📝 Apuntes de Programación: Variables, Tipos de Datos y Fundamentos

El código que has proporcionado cubre los fundamentos esenciales de Python: la declaración de variables, los tipos de datos, los operadores aritméticos, la entrada/salida de datos y las estructuras de control básicas.

1. Variables y Tipos de Datos (Data Types)

Una **variable** es un nombre que se usa para referirse a un valor almacenado en la memoria. Python es un lenguaje de tipado dinámico, lo que significa que no necesitas declarar el tipo de la variable antes de asignarle un valor.

Tipo de Dato	Propósito	Ejemplo en Código	Descripción
`int` (Entero)	Números sin parte decimal.	`edad = 56`	Almacena números enteros.
`str` (Cadena)	Secuencias de caracteres.	`texto = "Cincuenta seis"`	Siempre va entre comillas simples o dobles.
`float` (Flotante)	Números con parte decimal.	`precio = 54.4`	Usan el punto (`.`) como separador decimal.
`bool` (Booleano)	Valores lógicos.	`acertado = True`	Solo puede ser `True` o `False`.

A. Pseudo-Constantes

En Python no existen las constantes inmutables como tal. Por convención, se usan **nombres en MAYÚSCULAS y guiones bajos** para indicar a otros programadores que el valor no debe modificarse.

```
MESES_DEL_ANNO = 12 # Convención para indicar que no debe cambiar
```

2. Operadores Aritméticos

Los operadores son símbolos que realizan operaciones sobre variables y valores.

Operador	Nombre	Función	Ejemplo (5 y 2)	Resultado
`+`	Suma	Suma dos valores.	`5 + 2`	`7`
`-`	Resta	Resta dos valores.	`5 - 2`	`3`
`*`	Multiplicación	Multiplica dos valores.	`5 * 2`	`10`

`/`	División	Devuelve el resultado en **float**.	`5 / 2`	`2.5`
`//`	División Entera	Devuelve el **cociente** (parte entera de la división).	`5 // 2`	`2`
`%`	Módulo	Devuelve el **resto** de la división.	`5 % 2`	`1`

A. Prioridad de Operadores

Al igual que en matemáticas, las operaciones siguen un orden: **Paréntesis** primero, luego Multiplicación/División, y finalmente Suma/Resta.

```
nuevo = precio * (5 + edad) # Primero se resuelve el paréntesis (5 + edad)
```

3. Entrada/Salida y Conversión de Tipos (Casting)

A. Función `print()`

La función `print()` permite mostrar variables y textos, usando comas (`,`) o el operador `+` para concatenar.

- **Coma (`,`):** Concatena elementos separándolos automáticamente por un espacio. Es la forma más segura.
- **Operador `+` :** Requiere que todos los elementos sean del tipo `str` . Si una variable es numérica, debe convertirse primero con `str()` .

Parámetros Adicionales Importantes (Examen)

Parámetro	Propósito	Ejemplo	Resultado
`sep`	Define el separador entre elementos pasados por coma.	`print(1, 2, sep='-')`	`1-2`
`end`	Define el carácter que se añade al final de la línea (por defecto es el salto de línea `\n`).	`print('Hola', end=' ')`	`Hola Mundo` (sin salto)

B. Función `input()` y Conversión

La función `input()` **SIEMPRE devuelve un valor de tipo** `str` **(cadena)**. Para usar ese valor en operaciones matemáticas, debes convertirlo (hacer *casting*).

```
edad2 = int(input("Introduce tu edad: ")) # Convierte la cadena a Entero
sueldo = float(input("Dime tu sueldo: ")) # Convierte la cadena a Flotante
```

4. Estructuras de Control (Condicionales y Bucles)

A. Condicionales

Se usa `if`, `elif` y `else` para ejecutar bloques de código basados en condiciones.

```
if int(edad2) < 18:
    print("Eres menor de edad")
else:
    print("Eres mayor de edad")
```

B. Bucles `for` y `range()`

El bucle `for` se usa para iterar sobre una secuencia de elementos. La función `range()` se usa para generar secuencias numéricas.

$$\text{range}(\text{inicio}, \text{fin}, \text{paso})$$

Sintaxis	Propósito	Ejemplo
`range(fin)`	Genera números desde 0 hasta `fin-1`.	`range(3)` → `0, 1, 2`
`range(ini, fin)`	Genera números desde `ini` hasta `fin-1`.	`range(2, 5)` → `2, 3, 4`
`range(ini, fin, paso)`	Permite saltar valores. El `paso` puede ser negativo para decrecer.	`range(0, 10, 2)` → `0, 2, 4, 6, 8`
`range(fin, ini, -paso)`	Decrecer.	`range(10, 0, -1)` → `10, 9, 8, ... 1`

5. Utilidades Básicas

- `max()` / `min()`: Devuelve el valor más grande/pequeño de una secuencia de valores.
- `isinstance(objeto, tipo)`: Función muy útil para verificar si un objeto pertenece a un tipo específico. Devuelve `True` o `False`.

```
if isinstance(5, int):
    print("Es numero entero")
```

6. Consejos para Mejorar la Programación con Variables

🎯 **Principio 1: Nombres de Variables Significativos (Legibilidad)**

- **Evita nombres cortos o vagos** como `x`, `y`, `temp`, o `v1`.
- **Usa nombres descriptivos** como `sueldo_mensual` o `nombre_cliente`. Un código legible es un código fácil de mantener y aprobar.

🎯 **Principio 2: Evitar Conversiones Innecesarias**

- Planifica la entrada de datos para evitar tener que convertir tipos de forma repetida dentro del código. Por ejemplo, convierte la edad a `int` una sola vez justo después de recibir el `input()`.

🎯 **Principio 3: Tipos Inmutables vs. Mutables**

- **Entender la diferencia es clave:**
  - **Inmutables (** `int` , `float` , `str` , `tuple` **):** No cambian una vez creados.
  - **Mutables (** `list` , `dict` , `set` **):** Pueden modificarse después de la creación (añadir, eliminar elementos).
  - El código que has visto se enfoca principalmente en tipos inmutables.