Contenido

[1. Numpy. 1](#_Toc143279983)

[1.1 Arreglos 1](#_Toc143279984)

[1.1.1 Accesos en arreglos bidimensionales. 1](#_Toc143279985)

[1.1.2 Operaciones con arreglos 1](#_Toc143279986)

[1.2 Forma o shape de un arreglo. 1](#_Toc143279987)

[2. Información de interes. 2](#_Toc143279988)

# Python.

## Caracteristicas generales.

* En Python todo es considerado un objeto, es decir, integers, strings o funciones son objetos. Estos objetos tienen asociadas funciones que conoceremos como “métodos”.
* Las variables van por referencia, al igual que en java, Nodejs, js, etc, y el tipo de dato asociado no es estático, es decir, que es de tipado dinámico, sino que puede variar a lo largo del tiempo, esto indica que podemos hacer lo siguiente:

Texto

Descripción generada automáticamente

Como vemos, la variable X no se arraiga a ningún tipo de datos si no que a lo largo de la ejecución va teniendo diferentes tipos, sin embargo, esto no quiere decir que Python sea un lenguaje débilmente tipado, todo lo contrario, Python es considerado un lenguaje **fuertemente tipado.** Entiéndase fuertemente tipado como que no se hace conversiones automáticas de tipos para transformar un tipo de dato a otro y adaptarlo en ciertas operaciones, por ejemplo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En este caso no deja cocatenar un int con un str porque no hay conversión automática de tipos, es decir, que un tipo es un tipo y no se castea de uno a otro fácilmente, para ello, lo que habría que hacer, sería lo siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* En Python podemos usar indiscriminadamente comillas dobles “” o comillas simples ‘’ para representar los tipos de datos string.
* Los comentarios de una línea se hacen con “#”.

## Formateo de texto.

* F””
* “”.format(var1,var2)

## Funciones en Python.

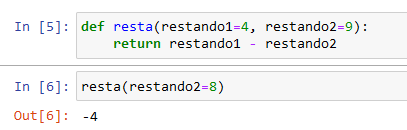
Las funciones en Python se declaran con la palabra def seguido de una etiqueta identificativa de función y entre paréntesis los argumentos necesarios, como vemos en la siguiente imagen, no se especifica ningún tipo de dato de retorno de la función, en la función de la derecha vemos una función sin retorno.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

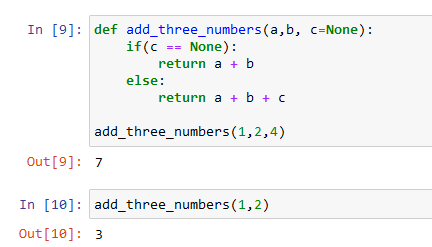
Descripción generada automáticamente

Además, en las funciones en Python los argumentos de las funciones pueden estar definidos tanto posicionalmente como por orden:



En este último ejemplo vemos que podemos saltarnos directamente mencionar el argumento uno y pasar directamente a referirnos al argumento 2 por equivalencia nominal, al no especificar el valor del restando1 tenemos el que se le ha asignado por defecto que es igual a 4.

Podemos ver los valores que se le dan a los argumentos formales de la definición de la función como los valores que se asignaran a esas variables si no se especifican en la llamada.

Cuando no asignamos un “=” al argumento formal de la función, entonces estamos indicando que es obligatorio. Además, los argumentos obligatorios van siempre delante de los argumentos opcionales. A la derecha se ve un ejemplo del uso de esto.

# Numpy.

Numpy es una librería

Suele llamarse np

## Arreglos

### Accesos en arreglos bidimensionales.

X[i] te dará la i-esima

Para obtener la fila i-esima hacemos:

Para obtener la columna j-esima hacemos:

### Operaciones con arreglos

Si tenemos un arreglo np.array([1,2,3,4,5]) podemos hacer la siguientes operaciones:

* x = np.array([1,2,3,4,5]);
* x += a 🡪 Implica sumarle “a” a cada elemento del vector.
* x -= b 🡪 Implica restarle “b” a cada elemento del vector.

## Forma o shape de un arreglo.

Cuando se habla de la "forma" o "shape" de un arreglo en términos de programación, especialmente en el contexto de bibliotecas como NumPy, nos referimos a las dimensiones de este arreglo. Por ejemplo, si tienes una matriz (o un arreglo bidimensional) que tiene 3 filas y 5 columnas, su forma es (3, 5).

En el contexto de un conjunto de datos, una forma (N, D) podría indicar N filas que representan una entidad y D podría representar las características de cada entidad, por ejemplo, podríamos tener una matriz (20, 5) tal que tenemos 20 personas con 5 características cada una (edad, peso, trabajo, estudios e ingresos mensuales).

Además, el shape de un np.array no se limita a dos dimensiones, si no que puede tener 1, 2 o más de dos dimensiones en general.

|  |  |  |
| --- | --- | --- |
|  | Resultado | Aclaraciones |
| np.shape(mi\_array\_o\_matriz) | (N\_filas, N\_columnas,…) |  |
| My\_array\_o\_matriz.shape | (n\_rows, n\_filas, …) | La propiedad shape del objeto np.array es una tupla de longitud “n” siendo “n” el numero de dimensiones del tensor. |

# Información de interes.

* Tanto a las tuplas, listas como los np array se accede a sus valores con corchetes rectangulares.