Números en `Python`

Tipos de números

- `int`: número entero
- `float`: número en coma flotante

Para saber el tipo de dato de un número podemos utilizar la función `type()`

```
type(5)
```

```
int
```

```
type(5.0)
```

```
float
```

Observación. En `Python`, para referirnos a números de tipo `float` con todo 0's en la parte decimal como `3.0`, basta que indiquemos `3.`. Es decir, `Python` entiende que los números `3.0` y `3.` son el mismo, incluyendo que son del mismo tipo: `float`.

```
type(3.0)
```

```
float
```

```
type(3.)
```

```
float
```

Podemos indicar el tipo de número que deseamos utilizar con las funciones `int()` y `float()`.

```
type(int(7.0))
```

```
int
```

```
type(int(9.))
```

```
int
```

```
type(float(3))
```

```
float
```

```
float
```

¡Cuidado! Es sencillo pasar de enteros a números en coma flotante, ya que siempre es posible, pero no siempre podemos pasar de números en coma flotante a números enteros, pues se pierde la parte decimal.

```
int(3.5)
```

```
3
```

Operaciones aritméticas

Suma

Para sumar dos números, utilizamos la función `+`

```
2 + 1
```

```
3
```

```
2.0 + 1.
```

```
3.0
```

```
2 + 1.0
```

```
3.0
```

Observación. Fijémonos que al combinar un número entero (int) y un número en coma flotante (float), el resutado es un número float. Esto ocurre para todas las operaciones aritméticas en `Python`.

Resta

Para restar dos números, utilizamos la función `-`

```
7 - 3
```

```
4
```

```
7.0 - 3.
```

```
4.0
```

```
7 - 3.0
```

```
4.0
```

Producto

Para multiplicar dos números, utilizamos la función `*`

```
8 * 6
```

```
48
```

```
8. * 6.
```

```
48.0
```

```
8.0 * 6
```

```
48.0
```

División

Para dividir dos números, utilizamos la función `/`

```
6 / 5
```

```
1.2
```

```
6. / 5.0
```

```
1.2
```

```
6 / 5.0
```

```
1.2
```

¡Cuidado! Hay que tener en cuenta el tipo de número (int o float) cuando vayamos a dividir en `Python`, porque en algunas versiones, si dividimos dos números enteros, se lleva a cabo la división entera automáticamente.

División entera o Euclídea

Dados dos números naturales a y b, con $b \neq 0$, la división Euclídea de a entre b asocia un cociente q y un resto r, ambos números naturales, que satisfacen

- $a = b \cdot q + r$
- $r < b$

Ejemplo 1

Si queremos la división entera de $a = 7$ entre $b = 5$, tendremos que el cociente es $q = 1$ y el resto es $r = 2$, ya que

$$7 = 5 \cdot 1 + 2$$

y el resto r es menor al divisor b. Es decir, $2 < 5$.

Para obtener el cociente de la división entera, utilizamos la función `//`

```
10 // 3
```

```
3
```

Para obtener el resto de la división entera, utilizamos la función `%`

```
10 % 3
```

```
1
```

Potencia

Para calcular la potencia n-ésima de un número, usamos la función `**`

```
5 ** 3
```

```
125
```

```
5.0 ** 3.0
```

```
125.0
```

```
5.0 ** 3
```

```
125.0
```

Para calcular la potencia n-ésima de un número, también podemos usar la función `pow()`

```
pow(5, 3)
```

```
125
```

```
pow(5., 3.0)
```

```
125.0
```

```
pow(5, 3.)
```

```
125.0
```

Orden de las operaciones aritméticas

El orden en que se llevan a cabo las operaciones aritméticas en `Python` es el siguiente:

- Primero se calcula lo que se halla entre paréntesis.
- A continuación, las potencias.
- Después, productos y divisiones. En caso de haber varias, el orden que se sigue es de izquierda a derecha.
- Finalmente, sumas y restas. En caso de haber varias, el orden que se sigue es de izquierda a derecha.

```
6 + 2 * 8 / 4 - 2 ** 3
```

```
2.0
```

```
(6 + 2) * (8 / (4 - 2)) ** 3
```

```
512.0
```

```
(6 + 2) * 8 / (4 - 2) ** 3
```

```
8.0
```

Observación. El uso de los paréntesis puede cambiar completamente el resultado. No conviene abusar de ellos, aunque es mejor que sobren, ya que ayudan a entender el orden en que se van a llevar a cabo las operaciones.

Números complejos

Definiciones

- **Número complejo.** Es un par ordenado de números reales $z = (a, b)$, con $a, b \in \mathbb{R}$.

- **Parte real.** Es el primer elemento del par ordenado, $\text{Re}(z) = a$.
- **Parte imaginaria.** Es el segundo elemento del par ordenado, $\text{Im}(z) = b$.
- **Complejo real.** $z = (a, 0)$.
- **Imaginario puro.** $z = (0, b)$.
- **Unidad imaginaria.** $i = (0, 1)$.
- **Conjunto de números complejos.** $\mathbb{C} = \{z = (a, b) \ : \ a, b \in \mathbb{R}\}$.

Operaciones

- Suma: $(a, b) + (c, d) = (a + c, b + d)$
- Resta: $(a, b) - (c, d) = (a - c, b - d)$
- Producto: $(a, b) \cdot (c, d) = (a \cdot c - b \cdot d, a \cdot d + b \cdot c)$
- División: $(a, b) \div (c, d) = \frac{(a \cdot c + b \cdot d, b \cdot c - a \cdot d)}{c^2 + d^2} = \left(\frac{a \cdot c + b \cdot d}{c^2 + d^2}, \frac{b \cdot c - a \cdot d}{c^2 + d^2}\right)$

Conjugado, Módulo y Argumento

Dado un complejo $z = (a, b)$,

- **Conjugado.** $\bar{z} = (a, -b)$.
- **Módulo.** $\text{Mod}(z) = |z| = \sqrt{\text{Re}(z)^2 + \text{Im}(z)^2} = \sqrt{a^2 + b^2}$.
- **Argumento.** $\text{Arg}(z) = \arctan\left(\frac{\text{Im}(z)}{\text{Re}(z)}\right) = \arctan\left(\frac{b}{a}\right)$

Unidad imaginaria

$i = (0, 1)$ satisface

$$i^2 = (0, 1)^2 = (0, 1)\cdot (0, 1) = (-1, 0)$$

De aquí obtenemos la igualdad $i = \sqrt{-1}$, que es otra de las definiciones que se le da a la unidad imaginaria.

Otras representaciones

Representación binómica: $z = a + bi$$z = a + bi$

- $a = \text{Re}(z)$$a = \text{Re}(z)$
- $b = \text{Im}(z)$$b = \text{Im}(z)$

Representación polar: $z = re^{i\phi}$$z = re^{i\phi}$

- $r = \text{Mod}(z)$$r = \text{Mod}(z)$
- $\phi = \text{Arg}(z)$$\phi = \text{Arg}(z)$

Números complejos en `Python`

Observación. En `Python`, los números complejos se definen en forma binómica y en vez de utilizar una `i`, se utiliza la letra `j` para representar la unidad imaginaria.

```
z = 2 + 5j
z
```

```
(2+5j)
```

```
type(z)
```

```
complex
```

También podemos definir números complejos en `Python` con la función `complex()`

```
z = complex(1, -7)
z
```

```
(1-7j)
```

```
type(z)
```

```
complex
```

Para obtener la parte real, utilizamos el método `.real`

```
z.real
```

```
1.0
```

Para obtener la parte imaginaria, utilizamos el método `.imag`

```
z.imag
```

```
-7.0
```

Para sumar números complejos, utilizamos la función `+`

```
z1 = 2-6j
z2 = 5+4j

z1 + z2
```

```
(7-2j)
```

Para restar números complejos, utilizamos la función `-`

```
z1 - z2
```

```
(-3-10j)
```

Para multiplicar una constante por un número complejo, o bien multiplicar dos números complejos, utilizamos la función `*`

```
-1 * z1
```

```
(-2+6j)
```

```
z1 * z2
```

```
(34-22j)
```

Para dividir números complejos, utilizamos la función `/`

```
z1 = -1 - 1j
z2 = 1 - 1j

z1 / z2
```

```
-1j
```

Observación. Si queremos indicar que la parte imaginaria es 1 o -1, no basta con poner `j` o `-j`, sino que hay que escribir `1j` o `-1j`, siempre que definamos el número complejo en su forma binómica.

Para calcular el conjugado de un número complejo, utilizamos el método `.conjugate()`

```
z = -2 + 1j
z.conjugate()
```

```
(-2-1j)
```

Para calcular el módulo de un número complejo, utilizamos la función `abs()`

```
z = -2j
abs(z)
```

```
2.0
```

Para calcular el argumento de un número complejo, utilizamos la función `phase()` del paquete `cmath`.

```
import cmath
cmath.phase(z)
```

```
-1.5707963267948966
```

Para pasar de forma binómica a forma polar, usamos la función `polar()` del paquete `cmath`.

```
z
```

```
(-0-2j)
```

```
cmath.polar(z)
```

```
(2.0, -1.5707963267948966)
```

Para pasar de forma polar a forma binómica, usamos la función `rect()` del paquete `cmath`.

```
cmath.rect(abs(z), cmath.phase(z))
```

```
(1.2246467991473532e-16-2j)
```