Python para Análisis de Datos

Módulo 01

Operaciones

Operadores aritméticos

Existe una variedad de operadores para trabajar con distintos tipos de datos y colecciones. Los operadores aritméticos (**+**, **-**, *****, **/**, **//**, ******, **%**) son operadores binarios que permiten realizar operaciones entre números, aunque algunos también funcionan para colecciones, como **+** y *****. También existe el operador **@** para multiplicación matricial (funciona con *arrays* de Numpy).

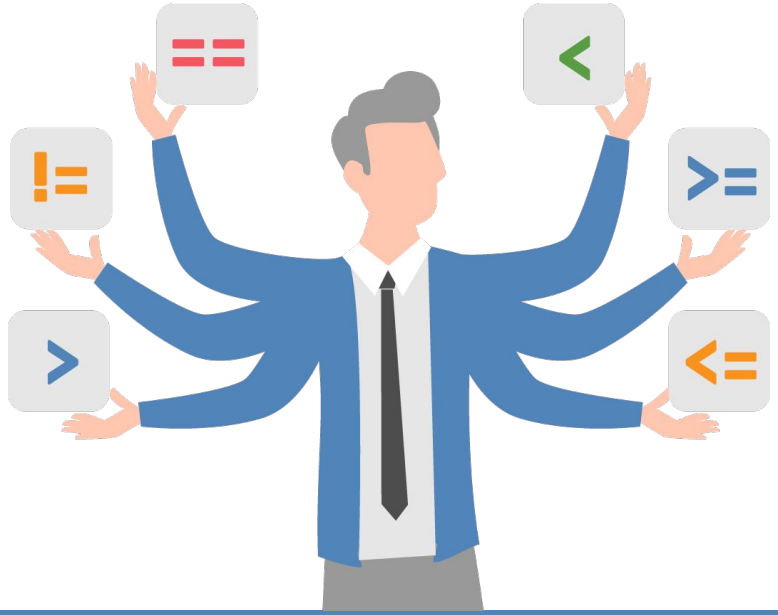
Estos operadores tienen un orden de precedencia: ****** se calcula antes que *****, **/**, **//**, **%** y estos antes que **+** y **-**. Para alterar el orden de estas operaciones en una expresión se utilizan paréntesis.

Operadores de comparación

Hay tres categorías de comparación: por valor, pertenencia e identidad y todos devuelven siempre booleanos. Los **operadores de comparación por valor** son **==**, **!=**, **>**, **>=**, **<**, **<=** que comparan los valores respectivos de los operandos, que no necesitan ser del mismo tipo de dato.

En general, los operadores de igualdad == y != están implementados para todos los tipos de datos mientras que los operadores de orden sólo están implementados para algunos tipos, como los numéricos y los strings.

Los operadores aritméticos tienen precedencia por sobre los de comparación.



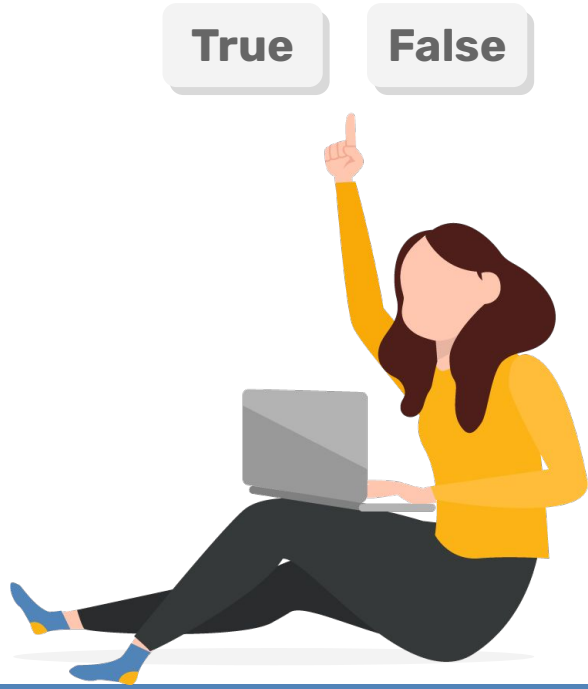
Operadores de pertenencia

Para chequear pertenencia tenemos los operadores **`in`** y **`not in`**.

Para contenedores como **listas, tuplas, sets** y **frozensets** el operador **in** da `True` si el objeto es un elemento del contenedor.

Para **strings** el operador **in** da `True` si el primer string es un substring del segundo, por lo que es más amplio que sólo chequear pertenencia de caracteres.

Para **diccionarios** se chequea si el objeto es una clave del diccionario, sin importar el valor correspondiente.



Identidad

Los operadores de identidad **`is`** e **`is not`** permiten evaluar si los operandos son el mismo objeto en la memoria, lo que internamente se determina usando la función **`id()`.**

Esto es útil para saber si dos variables apuntan al mismo objeto en la memoria o no, ya que el operador **`==`** sólo compara si los operandos, que pueden ser objetos distintos, tienen el mismo valor o no.

```
1 lista1 = [1,2,3]
2 lista2 = lista1
3 print(lista1 == lista2, lista1 is lista2)
```

```
True True
```

```
1 lista1 = [1,2,3]
2 lista2 = lista1.copy()
3 print(lista1 == lista2, lista1 is lista2)
```

```
True False
```

Operadores lógicos

Por último, tenemos los operadores lógicos, aquellos que **operan sobre booleanos y devuelven un booleano.** Se trata de los operadores de conjunción, disyunción y negación, cuyos operadores son **`and`, `or`** y **`not`.**

Estos operadores tienen el menor nivel de precedencia, por lo que se evalúan últimos en una expresión.

Los resultados de estos operadores están dados por las tablas de verdad.

Funciones incorporadas

Python tiene una variedad de funciones incorporadas para realizar distintos tipos de tareas sin la necesidad de importarlas.

Algunas son para construir y convertir entre distintos tipos de datos (como **`int()`, `float()`, `complex()`, `bool()`, `str()`, `list()`, `tuple()`, `dict()`, `set()`, `frozenset()`**), para trabajar con números (**`round()`, `abs()`, `divmod()`**), colecciones (**`len()`, `min()`, `max()`, `sorted()`, `reversed()`, `map()`, `filter()`**) y con el bucle *for* (**`range()`, `enumerate()`, `zip()`**). Esto le da gran flexibilidad al lenguaje.

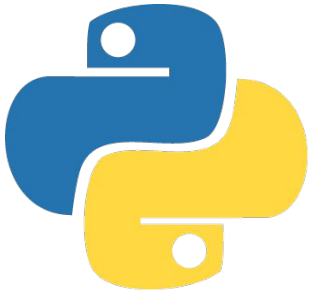
Para **consultar la documentación de una función** cualquiera desde **Jupyter Notebook** sólo hay que colocar el cursor al lado del nombre de la función y apretar Shift + Tab.

Métodos

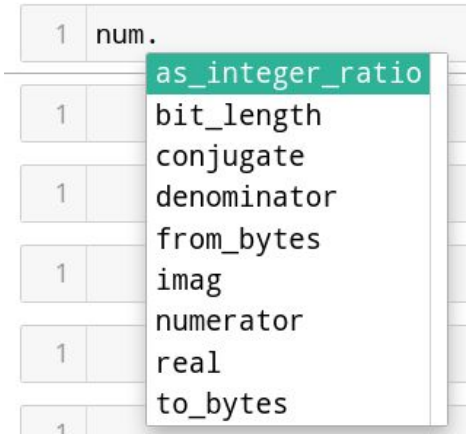
También se dispone de métodos para realizar acciones sobre tipos de dato en particular. Por ejemplo, para listas tenemos los métodos `append`, `insert`, `copy`, `pop`, `extend`, etc. Distintos tipos de datos implementan distintos métodos, extendiendo las funcionalidades del lenguaje.

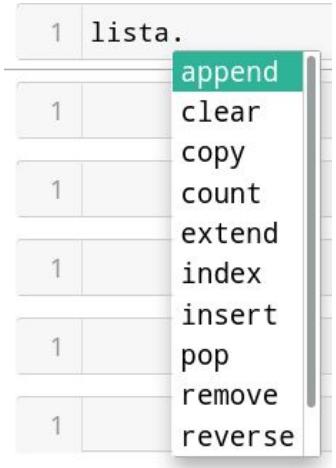
Para ver los métodos disponibles para un objeto en Jupyter Notebook sólo hay que tipear el nombre de la variable que referencia al objeto seguido de un punto y luego apretar Tab. Se muestra una lista con los métodos disponibles.

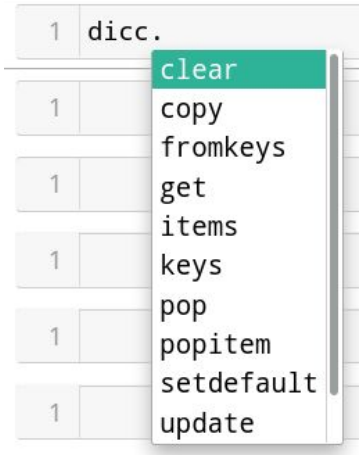
Acceder a la documentación de los métodos es igual que con las funciones.

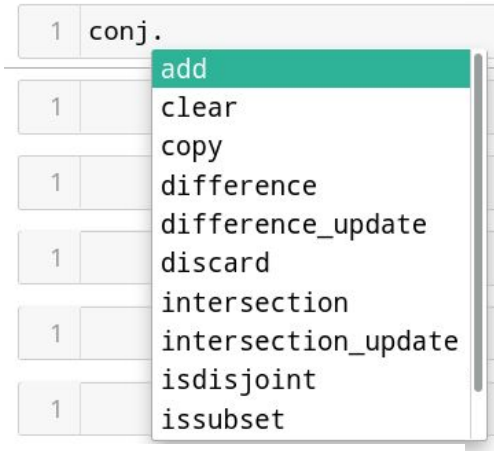


```
1 num = 1
2 lista = [1,2,3]
3 dicc = {1:1, 2:2}
4 conj = {1,2,3}
```









¡Muchas gracias!

¡Sigamos trabajando!