

Curso Python 2025 Día 2: Variables numéricas



Hoy:

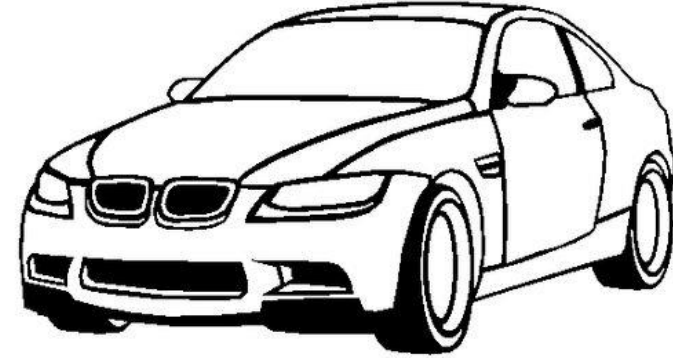
- ✓ Repaso: lenguaje y entorno
- ✓ Tipos numéricos: int, float
- ✓ Variables
- ✓ Operadores aritméticos



Repaso: Python

Python:

- Lenguaje de programación multipropósito
- Es interpretado (se ejecuta según se lee)
- Todo son objetos con atributos y funcionalidades



Atributos

- Color
- Marca
- Posición
- Velocidad

Funcionalidades

- Arrancar
- Encender luces
- Acelerar
- Frenar

Repaso: Lenguaje

PEP 8:

- Guía de estilo para la implementación de Python en C (C-Python)
- snake_case
- Uso de “_” para variables internas/reservadas/...

snake_case	PascalCase camelCase
bank_account	BankAccount bankAccount
__add__ __lt__ _internal_var

Repaso: Entorno

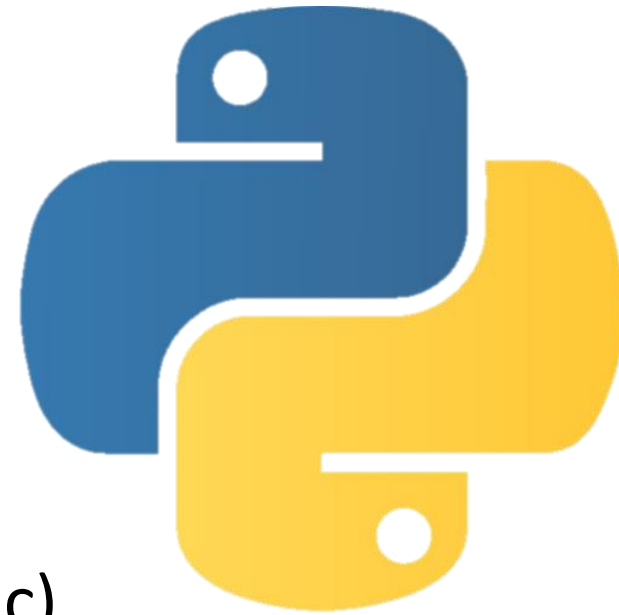
Herramientas:



Repaso: Entorno

Creación y activación entorno virtual:

- `python -m venv .venv`
- `.venv\Scripts\activate` (Windows)
- `source .venv/bin/activate` (Linux y Mac)



Entorno: Librerías

Una vez tengamos activo nuestro venv:

- `pip install <module>`
- `pip install -r requirements.txt`



Tipos

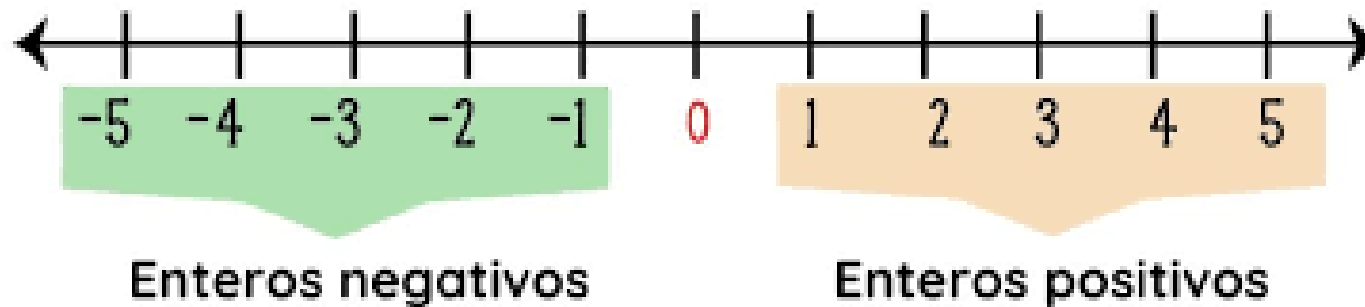
En un programa los elementos con los que trabajamos tienen un tipo asociado

- Javier es una persona
- El número de páginas de un libro es un número
- Una afirmación puede ser True o False
- ...

Tipos numéricos: integers

Valores numéricos enteros, en Python decimos que son de tipo **int**

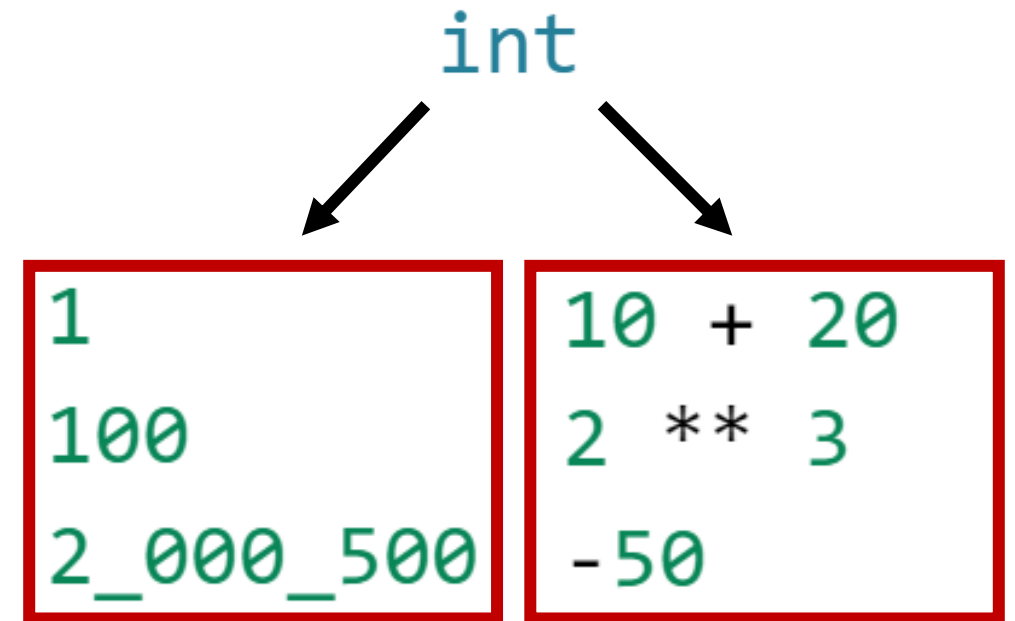
- 0, 2, 100, -100, etc
- Tienen representación exacta



Tipos numéricos: integers

Los podemos obtener de dos formas:

1. A partir de un literal
2. Como resultado de un cálculo

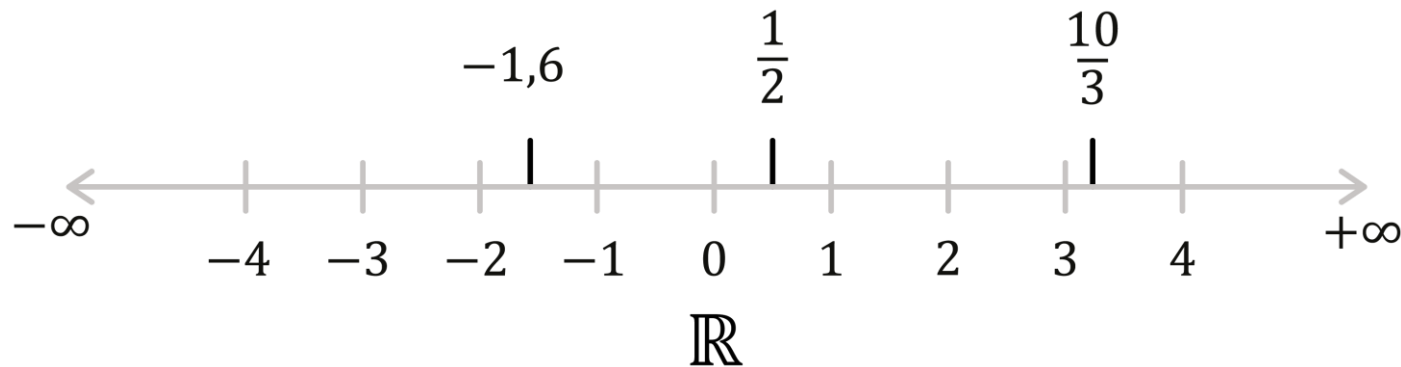


Nota: Un literal es la representación en código fuente

Tipos numéricos: floats

Valores numéricos en coma flotante (floating point),
en Python decimos que son de tipo **float**

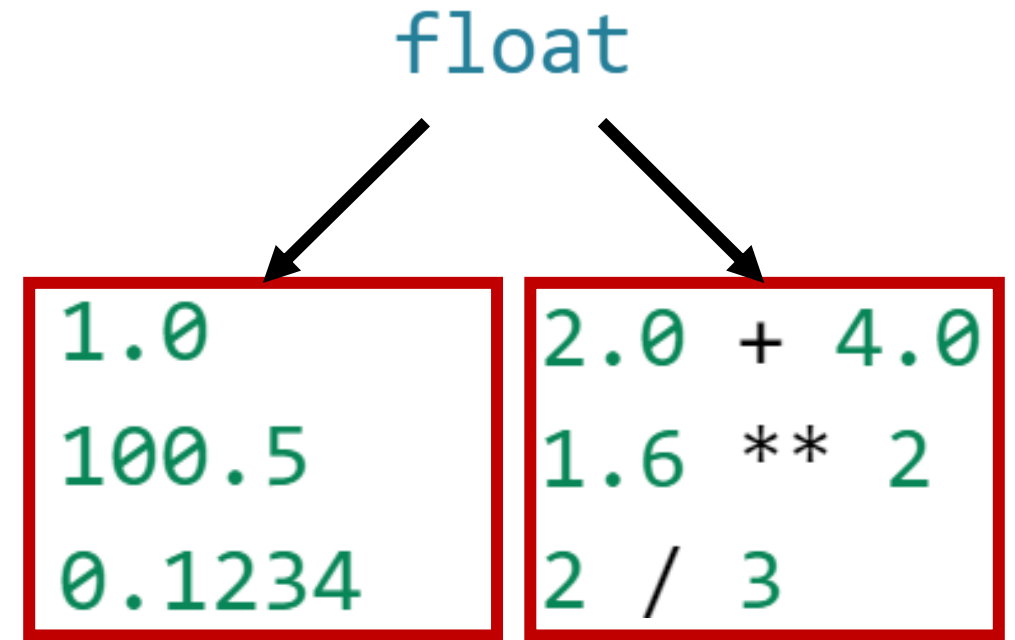
- Pueden no tener representación exacta!



Tipos numéricos: floats

Los podemos obtener de dos formas:

1. A partir de un literal
2. Como resultado de un cálculo



Nota: El punto decimal diferencia **int** de **float**

Representar floats

Para representar valores numéricos habitualmente utilizamos el sistema decimal (base 10)

$$283 \rightarrow \boxed{2} \cdot 10^2 + \boxed{8} \cdot 10^1 + \boxed{3} \cdot 10^0$$

$$2.357 \rightarrow 2 \cdot 10^0 + 3 \cdot 10^{-1} + 5 \cdot 10^{-2} + 7 \cdot 10^{-3}$$

$$\frac{1}{3} = 0.333 \dots = 0.\bar{3} \longrightarrow \text{No lo podemos representar!!!}$$

Representar floats

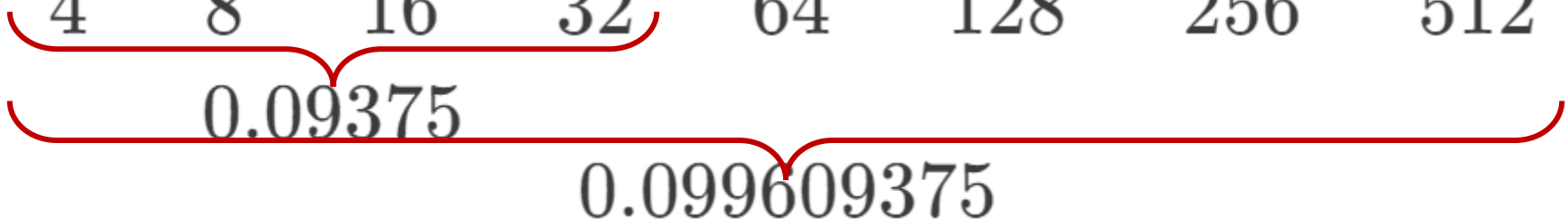
Las máquinas utilizan el sistema binario

$$0.101_2 = \boxed{1} \cdot 2^{-1} + \boxed{0} \cdot 2^{-2} + \boxed{1} \cdot 2^{-3} = \frac{1}{2} + \frac{0}{4} + \frac{1}{8} = 0.625_{10}$$

Algunos reales como 0.625 tienen representación exacta...

Pero otros no (igual que en base 10):

$$0.1 = \frac{0}{2} + \frac{0}{4} + \frac{0}{8} + \frac{1}{16} + \frac{1}{32} + \frac{0}{64} + \frac{0}{128} + \frac{1}{256} + \frac{1}{512} + \dots$$



Floats: Cuidado

No todos los números tienen representación exacta

- Podemos no estar trabajando con valores exactos! (usar tolerancias)
- Existen tipos para manejar de manera exacta pero no son eficientes:
 - Alto consumo en memoria
 - Baja velocidad de cálculo

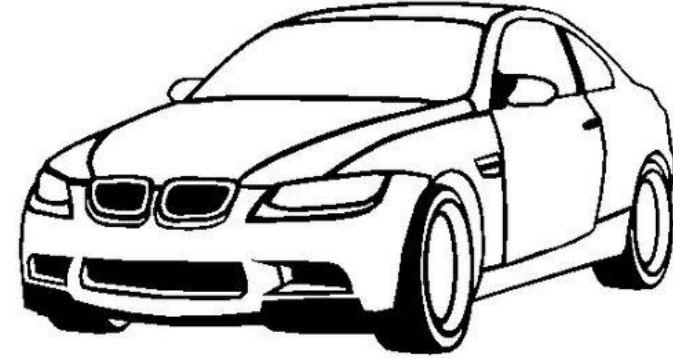


Objetos

En Python todo son objetos

- Tienen estado (atributos)
- Tienen Funcionalidad (métodos)

Nota: Las funcionalidades suelen requerir de parámetros adicionales



Atributos

- Color
- Marca
- Posición
- Velocidad

Funcionalidades

- Arrancar
- Encender luces
- Acelerar
- Frenar

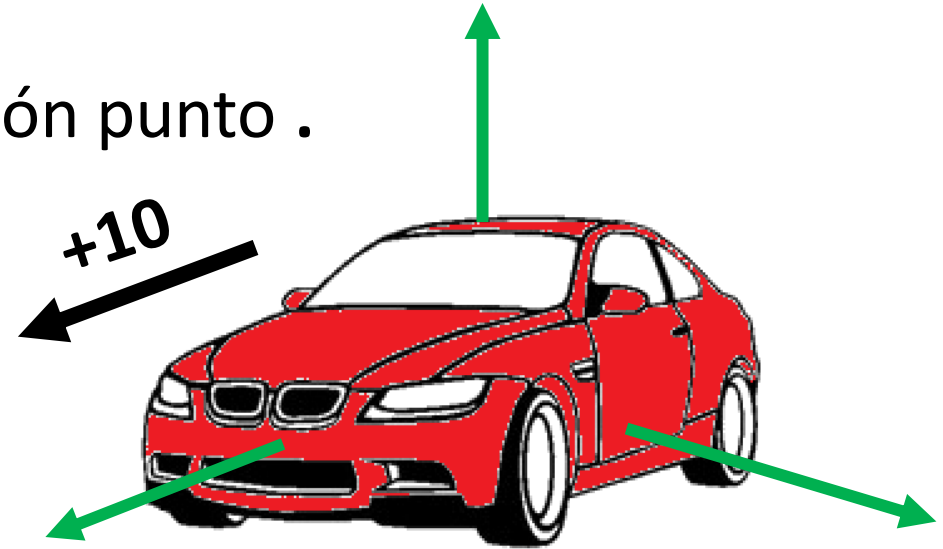
Objetos

Para acceder a ellos utilizamos la notación punto .



`car_1.color` → ●

`car_1.start()`



`car_2.color` → ●

`car_2.accelerate(10, "km")`

Objetos: int y float

Como todos, los tipos numéricos vistos también son objetos

- Tienen un estado interno (su valor)

10

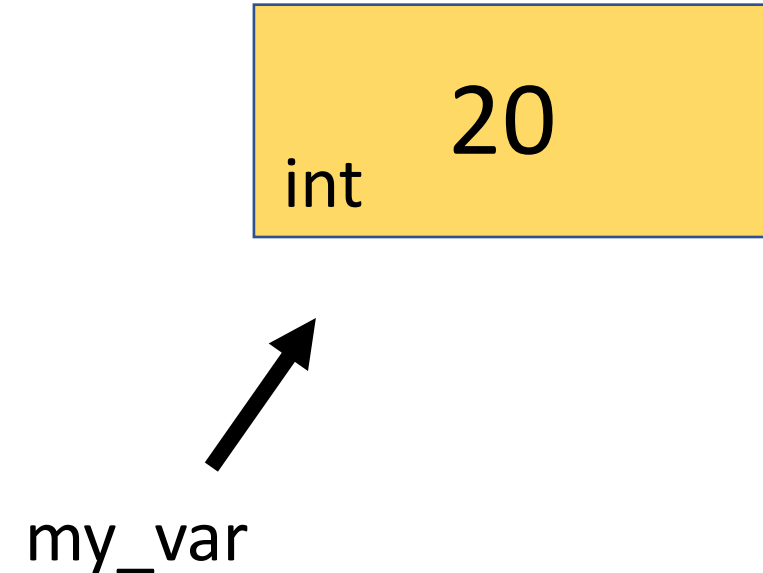
- Saben sumarse con otros objetos (entre otras funcionalidades)

(10).__add__(5) → 15

Variables

Frecuentemente necesitamos etiquetar los objetos:

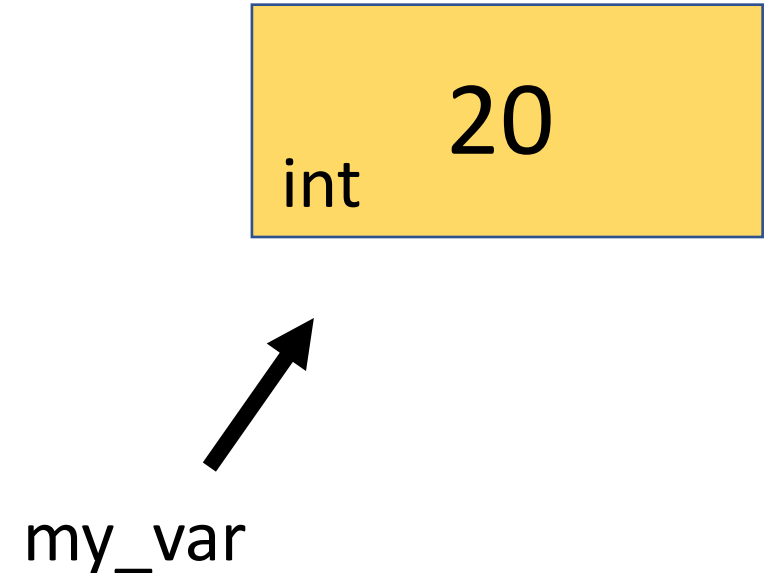
- Nos ayuda a acordarnos de cual era su utilidad
- Nos permite reutilizar un objeto en distintas partes de nuestro código



Variables: asignación

- Para definir una variable utilizamos el operador de asignación “=”
- Decimos que nuestra etiqueta referencia a un objeto

Ejemplo: `my_var = 20`



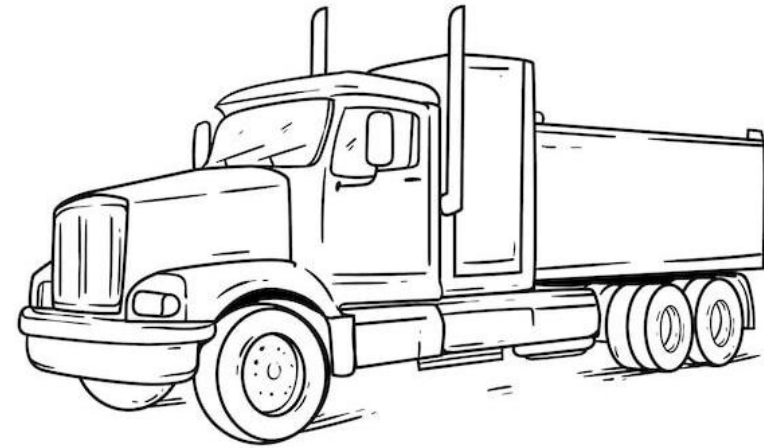
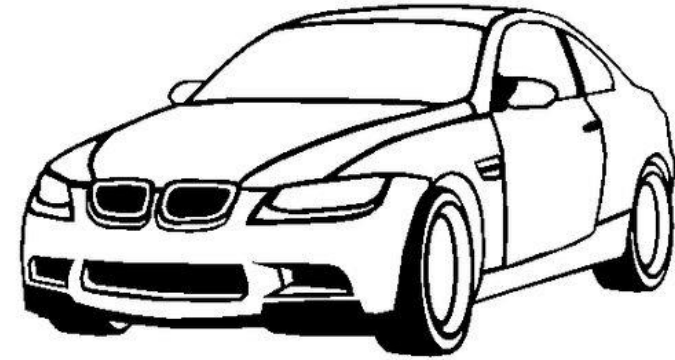
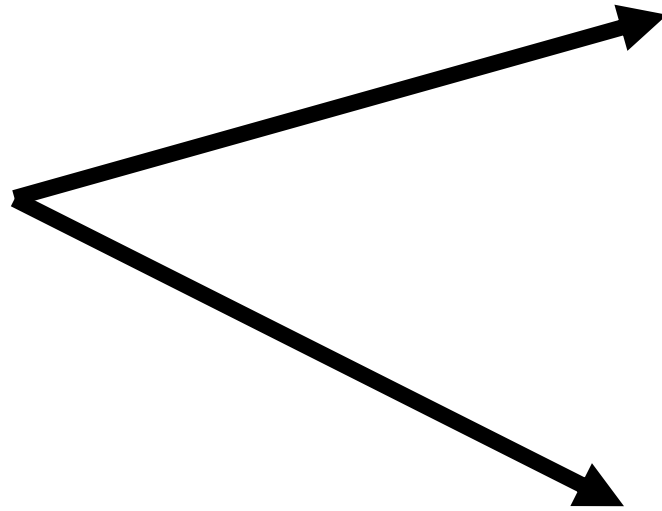
Variables: variabilidad

my_car

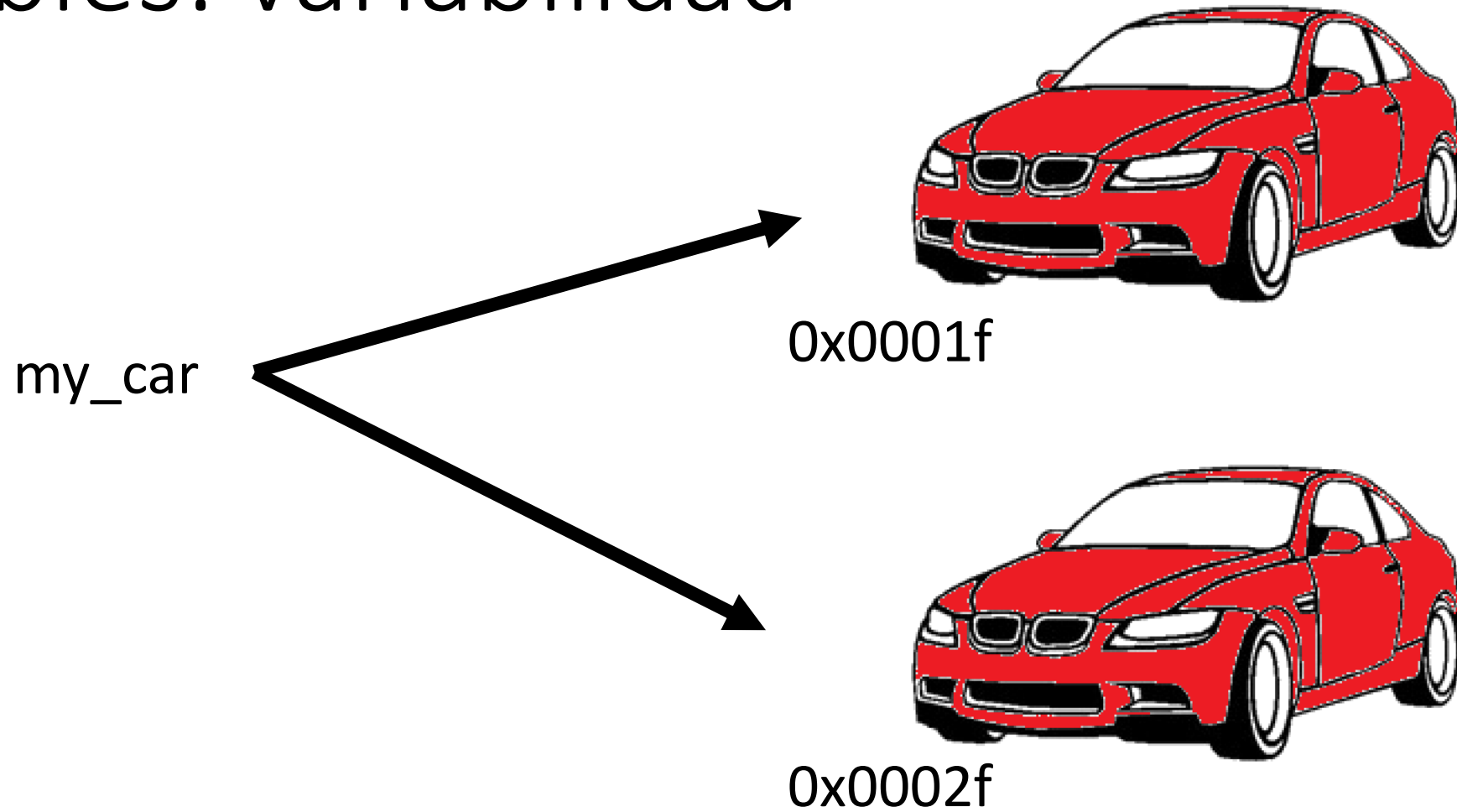


Variables: variabilidad

my_car



Variables: variabilidad



Variables: asignación

¿Cómo ocurre la asignación?

1. Primero se evalúa completamente RHS
2. Segundo se evalúa completamente LHS



Sencillo... pero
importante en
datos
desempaquetables

$\underbrace{\text{my_var}}_{\text{LHS}} = \underbrace{20}_{\text{RHS}}$

Operadores

Un operador es un símbolo que realiza una operación entre uno o más valores

- Aritméticos
- Relacionales (comparaciones)
- Lógicos (and, or, not)



Operadores aritméticos

Dependiendo de sobre cuantos valores

- Unarios
 - 10
 - +0.2
- Binarios
 - 10 + 20
 - 2 ** 3
 - ...



Operadores aritméticos

$a + b$	# Addition
$a - b$	# Subtraction
$a * b$	# Multiplication
a / b	# Division
$a ** b$	# Exponentiation
$a // b$	# Floor division
$a \% b$	# Modulo



Operadores aritméticos

Cuidado con la jerarquía de operadores, usar parentésis!

$$10 + 4 / 2 \quad \# \quad 12$$

$$(10 + 4) / 2 \quad \# \quad 7$$

Operadores aritméticos: Precedencia

Operators	Meaning
<code>()</code>	Parentheses
<code>**</code>	Exponent
<code>+x</code> , <code>-x</code> , <code>~x</code>	Unary plus, Unary minus, Bitwise NOT
<code>*</code> , <code>/</code> , <code>//</code> , <code>%</code>	Multiplication, Division, Floor division, Modulus
<code>+</code> , <code>-</code>	Addition, Subtraction



Operadores aritméticos

En Python lo que hace el operador lo determina el objeto que se está operando

`a * b` `# Multiplication`

Lo que realmente ocurre: `a.__mul__(b)`

Módulos math y cmath

Más tipos numéricos y operadores como módulos de la librería estándar

```
import math
```

```
math.sqrt(9)
```

Raíz cuadrada

```
math.floor(3.14)
```

Entero más grande menor igual que x

```
math.log1p(x)
```

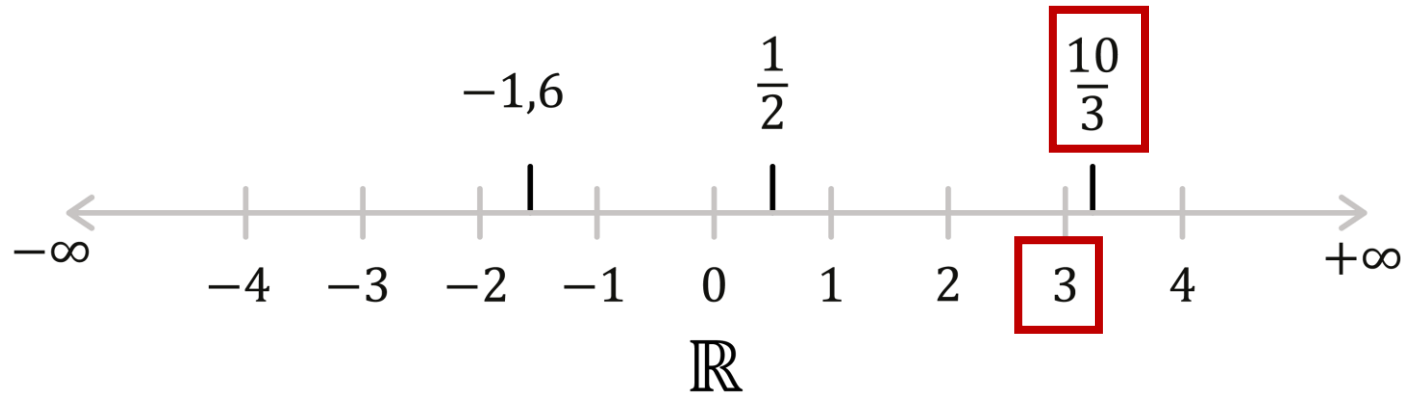
Logaritmo natural (base e) de x + 1



`math.floor(x)`

Devuelve el entero más grande menor igual que x

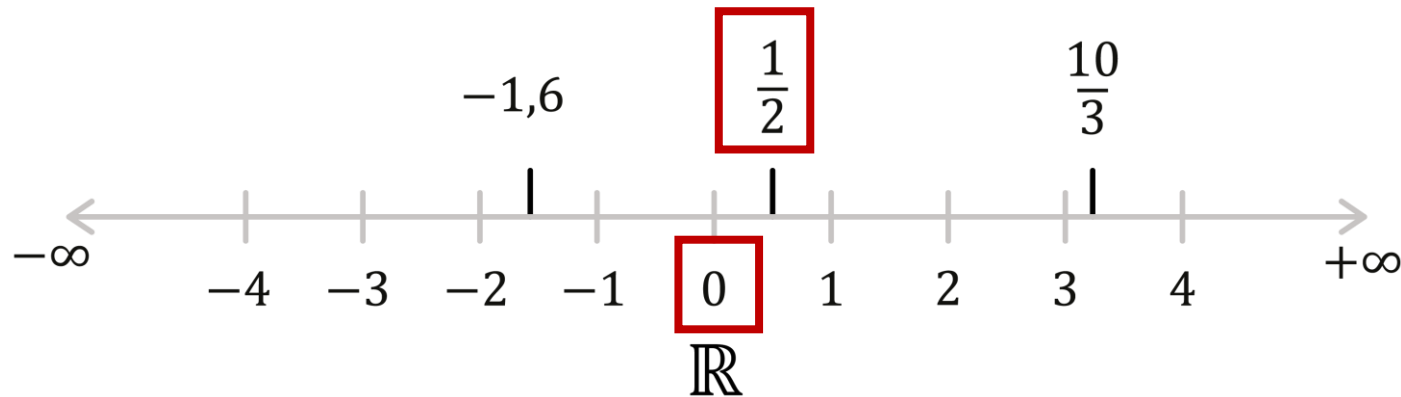
`math.floor(3.3)`



`math.floor(x)`

Devuelve el entero más grande menor igual que x

`math.floor(0.5)`

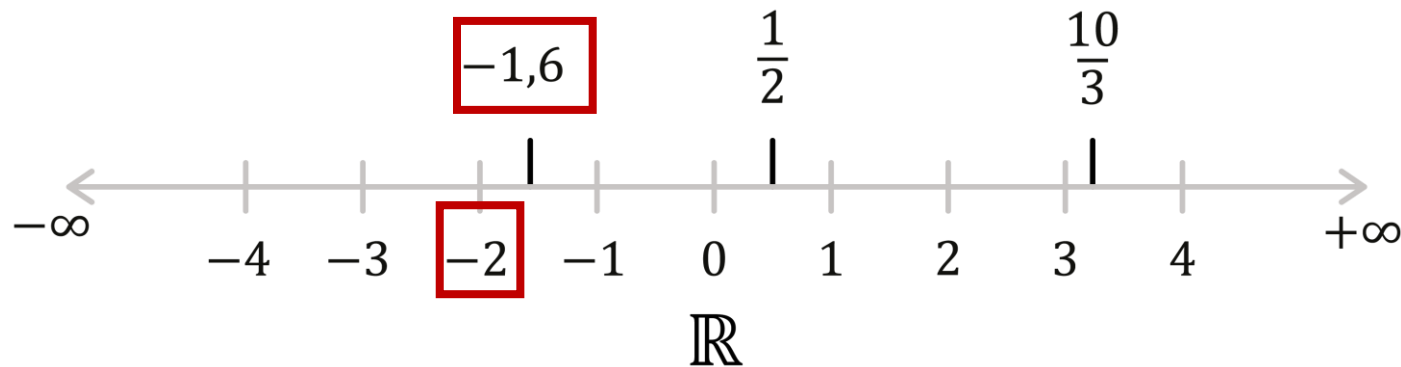


`math.floor(x)`

Devuelve el entero más grande menor igual que x

`math.floor(-1.6)`

NO TRUNCA X!!!



Operadores aritméticos

`a // b = math.floor(a / b) # Floor division`

`a % b = a - b (a // b) # Modulo`

`a / b = a // b + (a % b) / b`

