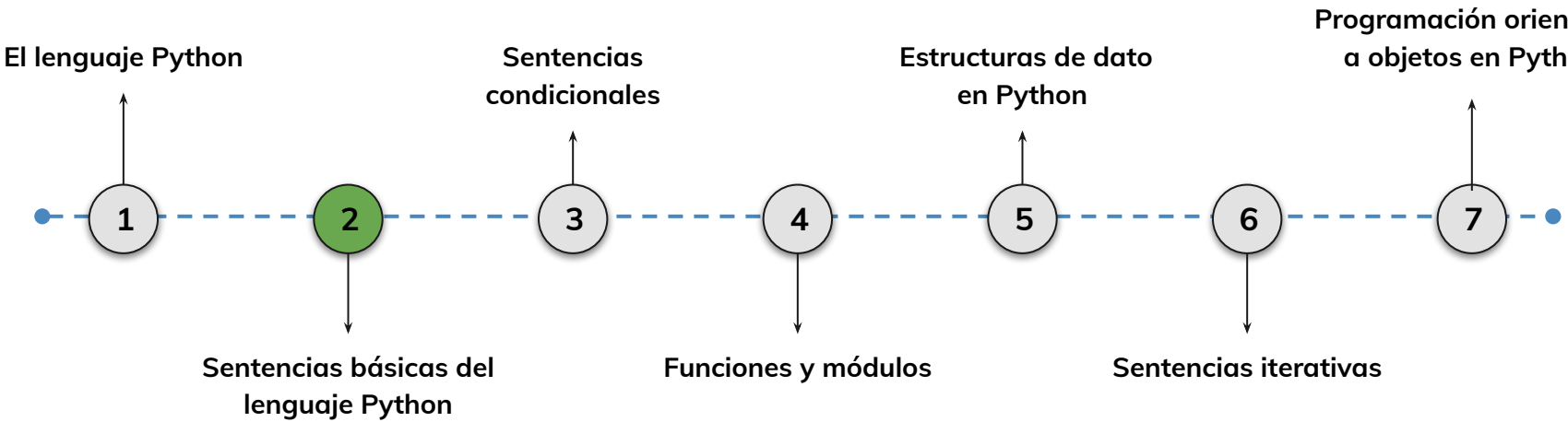
¿Cuáles skills conforman el programa? **Fundamentos de Ciencia de Datos**



alkemy

Roadmap de lecciones

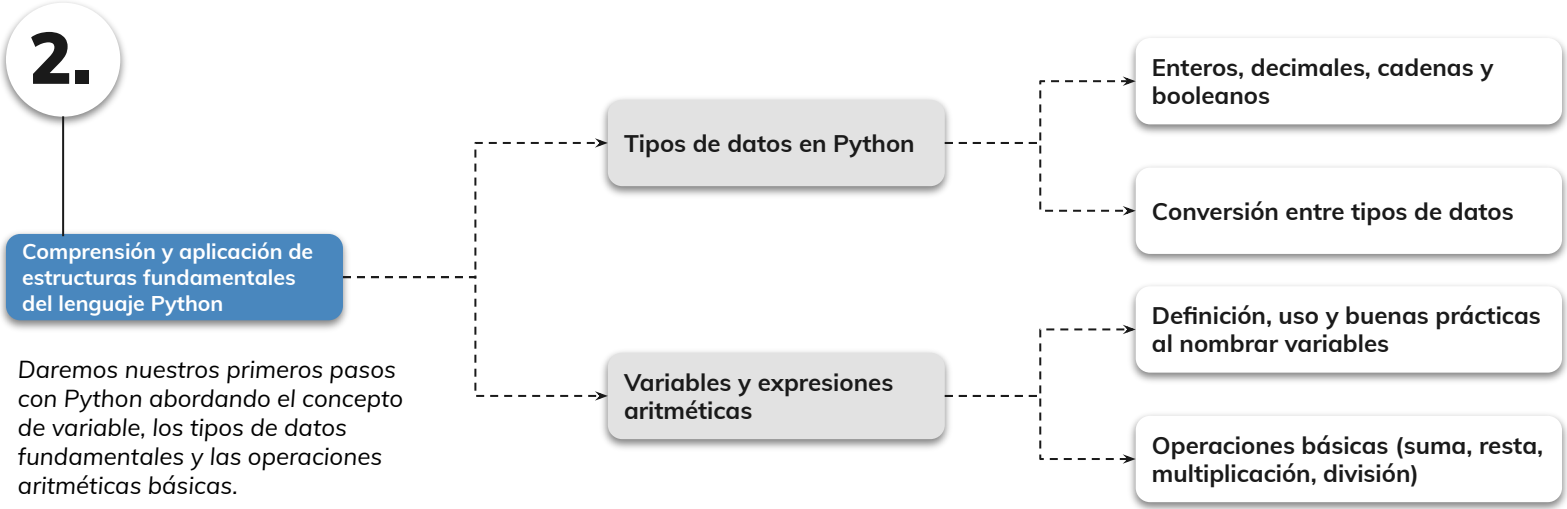
*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?



alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Comprender qué es una variable y cómo se utiliza en Python
- Identificar los tipos de datos fundamentales y su utilidad
- Utilizar operadores aritméticos para construir expresiones
- Aplicar conversiones entre tipos de datos en rutinas simples
- Crear una rutina funcional usando variables y cálculos básicos

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → *Comprendimos las diferencias entre Python 2 y Python 3, y sus implicancias prácticas.*
- → *Exploramos las novedades clave introducidas en versiones recientes de Python 3.x.*
- → *Conocimos herramientas como Anaconda, Jupyter, Google Colab, Spyder y VS Code.*
- → *Aprendimos a elegir el entorno de trabajo más adecuado según el tipo de proyecto o problema.*

Sentencias básicas del lenguaje python

alkemy

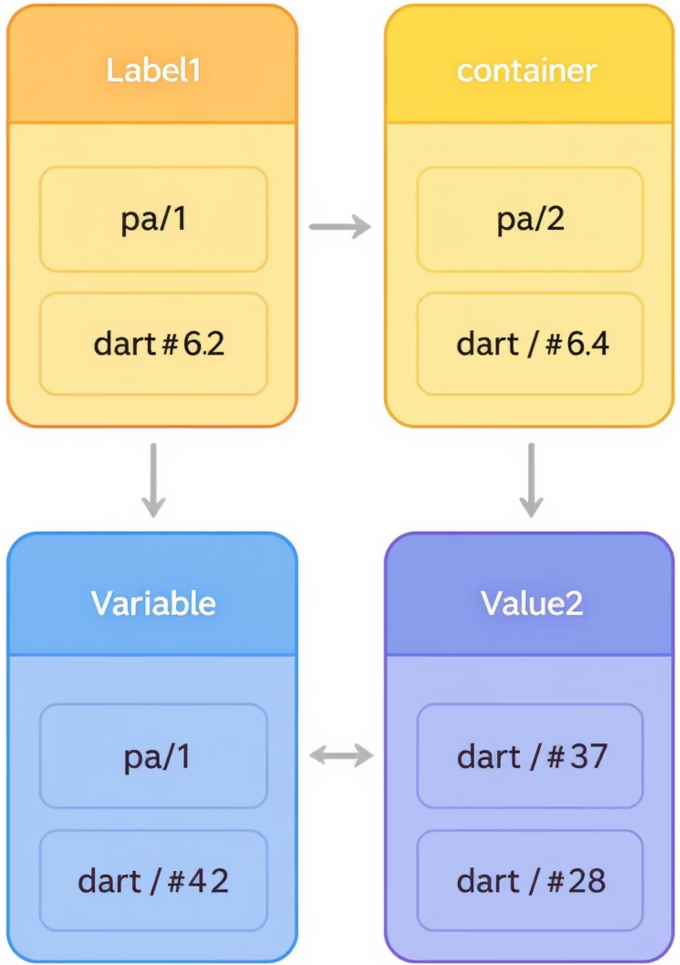
Sentencias básicas del lenguaje Python

Python es un lenguaje de programación conocido por su simplicidad y claridad, lo que lo hace ideal tanto para principiantes como para programadores experimentados. Al aprender las sentencias básicas de Python, los nuevos desarrolladores adquieren las habilidades fundamentales para manipular datos, realizar cálculos, manejar el flujo de datos en un programa y obtener interacción con el usuario.

alkemy

Variables

Las variables son uno de los conceptos fundamentales en la programación con Python. Permiten almacenar y manipular datos de manera eficiente durante la ejecución de un programa.



< >

Definición y Uso de Variables

En Python, una variable es un contenedor para almacenar datos que pueden utilizarse y manipularse en un programa. Las variables permiten asignar valores a un nombre específico, lo que facilita su referencia y manipulación en operaciones posteriores.

A diferencia de otros lenguajes, Python no requiere declarar el tipo de una variable antes de usarla, ya que es un lenguaje de tipado dinámico. Esto significa que el tipo de datos de una variable se determina automáticamente cuando se le asigna un valor.

< >

Asignación de Variables

Para asignar un valor a una variable, se utiliza el operador de asignación =, de modo que el nombre de la variable se coloca a la izquierda del signo igual y el valor asignado a la derecha. Por ejemplo, edad = 25 asigna el valor 25 a la variable edad.

Las variables pueden almacenar diferentes tipos de datos, como números, cadenas de caracteres y valores booleanos, y se pueden cambiar o reasignar en cualquier momento.

Uso de Variables

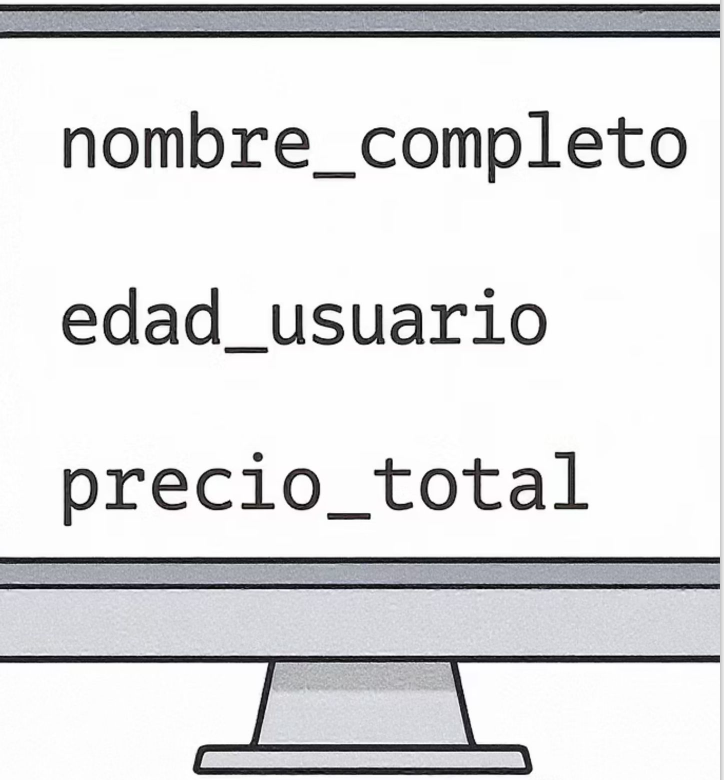
Las variables pueden usarse posteriormente en cualquier operación o impresión, como se muestra en el siguiente ejemplo: *Una vez que una variable ha sido definida, puede ser utilizada en múltiples partes del programa, lo que facilita la reutilización de datos.*

< >

Buenas Prácticas al Nombrar Variables

Elegir nombres descriptivos y coherentes para las variables es una buena práctica que facilita la lectura y el mantenimiento del código. En Python, es común utilizar nombres de variables en minúsculas y separados por guiones bajos (_) cuando el nombre contiene más de una palabra, como en edad_usuario.

Evitar el uso de nombres de variables muy cortos o genéricos ayuda a evitar confusiones y errores en el código, especialmente en proyectos grandes o colaborativos.



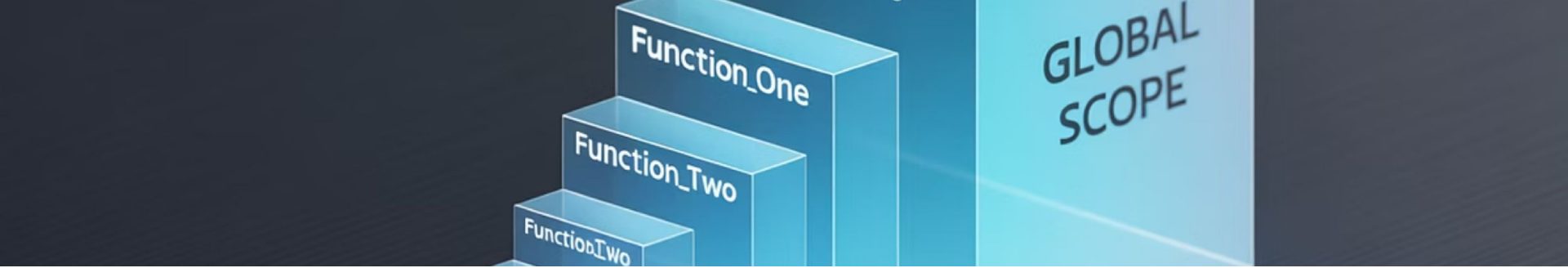
Ejemplos de Buenos Nombres de Variables

Los nombres descriptivos como "nombre_completo", "edad_usuario" y "precio_total" hacen que el código sea más legible y comprensible para otros programadores.

Variables Dinámicas en Python

Como Python es un lenguaje de tipado dinámico, una variable puede cambiar de tipo durante la ejecución del programa. Por ejemplo, una variable que inicialmente contiene un número puede cambiarse para almacenar una cadena de texto más adelante.

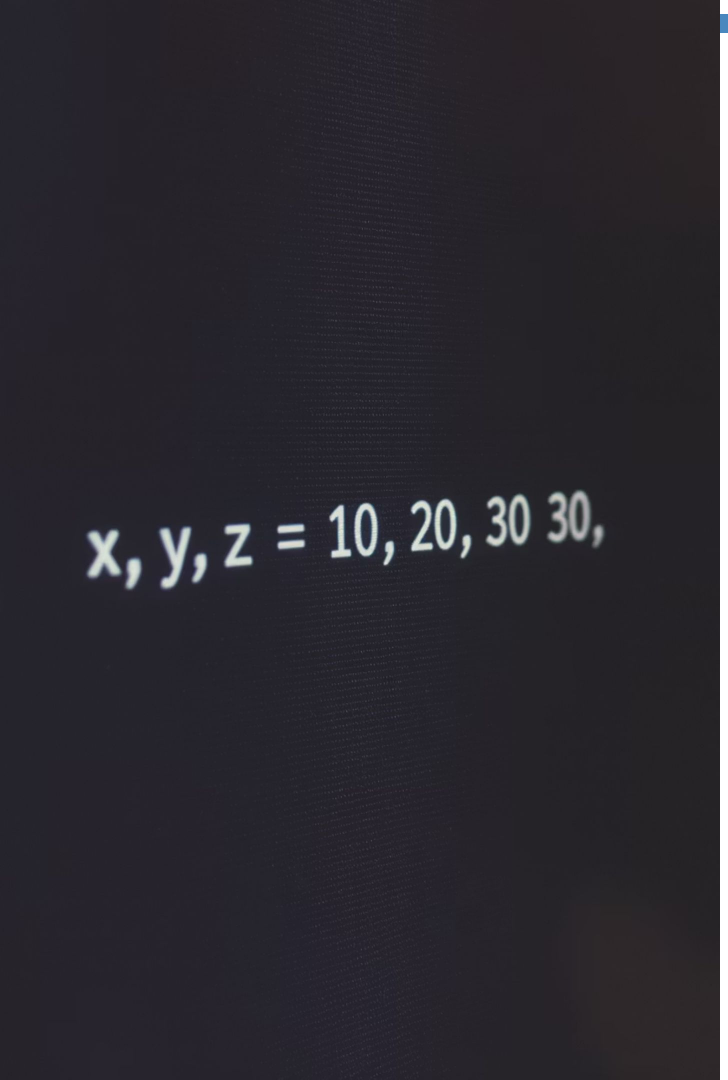
Aunque esta flexibilidad facilita el desarrollo rápido, se debe tener cuidado al cambiar el tipo de una variable para evitar errores en el programa.



Scope o Alcance de las Variables

El alcance de una variable se refiere a la parte del programa donde la variable es accesible. En Python, una variable definida dentro de una función es local y solo puede utilizarse dentro de esa función.

Por otro lado, una variable definida fuera de una función tiene un alcance global y puede accederse desde cualquier parte del programa. Es importante gestionar el alcance de las variables para evitar conflictos y asegurar la coherencia de los datos.



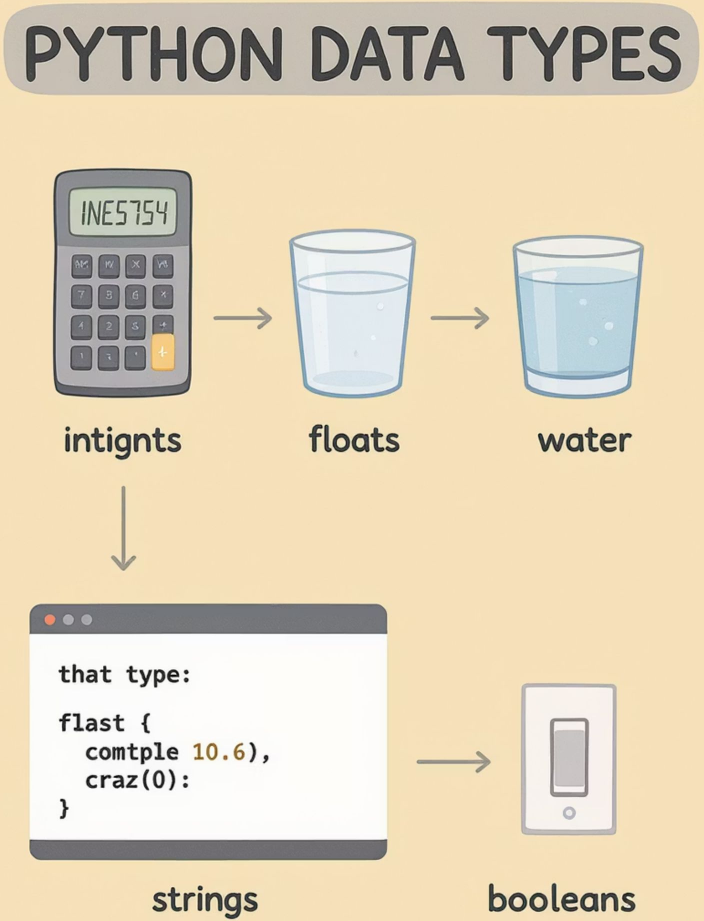
Declaración Múltiple de Variables

Python permite asignar valores a varias variables en una sola línea, lo que puede ser útil en situaciones donde se deben definir múltiples variables al mismo tiempo. Por ejemplo, x, y, z = 1, 2, 3 asigna los valores 1, 2 y 3 a x, y y z respectivamente.

Esta sintaxis hace que el código sea más compacto y legible cuando se trabaja con múltiples variables de manera simultánea.

Tipos de Dato Fundamentales

Python ofrece diversos tipos de datos fundamentales que permiten representar diferentes clases de información en los programas.



< >

Enteros

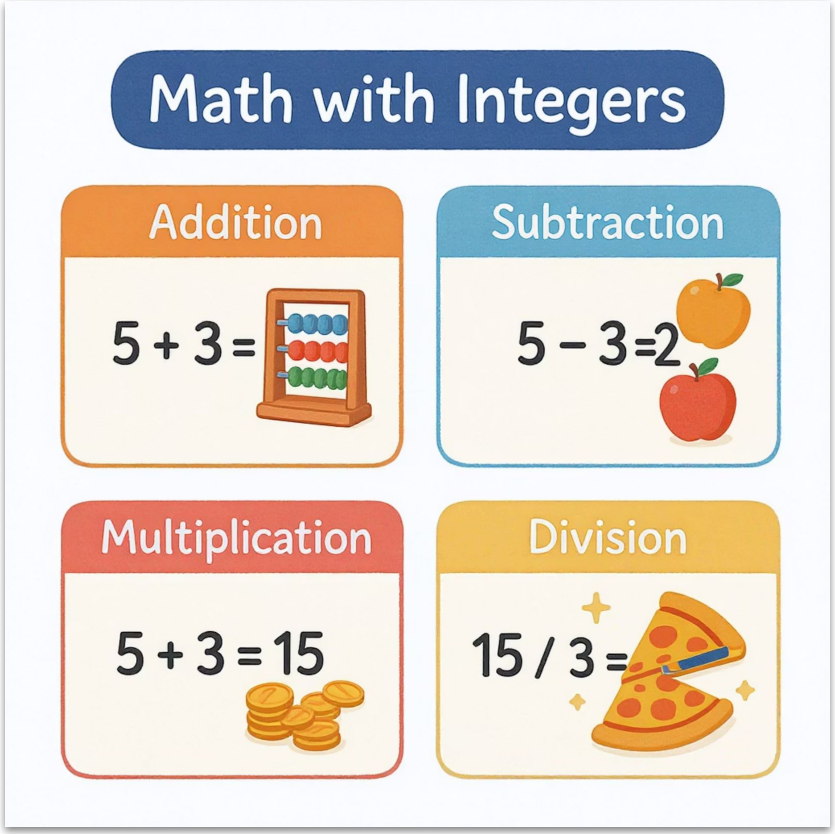
Los números enteros son uno de los tipos de datos más básicos en Python, representados por el tipo int. Los enteros permiten almacenar valores numéricos sin decimales, como -5, 0, y 100.

Los enteros en Python tienen un rango amplio y no están limitados en tamaño, lo que permite trabajar con números extremadamente grandes sin necesidad de tipos adicionales. Este tipo de dato es útil en cálculos matemáticos básicos y en situaciones donde se necesita contar o iterar en bucles.

Operaciones con Enteros

Estas operaciones son fundamentales para realizar cálculos en programas Python.

Los enteros en Python permiten realizar diversas operaciones matemáticas de manera sencilla:

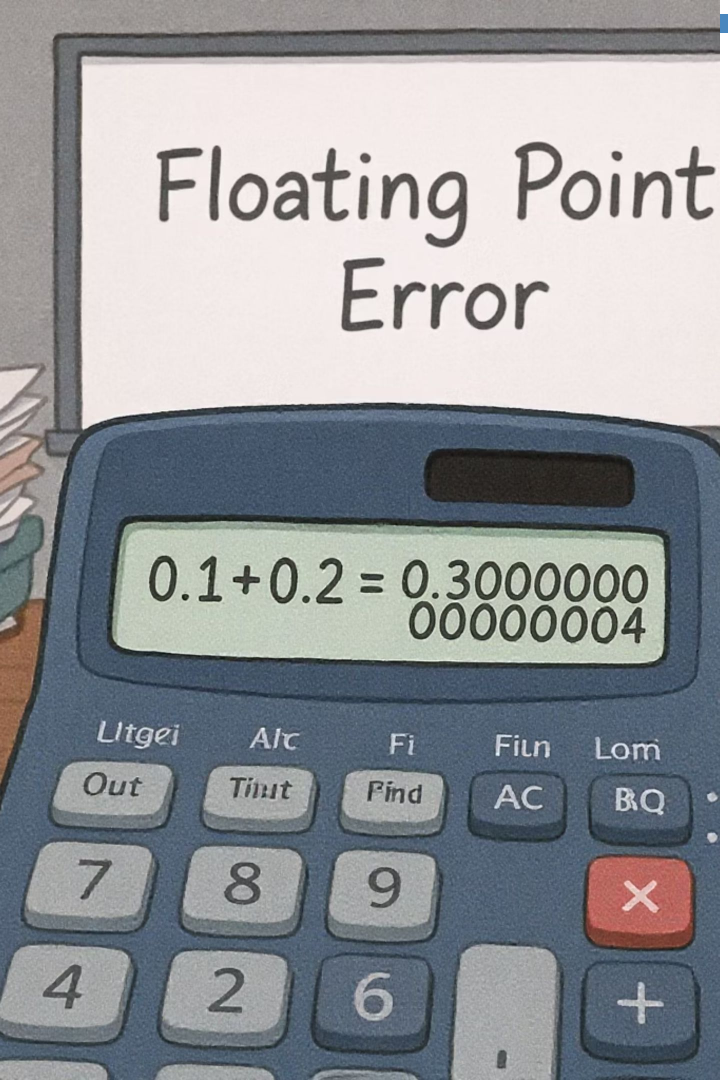


< >

Decimales

Los números decimales, o de punto flotante, están representados por el tipo float en Python y permiten almacenar números con decimales, como 3.14, -2.5, y 0.001.

Los float son útiles cuando se requiere precisión en cálculos que involucran fracciones o porcentajes. Python también ofrece el módulo decimal para manejar decimales con mayor precisión, lo cual es útil en aplicaciones financieras o científicas donde la precisión es crucial.



Uso de Decimales

Los números de punto flotante permiten realizar cálculos con precisión decimal:

Es importante tener en cuenta que los cálculos con números de punto flotante pueden tener pequeños errores de precisión debido a la forma en que se representan internamente.

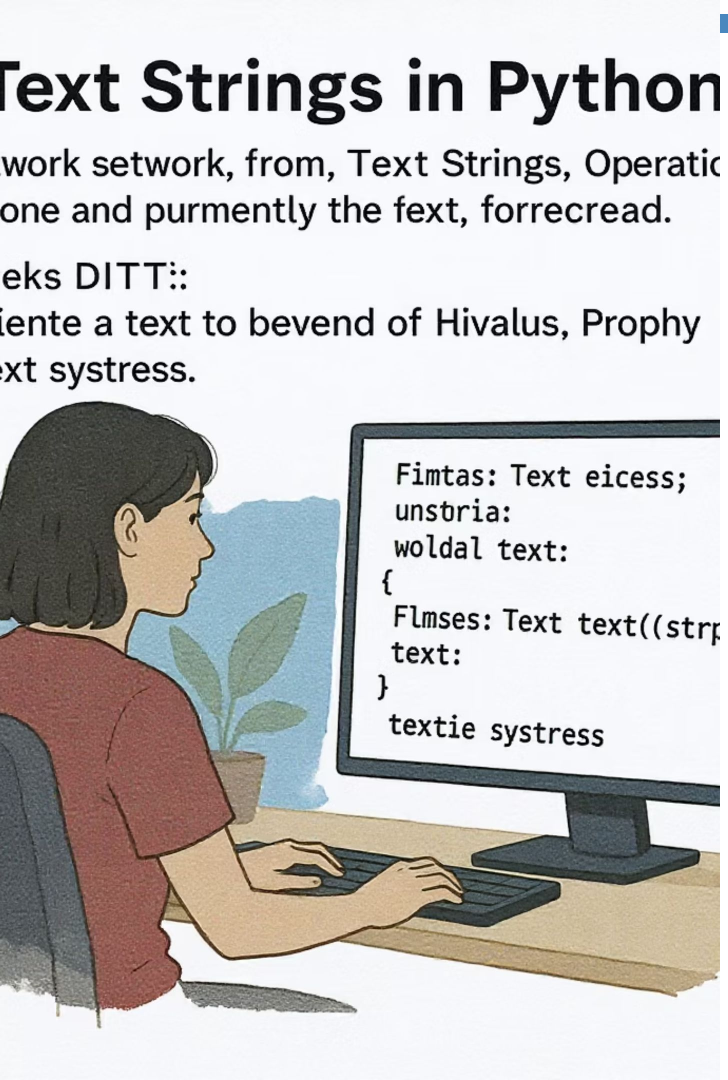
alkemy

< >

Cadena de Caracteres

Las cadenas de caracteres, o str, representan secuencias de texto y son ampliamente utilizadas para almacenar y manipular datos textuales. En Python, las cadenas pueden definirse entre comillas simples ('texto') o dobles ("texto"), y se pueden concatenar, dividir y manipular con facilidad.

Python ofrece una serie de métodos para trabajar con cadenas, como upper(), lower(), y replace(), que permiten transformar y modificar el texto de forma eficiente.



Uso de Cadenas

Las cadenas de texto son versátiles y permiten diversas operaciones:

La manipulación de cadenas es fundamental para el procesamiento de texto en Python.

alkemy

< >

Booleanos

El tipo de dato booleano (bool) representa valores de verdad y puede ser True o False. Los booleanos son fundamentales en Python, especialmente en estructuras de control como las sentencias if y while, ya que permiten tomar decisiones basadas en condiciones.

El valor booleano es el resultado de expresiones lógicas, como comparaciones entre números o cadenas de caracteres.

< >

Comparación y Valor Booleano

Los valores booleanos son el resultado de operaciones de comparación:

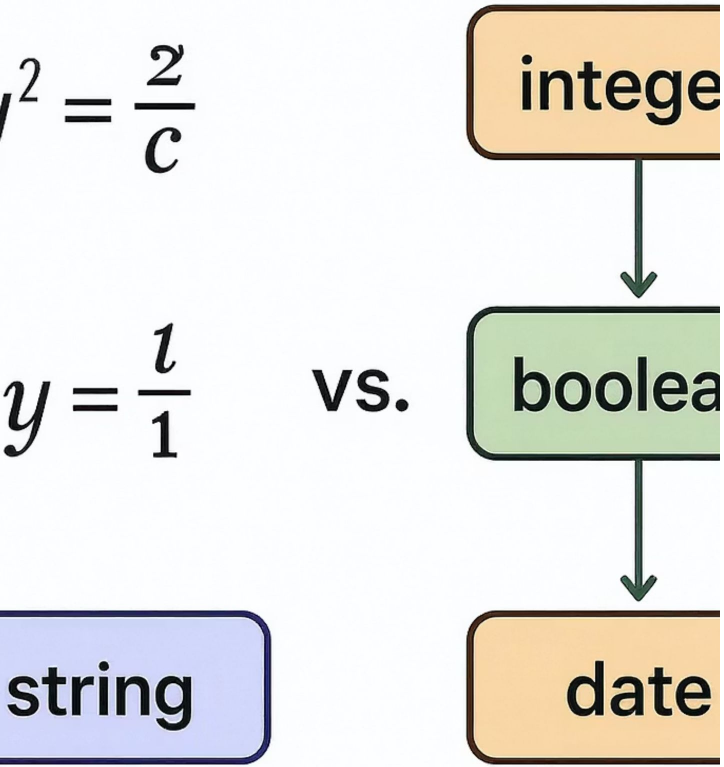
Estos valores son esenciales para controlar el flujo de ejecución en un programa.

alkemy

Conversión entre Tipos de Datos

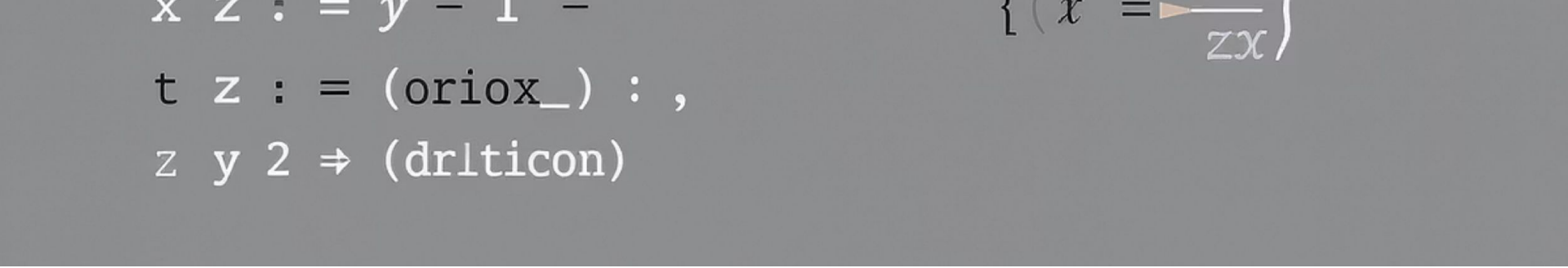
Python permite convertir entre tipos de datos utilizando funciones de conversión como int(), float(), str(), y bool(). La conversión es útil cuando se necesita adaptar un tipo de datos específico para una operación o para mejorar la coherencia en el programa.

Por ejemplo, si se necesita concatenar un número con una cadena, se puede convertir el número a cadena utilizando str(numero). Esta flexibilidad en la conversión facilita la manipulación de datos en Python.



Ejemplos de Conversión de Tipos

Las conversiones de tipo son operaciones comunes que permiten adaptar los datos según las necesidades del programa.



Expresiones Aritméticas

Las expresiones aritméticas en Python permiten realizar cálculos matemáticos de manera eficiente y clara.

< >

Operadores Aritméticos Básicos

Python permite realizar operaciones matemáticas básicas mediante operadores aritméticos como + para la suma, - para la resta, * para la multiplicación, y / para la división.

Estos operadores se pueden usar con variables y valores constantes para crear expresiones aritméticas que facilitan la manipulación de datos numéricos. Las operaciones se ejecutan siguiendo el orden de operaciones, lo que asegura la coherencia de los cálculos.

< >

Ejemplos de Operaciones Aritméticas

Estas operaciones básicas son el fundamento de los cálculos matemáticos en Python.

```
a = 10
b = 3
suma = a + b
resta = a - b
multiplicacion = a * b
division = a / b
print("Suma:", suma, "Resta:", resta, "Multiplicación:", multiplicacion
```

< >

Operador de Módulo y División Entera

Además de los operadores básicos, Python incluye el operador de módulo (%), que devuelve el residuo de una división. Esto es útil en casos donde se necesita determinar si un número es divisible por otro, como en la verificación de números pares o impares.

Python también permite realizar divisiones enteras utilizando //, que devuelve solo la parte entera del resultado, sin el decimal. Estas operaciones son útiles en situaciones que requieren precisión en el manejo de divisores y múltiplos.

< >

Uso de Módulo y División Entera

El operador de módulo y la división entera son especialmente útiles en algoritmos que requieren trabajar con divisibilidad o partes enteras de números.

```
dividendo = 10
divisor = 3
residuo = dividendo % divisor
division_entera = dividendo // divisor
print("Residuo:", residuo)  # Salida: Residuo: 1
print("División entera:", division_entera)  # Salida: División entera: 3
```

< >

Operador de Exponenciación

El operador ** en Python permite calcular potencias, como 3**2 para elevar 3 al cuadrado. La exponenciación es común en problemas matemáticos y científicos, y Python permite realizar esta operación con alta precisión.

Además, la biblioteca matemática (math) de Python ofrece funciones avanzadas, como el cálculo de raíces y logaritmos, que permiten realizar operaciones matemáticas complejas de manera eficiente.

< >

Ejemplos de Exponenciación

La exponenciación es fundamental para cálculos científicos y matemáticos avanzados.

```
base = 2
exponente = 3
potencia = base ** exponente
print("Potencia:", potencia)  # Salida: Potencia: 8
```

Live Coding

¿En qué consistirá la Demo?

Construiremos una mini rutina interactiva combinando distintos tipos de datos, operaciones y estructuras de Python. Exploraremos cómo usar print(), type(), listas, diccionarios y operaciones básicas para modelar información del mundo real.

1. *Crear variables: nombre, edad, altura, es_estudiante*
2. *Imprimir valores y tipos con print() y type()*
3. *Calcular promedio de edad entre tres personas*
4. *Mostrar diferencia entre int y float al dividir*
5. *Construir lista de productos y diccionario de precios*
6. *Calcular precio total de un carrito*
7. *Usar set() para obtener elementos únicos*
8. *Evaluar una condición booleana con == o !=*

Tiempo: 25 Minutos

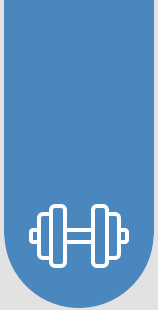
#Momentode Preguntas...

- *¿Qué diferencia hay entre una variable y un valor literal?*
- *¿Cuándo usamos int y cuándo float?*
- *¿Por qué se considera que las listas son mutables y las tuplas no?*
- *¿Qué tipo de dato usarías para representar una fecha, un nombre y una edad?*

Momento:

Time-out!

⏰5 -10 min.



Ejercicio N° 1

Mi primera rutina inteligente

Mi primera rutina inteligente

Contexto: 🙌

Estás comenzando a desarrollar un sistema básico para registrar actividad física diaria. Para ello, deberás trabajar con distintos tipos de datos y cálculos simples que podrían escalarse a un sistema más robusto.

Consigna: ✍️

Aplicar el concepto de variable, tipos de dato fundamentales y expresiones aritméticas utilizando el lenguaje Python para la creación de una rutina de baja complejidad.

Tiempo 🕛**:** *30 Minutos*

Mi primera rutina inteligente

Paso a paso: ⚙️

1. **Crear variables que representen:**
    - nombre de la persona (str)
    - edad (int)
    - altura en metros (float)
    - rutina completada (bool)
    - calorías consumidas por día (lista de 5 valores)
2. **Calcular e imprimir:**
    - promedio de calorías consumidas
    - diferencia entre el valor más alto y más bajo de la lista
    - resultado de multiplicar la altura por la edad
    - mensaje personalizado que indique si la rutina fue completada o no

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Identificamos los tipos de datos fundamentales en Python***
- ✓ ***Creamos variables y aprendimos buenas prácticas de nombrado***
- ✓ ***Aplicamos expresiones aritméticas en rutinas simples***
- ✓ ***Usamos conversiones de tipo entre int, float, str y bool***
- ✓ ***Creamos nuestra primera rutina de baja complejidad***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

¡Muchas gracias!

alkemy