**1D Numpy en Python**

Objetivos

**\*Importar y usar la biblioteca numpy**

**\*Realizar operaciones con numpy**

**\*Tabla de contenido**

**\*Lista en Python**

**\*¿Qué es Numpy?**

**\*Tipo**

**\*Asignar valor**

**\*rebanar**

**\*Asignar valor con lista**

**\*Otros atributos**

**\*Operaciones de matriz numpy**

**\*Adición de matriz**

**\*Multiplicación de matrices**

**\*Producto de dos matrices Numpy**

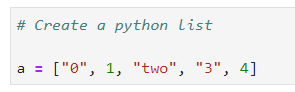
**\*Producto escalar**

**\*Agregar constante a una matriz Numpy**

**\*Funciones Matemáticas**

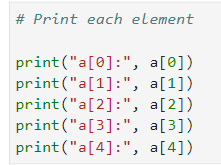
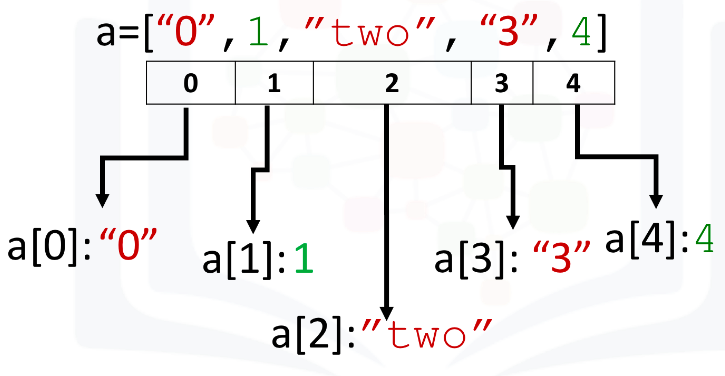
**\*Espacio Linfático**

**Cree una lista de Python de la siguiente manera:**



**Create a python list** **..Crear una lista de Python**

**Podemos acceder a los datos a través de un índice:**



**Podemos acceder a cada elemento usando un corchete de la siguiente manera:**

**Print each element........Imprime cada elemento**

**¿Qué es Numpy?**

**NumPy es una biblioteca de Python que se utiliza para trabajar con matrices, álgebra lineal, transformada de Fourier y matrices.**

**\*Una matriz numpy es similar a una lista.**

**NumPy significa Numerical Python y es un proyecto de código abierto.**

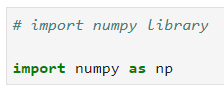
**El objeto de matriz en NumPy se llama ndarray, proporciona muchas funciones de apoyo que hacen que trabajar con ndarray sea muy fácil.**

**Las matrices se utilizan con mucha frecuencia en la ciencia de datos, donde la velocidad y los recursos son muy importantes.**

**NumPy generalmente se importa con el alias np.**

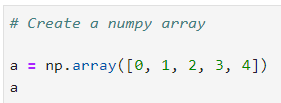
**Por lo general, tiene un tamaño fijo y cada elemento es del mismo tipo.**

**Podemos convertir una lista en una matriz numpy importando primero numpy:**



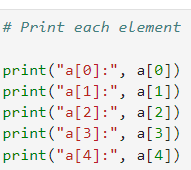
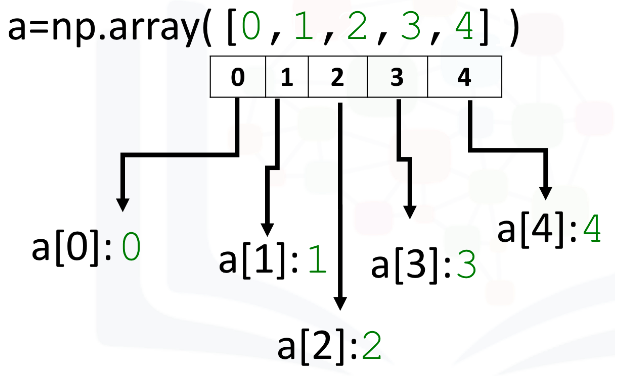
**import numpy library = importar biblioteca numpy**

**A continuación, elaboramos la lista de la siguiente manera:**



**Create a numpy array = Crear una matriz numpy**

**Cada elemento es del mismo tipo, en este caso enteros:**



**Print each element** = **Imprime cada elemento**

**Al igual que con las listas, podemos acceder a cada elemento mediante un corchete:**

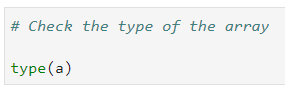
**Comprobación de la versión NumPy**

**La cadena de versión se almacena en el atributo de versión.**

**print(np.\_\_version\_\_) = imprimir (np. \_\_ versión \_\_)**

**Tipo**

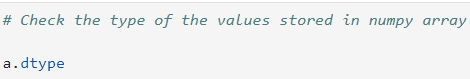
**Si verificamos el tipo de matriz, obtenemos numpy.ndarray:**



**Check the type of the array = Comprobar el tipo de matriz**

**Como las matrices numpy contienen datos del mismo tipo, podemos usar el atributo "dtype" para obtener el tipo de datos de los elementos de la matriz.**

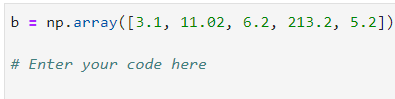
**En este caso, es un entero de 64 bits:**

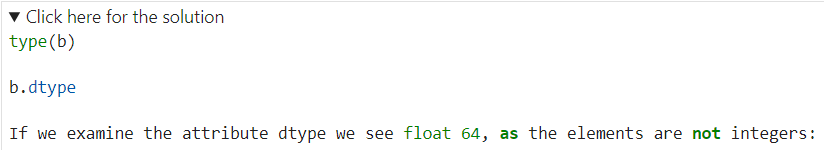


**Check the type of the values stored in numpy array = Verifique el tipo de los valores almacenados en la matriz numpy**

**Inténtalo tú mismo**

**Verifique el tipo de matriz y el tipo de valor para la matriz dada c**

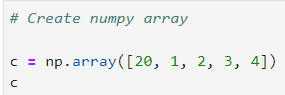




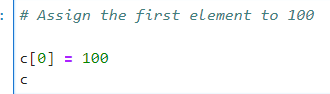
**Si examinamos el atributo dtype vemos float 64, ya que los elementos no son enteros: (FLOAT CON DECIMALES)**

**Asignar valor**

**Podemos cambiar el valor de la matriz. Considere la matriz c:**

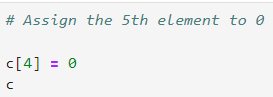
**Create numpy array** **= Crear matriz numpy**

**Podemos cambiar el primer elemento de la matriz a 100 de la siguiente manera:**



**# Assign the first element to 100** **= # Asignar el primer elemento a 100**

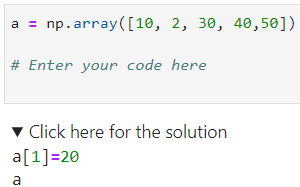
**Podemos cambiar el quinto elemento de la matriz a 0 de la siguiente manera:**



**Assign the 5th element to 0** = **Asigne el quinto elemento a 0**

**Inténtalo tú mismo**

**Asigne el valor 20 para el segundo elemento en la matriz dada.**



**IMPORTANTE ENTRE CORCHETES LA UBICACION DE LOS NUMEROS SE CUENTA EL 0 COMO EL 1, EL SEGUNDO NUMERO COMO EL TERCERO , ETC**

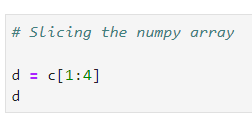
**Rebanar**

**Al igual que las listas, podemos dividir la matriz numpy.**

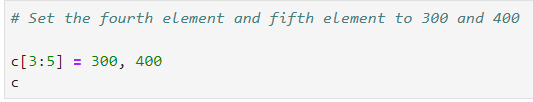
**Cortar en python significa tomar los elementos del índice dado a otro índice dado.**

**Pasamos un segmento como este: [inicio: final]. El elemento en el índice final no se incluye en la salida.**

**Podemos seleccionar los elementos del 1 al 3 y asignarlos a una nueva matriz numpy d de la siguiente manera:**

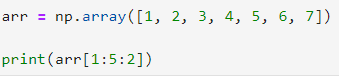
**Slicing the numpy array = Cortando la matriz numpy**

**Podemos asignar los índices correspondientes a los nuevos valores de la siguiente manera:**



**Set the fourth element and fifth element to 300 and 400** **= Establezca el cuarto elemento y el quinto elemento en 300 y 400**

**También podemos definir los pasos en el corte, así: [start:end:step].**



**Si no pasamos start se considera 0**



**Si no pasamos end, considera hasta la longitud de la matriz.**

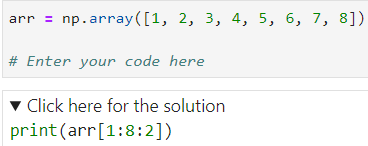


**Si no pasamos el step se considera 1**



**Inténtalo tú mismo**

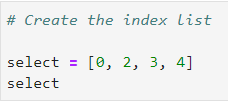
**Imprime los elementos pares en la matriz dada.**



**Asignar valor con lista**

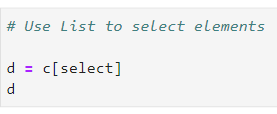
**De manera similar, podemos usar una lista para seleccionar más de un índice específico.**

**La selección de lista contiene varios valores:**

**Create the index list= Crear la lista de índice**

**Podemos usar la lista como argumento entre paréntesis.**

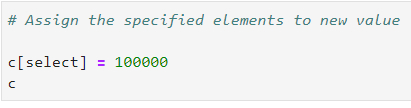
**La salida son los elementos correspondientes a los índices particulares:**



**Use List to select elements = Usar lista para seleccionar elementos**

**Podemos asignar los elementos especificados a un nuevo valor.**

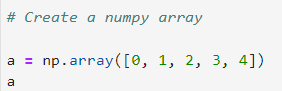
**Por ejemplo, podemos asignar los valores a 100 000 de la siguiente manera:**



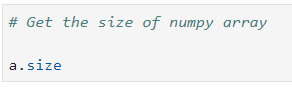
**Assign the specified elements to new value = Asignar los elementos especificados a un nuevo valor**

**Otros atributos**

**Revisemos algunos atributos básicos de la matriz usando la matriz a:**

**Create a numpy array= Crear una matriz numpy**

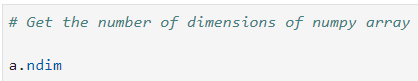
**El tamaño del atributo es el número de elementos en la matriz:**



**Get the size of numpy array= Obtenga el tamaño de la matriz numpy**

**Los siguientes dos atributos tendrán más sentido cuando lleguemos a dimensiones más altas, pero repasémoslos.**

**El atributo ndim representa el número de dimensiones de la matriz o el rango de la matriz. En este caso, uno:**



**Get the number of dimensions of numpy array = Obtener el número de dimensiones de la matriz numpy**

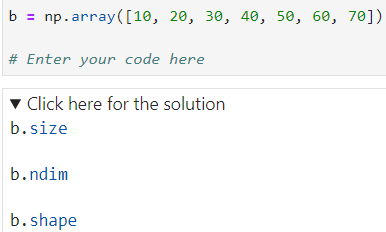
**La forma del atributo es una tupla de enteros que indica el tamaño de la matriz en cada dimensión:**

**shape = forma**

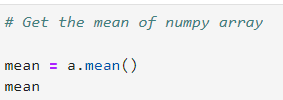
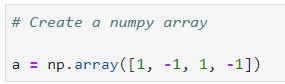
**Get the shape/size of numpy array = Obtenga la forma/tamaño de la matriz numpy**

**Inténtalo tú mismo**

**Encuentre el tamaño, la dimensión y la forma de la matriz dada b**



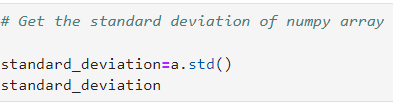
**Funciones estadísticas numpy3**



**Create a numpy array = # Crear una matriz numpy**

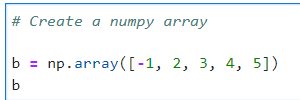
**Get the mean of numpy array =Obtenga la media de la matriz numpy**

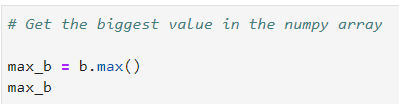
**Mean = significar array =formación**



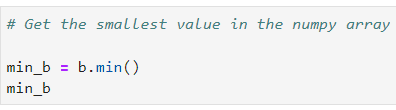
**Get the standard deviation of numpy array = Obtenga la desviación estándar de la matriz numpy**

**standard\_deviation = Desviación Estándar**

**Create a numpy array = Crear una matriz numpy**



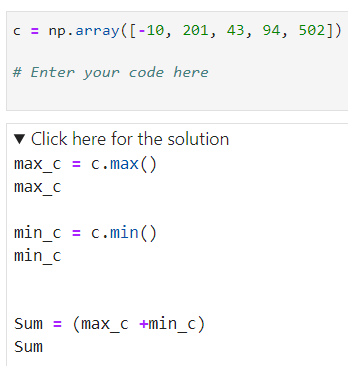
**Get the biggest value in the numpy array = Obtenga el mayor valor en la matriz numpy**



**Get the smallest value in the numpy array = Obtenga el valor más pequeño en la matriz numpy**

**Inténtalo tú mismo**

**Encuentre la suma del valor máximo y mínimo en la matriz numpy dada**

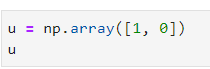
**Enter your code here= Introduce tu código aquí**

**Operaciones de matriz numpy**

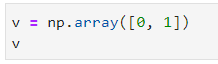
**Podría usar operadores aritméticos directamente entre matrices NumPy**

**Adición de matriz**

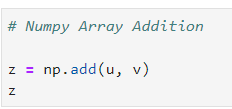
**Considere la matriz numpy u:**



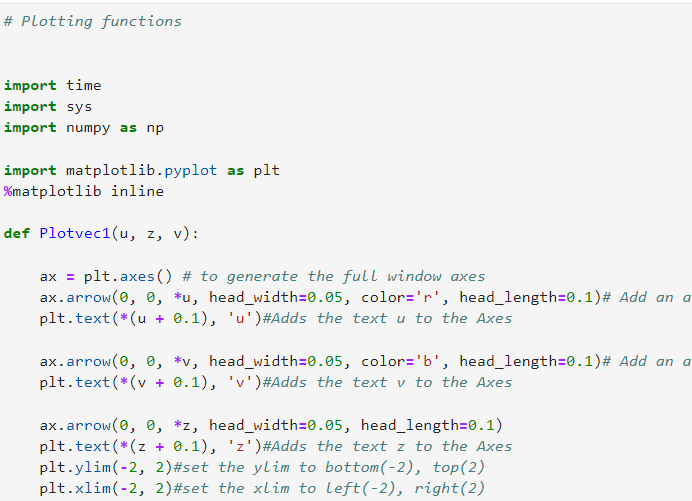
**Considere la matriz numpy v:**



**Podemos sumar las dos matrices y asignarlas a z:**

**Numpy Array Addition = Adición de matriz Numpy**

**La operación es equivalente a la suma de vectores:**



**Plotting functions = Funciones de trazado**

**import time = tiempo de importación**

**import sys = sistema de importación**

**import numpy as np = importar numpy como np**

**import matplotlib .pyplot as plt = importar matplotlib .pyplot como plt**

**%matplotlib inlin e= % matplotlib en línea**

**def Plotvec1(u, z, v):**

**ax = plt.axes() # para generar los ejes de la ventana completa**

**ax.arrow(0, 0, \*u, head\_width=0.05, color='r', head\_length=0.1)# Agregue una flecha a los ejes U con**

**ancho de punta de flecha 0.05, color rojo y longitud de punta de flecha 0.1**

**plt.text(\*(u + 0.1), 'u')#Agrega el texto u a los Ejes**

**ax.arrow(0, 0, \*v, head\_width=0.05, color='b', head\_length=0.1)# Agrega una flecha a los ejes v con**

**ancho de punta de flecha 0.05, color rojo y longitud de punta de flecha 0.1**

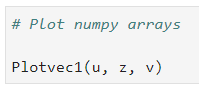
**plt.text(\*(v + 0.1), 'v')#Agrega el texto v a los Ejes**

**ax.arrow(0, 0, \*z, head\_width=0.05, head\_length=0.1)**

**plt.text(\*(z + 0.1), 'z')#Agrega el texto z a los Ejes**

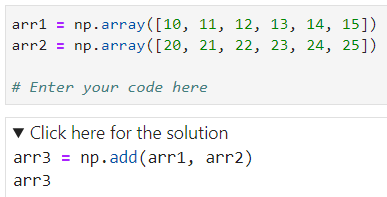
**plt.ylim(-2, 2)#establecer el ylim en la parte inferior (-2), superior (2)**

**plt.xlim (-2, 2) # establece el xlim a la izquierda (-2), derecha (2)**

**Plot numpy arrays** = **Trazar matrices numpy**

**Inténtalo tú mismo**

**Realice la operación de suma en la matriz numpy dada arr1 y arr2:**



**Sustracción de matrices (resta)**

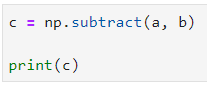
**Considere la matriz numpy a:**

**ARRAY = FORMACIÓN**

**Considere la matriz numpy b:**

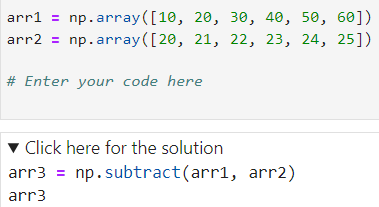


**Podemos restar las dos matrices y asignarlas a c:**

**subtract = sustraer**

**Inténtalo tú mismo**

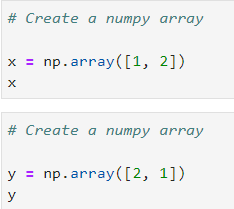
**Realice la operación de resta en la matriz numpy dada arr1 y arr2:**



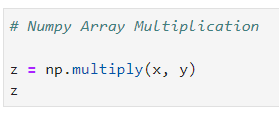
**ARRAY = FORMACIÓN subtract = sustraer**

**Multiplicación de matrices**

**Considere la matriz vectorial numpy y:**

**Create a numpy array = Crear una matriz numpy**

**Podemos multiplicar cada elemento de la matriz por 2:**

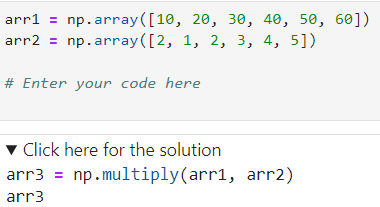


**Numpy Array Multiplication = Multiplicación de matrices numéricas**

**Esto es equivalente a multiplicar un vector por un escalador:**

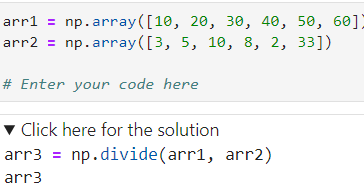
**Inténtalo tú mismo**

**Realice la operación de multiplicación en la matriz numpy dada arr1 y arr2:**



**División de matriz**

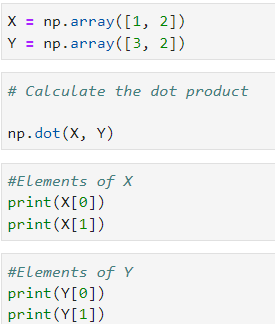
**Considere la matriz vectorial numpy a:**



**Enter your code here = Introduce tu código aquí**

**Producto escalar**

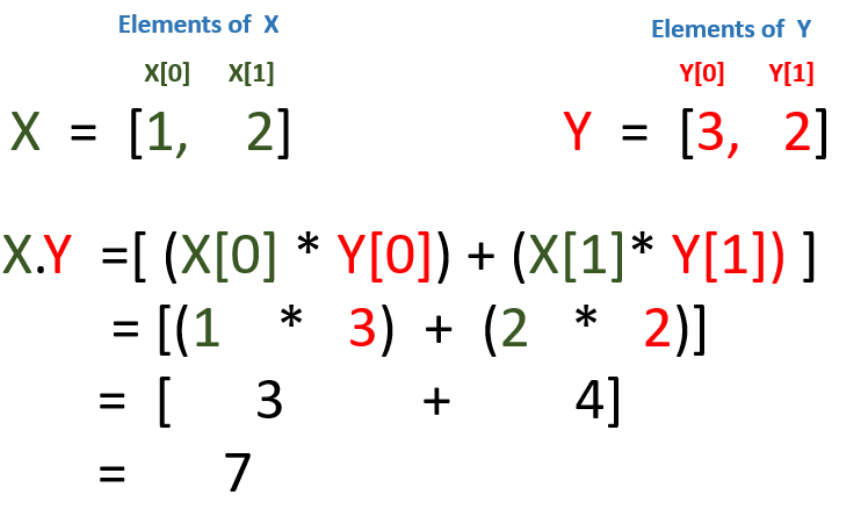
**El producto escalar de las dos matrices numéricas u y v viene dado por:**



**Calculate the dot product = Calcular el producto escalar**

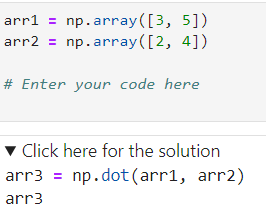
**Elements of Y = Elementos de Y**

**Estamos realizando el producto escalar que se muestra a continuación**



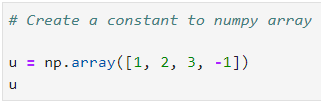
**Inténtalo tú mismo**

**Realice la operación de punto en la matriz numpy dada ar1 y ar2:**



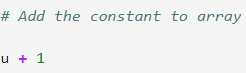
**Agregar constante a una matriz Numpy**

**Considere la siguiente matriz:**



**Create a constant to numpy array = Crear una matriz constante a numpy**

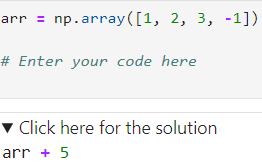
**Agregando la constante 1 a cada elemento de la matriz:**



**Add the constant to array = Agrega la constante a la matriz**

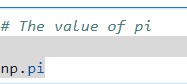
**Inténtalo tú mismo**

**Agregue Constant 5 a la matriz numpy dada ar:**

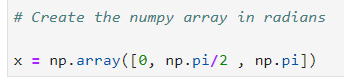


**Funciones Matemáticas**

**Podemos acceder al valor de pi en numpy de la siguiente manera:**

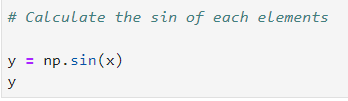
**The value of pi = el valor de pi**

**Podemos crear la siguiente matriz numpy en radianes:**



**Create the numpy array in radians = Crea la matriz numpy en radianes**

**Podemos aplicar la función sin al arreglo x y asignar los valores al arreglo y; esto aplica la función seno a cada elemento de la matriz:**



**Calculate the sin of each elements = Calcular el seno de cada elemento**

**Espacio Linfático**

**Una función útil para trazar funciones matemáticas es linspace.**

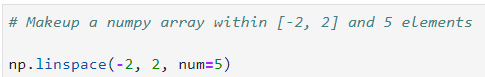
**Linspace devuelve números espaciados uniformemente en un intervalo específico.**

**numpy.linspace(inicio, parada, num = valor int)**

**inicio : inicio del rango de intervalo**

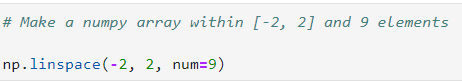
**detener: fin del rango de intervalo**

**num : Número de muestras a generar.**

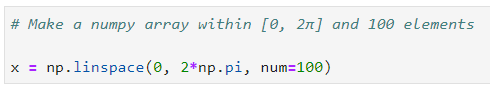


**Makeup a numpy array within [-2, 2] and 5 elements = Cree una matriz numpy dentro de [-2, 2] y 5 elementos**

**Si cambiamos el parámetro num a 9, obtenemos 9 números espaciados uniformemente en el intervalo de -2 a 2:**

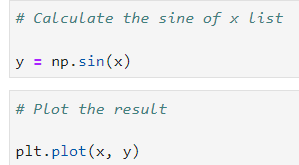


**Make a numpy array within [-2, 2] and 9 elements = # Hacer una matriz numpy dentro de [-2, 2] y 9 elementos**



**Make a numpy array within [0, 2π] and 100 elements = # Hacer una matriz numpy dentro de [0, 2π] y 100 elementos**

**Podemos aplicar la función seno a cada elemento de la matriz x y asignarla a la matriz y:**

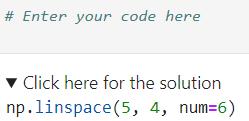


**Calculate the sine of x list = Calcular el seno de x lista**

**Plot the result = Graficar el resultado**

**Inténtalo tú mismo**

**Haz una matriz numpy dentro de [5, 4] y 6 elementos**



**Linspace = espacio lineal**

**Iterando arreglos 1-D**

**Iterar significa pasar por los elementos uno por uno.**

**Si iteramos en una matriz 1-D, pasará por cada elemento uno por uno.**

**Si ejecutamos la matriz numpy, obtenemos el formato de matriz**

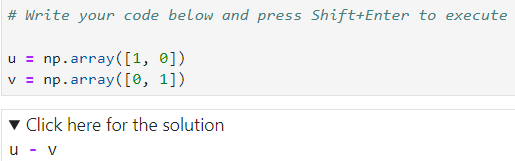


**Pero si desea obtener la forma de la lista, puede usar for loop: (for loop**: = **EN BUCLE)**

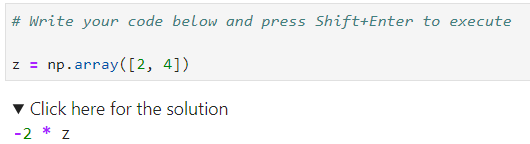


**Cuestionario sobre 1D Numpy Array**

**Implemente la siguiente resta vectorial en numpy: u-v**

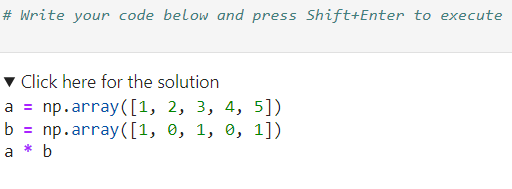


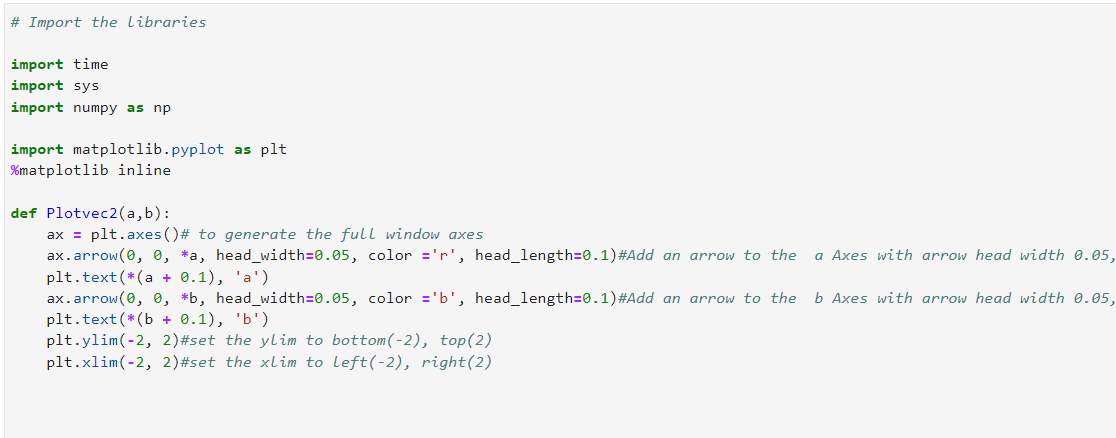
Multiply the numpy array z with -2



**Write your code below and press Shift+Enter to execute = Escriba su código a continuación y presione Shift+Enter para ejecutar:**

**Considere la lista [1, 2, 3, 4, 5] y [1, 0, 1, 0, 1]. Transmita ambas listas a una matriz numpy y luego multiplíquelas juntas:**





**Importar las bibliotecas**

**tiempo de importación**

**sistema de importación**

**importar numpy como np**

**importar matplotlib.pyplot como plt**

**% matplotlib en línea**

**def Plotvec2(a,b):**

**ax = plt.axes()# para generar los ejes de la ventana completa**

**ax.arrow(0, 0, \*a, head\_width=0.05, color ='r', head\_length=0.1)#Agregue una flecha a los ejes con ancho de punta de flecha 0.05, color rojo y longitud de punta de flecha 0.1**

**plt.text(\*(a + 0.1), 'a')**

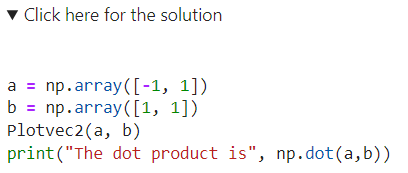
**ax.arrow(0, 0, \*b, head\_width=0.05, color ='b', head\_length=0.1)#Agregue una flecha a los ejes b con ancho de punta de flecha 0.05, color azul y longitud de punta de flecha 0.1**

**plt.text(\*(b + 0.1), 'b')**

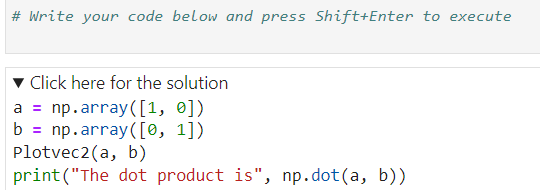
**plt.ylim(-2, 2)#establecer el ylim en la parte inferior (-2), superior (2)**

**plt.xlim (-2, 2) # establece el xlim a la izquierda (-2), derecha (2)**

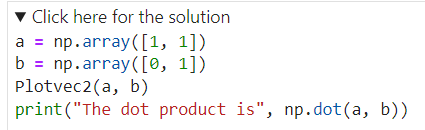
**Convierta la lista [-1, 1] y [1, 1] en matrices numéricas a y b. Luego, trace las matrices como vectores usando la función Plotvec2 y encuentre su producto escalar:**



**Convierta la lista [1, 0] y [0, 1] en matrices numéricas a y b. Luego, trace las matrices como vectores usando la función Plotvec2 y encuentre su producto escalar:**



**Convierta la lista [1, 1] y [0, 1] en matrices numéricas a y b. Luego grafica las matrices como vectores usando la función Plotvec2 y encuentra su producto punto:**

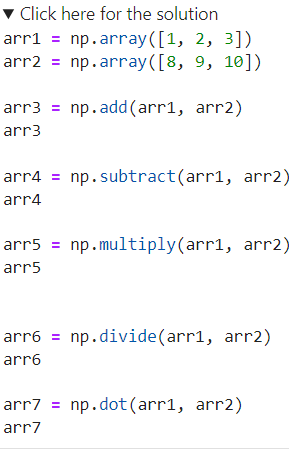


**¿Por qué los resultados del producto escalar para [-1, 1] y [1, 1] y el producto escalar para [1, 0] y [0, 1] son ​​cero, pero no cero para el producto escalar para [1, 1] y [0, 1]?**

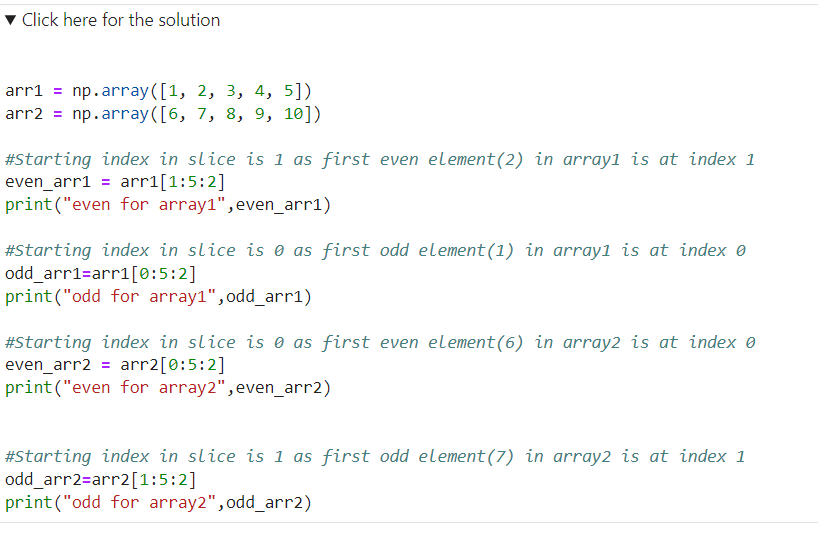
**Sugerencia: estudie las figuras correspondientes, preste atención a la dirección hacia la que apuntan las flechas.**

**Los vectores usados ​​para las preguntas 4 y 5 son perpendiculares. Como resultado, el producto escalar es cero.**

**Convierta la lista [1, 2, 3] y [8, 9, 10] en matrices numpy arr1 y arr2. Luego realice operaciones de suma, resta, multiplicación, división y punto en arr1 y arr2.**



**Convierta la lista [1, 2, 3, 4, 5] y [6, 7, 8, 9, 10] en arreglos numéricos arr1 y arr2. Luego encuentra los números pares e impares de arr1 y arr2.**



**matriz1 = np.matriz([1, 2, 3, 4, 5])**

**matriz2 = np.matriz([6, 7, 8, 9, 10])**

**#El índice inicial en el sector es 1 ya que el primer elemento par (2) en el arreglo 1 está en el índice 1**

**par\_arr1 = arr1[1:5:2]**

**imprimir("incluso para matriz1",even\_arr1)**

**#El índice inicial en el segmento es 0 ya que el primer elemento impar (1) en el arreglo1 está en el índice 0**

**impar\_arr1=arr1[0:5:2]**

**imprimir ("impar para matriz1", impar\_arr1)**

**#El índice inicial en el sector es 0 ya que el primer elemento par (6) en el arreglo2 está en el índice 0**

**even\_arr2 = arr2[0:5:2]**

**imprimir("incluso para matriz2",even\_arr2)**

**#El índice inicial en el segmento es 1 ya que el primer elemento impar (7) en el arreglo2 está en el índice 1**

**impar\_arr2=arr2[1:5:2]**

**imprimir ("impar para matriz2", impar\_arr2)**