Python TOTAL

# redondeo

El redondeo facilita la interpretación de los valores calculados al limitar la cantidad de decimales que se muestran en pantalla. También, nos permite aproximar valores decimales al entero más próximo.

```
round(number,ndigits)
```

valor a redondear

cantidad de decimales

*(si se omite, el resultado es entero)*

## algunos ejemplos de uso

```
print(round(100/3))
>> 33

print(round(12/7,2))
>> 1.71
```

Python TOTAL

# operadores matemáticos

Veamos cuáles son los operadores matemáticos básicos de Python, que utilizaremos para realizar cálculos:

| | |
|---|---|
| Suma: | `+` |
| Resta: | `-` |
| Multiplicación: | `*` |
| División: | `/` |
| Cociente (división "al piso"): | `//` |
| Resto (módulo): | `%` |
| Potencia: | `**` |
| Raíz cuadrada: | `**0.5` |

útil para detectar valores pares ;)

¡es un caso especial de potencia!

Python TOTAL

# formatear cadenas

Para facilitar la concatenación de variables y texto en Python, contamos con dos herramientas que nos evitan manipular las variables, para incorporarlas directamente al texto:

- **Función format**: se encierra las posiciones de las variables entre corchetes { }, y a continuación del string llamamos a las variables con la función format

```
print("Mi auto es {} y de matrícula
{}".format(color_auto,matricula))
```

- **Cadenas literales (f-strings)**: a partir de Python 3.8, podemos anticipar la concatenación de variables anteponiendo f al string

```
print(f"Mi auto es {color_auto} y de
matrícula {matricula}")
```

Python TOTAL

# conversiones

Python realiza conversiones implícitas de tipos de datos automáticamente para operar con valores numéricos. En otros casos, necesitaremos generar una conversión de manera explícita.

```
int(var)
>> <class 'int'>
```

Convierte el dato en integer

```
float(var)
>> <class 'float'>
```

Convierte el dato en float

# integers & floats

Python TOTAL

Existen dos tipos de datos numéricos básicos en Python: int y float. Como toda variable en Python, su tipo queda definido al asignarle un valor a una variable. La función type() nos permite obtener el tipo de valor almacenado en una variable.

## int

Int, o integer, es un número entero, positivo o negativo, sin decimales, de un largo indeterminado.

```
num1 = 7
print(type(num1))
>> <class 'int'>
```

## float

Float, o "número de punto flotante" es un número que puede ser positivo o negativo, que a su vez contiene una o más posiciones decimales.

```
num2 = 7.525587
print(type(num2))
>> <class 'float'>
```

Python TOTAL

# nombres de variables

> Existen convenciones y buenas prácticas asociadas al nombre de las variables creadas en Python. Las mismas tienen la intención de facilitar la interpretabilidad y mantenimiento del código creado.

## reglas

1. **Legible**: nombre de la variable es relevante según su contenido
2. **Unidad**: no existen espacios (puedes incorporar guiones bajos)
3. **Hispanismos**: omitir signos específicos del idioma español, como tildes o la letra ñ
4. **Números**: los nombres de las variables no deben empezar por números (aunque pueden contenerlos al final)
5. **Signos/símbolos**: no se deben incluir : " ' , < > / ? | \ ( ) ! @ # $ % ^ & * ~ - +
6. **Palabras clave**: no utilizamos palabras reservadas por Python

Python TOTAL

# variables

Las variables son espacios de memoria que almacenan valores o datos de distintos tipos, y (como su nombre indica) pueden variar. Se crean en el momento que se les asigna un valor, por lo cual en Python no requerimos declararlas previamente.

## algunos ejemplos de uso

```
pais = "México"


nombre = input("Escribe tu nombre: ")
print("Tu nombre es " + nombre)


num1 = 55
num2 = 45
print(num1 + num2)
>> 100
```

# tipos de datos

Python TOTAL

En Python tenemos varios tipos o estructuras de datos, que son fundamentales en programación ya que almacenan información, y nos permiten manipularla.

## texto (str)

```
"Python"
"750"
```

## números

```
int 250
float 12.50
```

## booleanos

```
True
False
```

| estructuras | mutable | ordenado | duplicados |
|---|---|---|---|
| listas [ ] | ✓ | ✓ | ✓ |
| tuplas ( ) | ✗ | ✓ | ✓ |
| sets { } | ✓ | ✗ | ✗ |
| diccionarios { } | ✓ | ✗* | ✗:✓** |

*: En Python 3.7+, existen consideraciones

**: key es única; value puede repetirse