# 1. Datos Simples: Numéricos:

Como literales numéricos están:

## 1.1. Enteros: <class 'int'>.

¿*Limite rango enteros*?

En Python 3, no hay límite para un número entero. Por supuesto, está restringido por la cantidad de memoria que tiene su sistema, al igual que todas las cosas, pero más allá de eso, un número entero puede ser tan largo como lo necesite:

Los números **Enteros** (`int`) entero: **<class 'int'>**, con varios posibles formatos, según base (decimal, octal, hexadecimal, binario).

¿*Creación*?

- Utilizando el **formato literal**:
- Utilizando **funciones** (int(), oct(), hex(), bin().

  Nota: Las funciones también se utilizan para "reconvertir datos"

**`int(x)`** ----------La función constructora int(). Devuelve un número **entero**, y dado el caso en una base concreta.

+

```
[variable =] int(valor[base])
```

+ **`valor`** : Un número o una cadena que se puede convertir en un número entero. un literal flotante (redondeando hacia abajo al número entero anterior), o un literal de cadena (siempre que la cadena represente un número entero)

+ `base` : Un número que representa el formato del número. Valor por defecto: 10. Las bases válidas son 0 y 2-36.

**`oct(x)`** ----------La función oct(). Convierte un número entero en formato decimal a formato octal, con el prefijo 0o.

```
[variable =] oct(valor-entero)
```

**`hex(x)`** ----------La función hex(). Convierte un número entero en formato decimal a formato hexadecimal, con el prefijo 0x.

+

```
[variable =] hex(valor-entero)
```

**`bin(x)`** ----------La función bin(). Returns the binary version of a number-

P2● ~~Entero largo (**`long`**) *entero long* / :~~*~~¿?~~ entero_largo= 23~~*~~L~~

- Algunas funciones asociadas: round(),abs() . .

**`round(x)`** ----------La función round(). Devuelve un número **entero**. Redondea.

+

```
[variable =] round(valor[, dígitos])
```

+ **valor** El número a redondear. Un número o una cadena que se puede convertir en un número entero o en un literal flotante (redondeando hacia abajo al número entero anterior), o un literal de cadena (siempre que la cadena represente un número entero)
+ dígitos opcional. El número de decimales a utilizar al redondear el número. El valor predeterminado es 0.

**abs (*x*)** ----------La función abs(), devuelve el valor absoluto del número especificado.

```
[variable =] abs (valor)
```

+ **valor** El número para el valor absoluto.

## 1.2. Flotantes: <class 'float'>.

Casi todas las plataformas representan los valores `float` Python como valores de "doble precisión" de 64 bits, según el estándar IEEE 754 . En ese caso, el valor máximo que puede tener un número de punto flotante es aproximadamente $1.8 \times 10^{308}$ .

Los números **Flotantes**. Reales, con parte decimal (**float**) : **<class 'float'>** .

¿*Creación*?

- Utilizando el **formato literal**.
  + Un numero con el punto decimal.
  + Números en formato exponencial, utilizando el carácter e o E.
- Función constructora: float() .

**float (*x*)** ----------La función constructora float(). Devuelve un número en coma flotante.

+ construye un número flotante a partir de un literal entero, un literal flotante o un literal de cadena (siempre que la cadena represente un flotante o un entero)

```
[variable =] float (xxx)
```

- Otras: la división normal **/** siempre retorna un numero en coma flotante.

## 1.3. Complejos: <class 'complex'>.

Los números **Complejos**. Imaginarios (**complex**) *complejo* / : **<class 'complex'>**, los complejos se especifican como *<real part>+<imaginary part>j* .

¿*Creación*?

- Utilizando el **formato literal**.
  + Se usa el sufijo **j** o J para indicar la parte imaginaria.
- Función constructora: complex().

**complex (*x*)** ----------La función constructora complex().Devuelve un número en formato complejo.

```
[variable =] complex (xxxx)
```

## 2. Datos Simples: Booleanos (verdadero/falso) <class 'bool'>

¿*Creación*?

- Utilizando el formato “literal”, utilizando constantes: True /False.

En la mayoría de los lenguajes: `True` es igual a 1 y `False` es igual a 0.
En Python, además se pueden usar los literales booleanos en las expresiones matemáticas.

- Función constructora: bool() .

**`bool(x)`** ---------- La función constructora bool().Devuelve el valor booleano del objeto especificado.
El objeto siempre devolverá True, a menos que:
El objeto está vacío, como [], (), {}
El objeto es falso
El objeto es 0
El objeto es ninguno

```
[variable =] bool(objeto)
```

+ **`objeto`**: Cualquier objeto, como cadena, lista, número, etc.