GOBIERNO DEL
ESTADO DE MÉXICO

MANUAL DE PRÁCTICAS

Nombre de la práctica	Manual de Practicas Numpy en Anaconda			**No.**	**1**
Asignatura:	Simulación 3501	**Carrera:**	Ingeniería en Sistemas Computacionales	**Duración de la práctica (Hrs)**	**8**

I. Competencia(s) específica(s):

Alumno: Raúl Ciriaco Castillo 3501

Github:

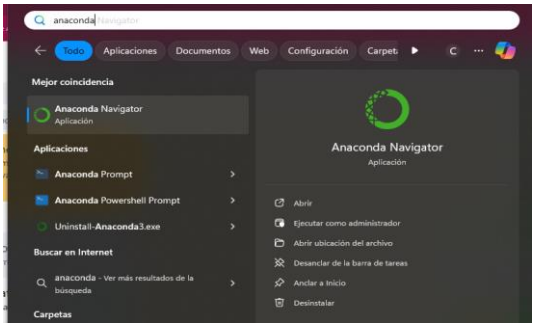
Comentado [T1]: https://github.com/RaulCiriaco

II.Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

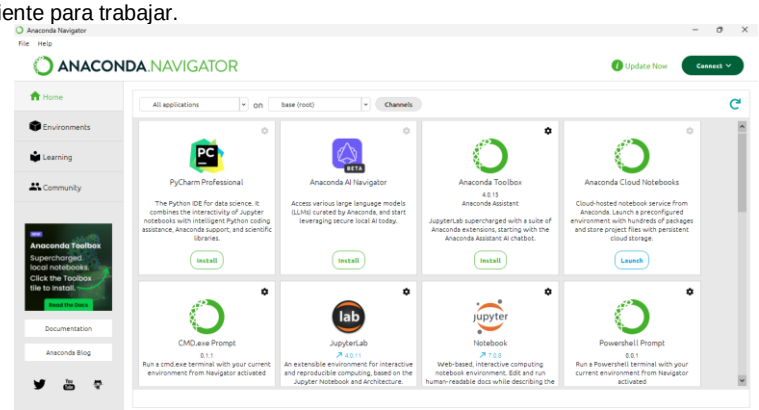
III. Material empleado:

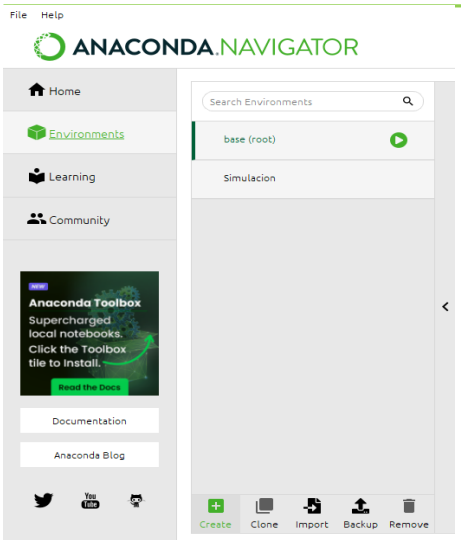
Desarrollo de la práctica:

Como primeros pasos debemos instalar Anaconda Studio en nuestro Equipo de Computo que vamos a utilizar



Una vez que instalamos las paqueterías y todo lo requerido abrimos Anaconda y procedemos a crear nuestro ambiente para trabajar.

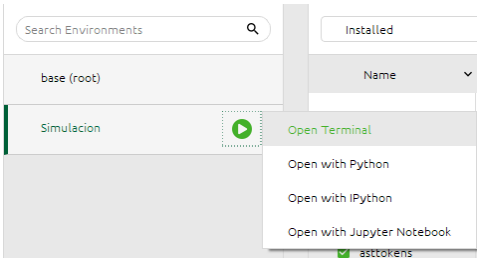




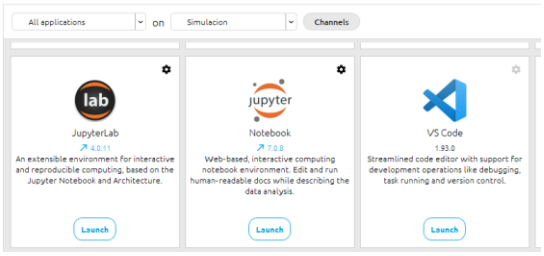
Debemos dar click en el apartado “Environments” y abajo pulsamos el “Create”

Después pulsamos en Open Terminal y vamos a proceder a instalar las paqueterías:

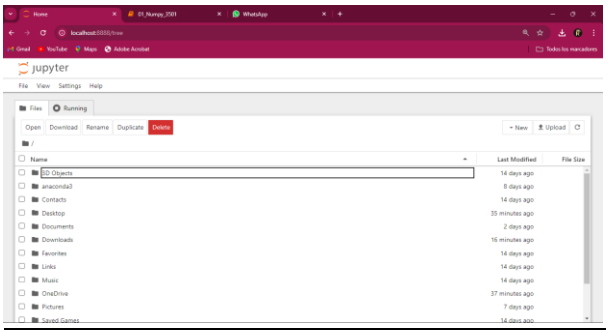
- Numpy
- Pandas
- Matplotlib
- SkLearn



Vamos a descargar el Jupyter Notebook y pulsamos en Launch.

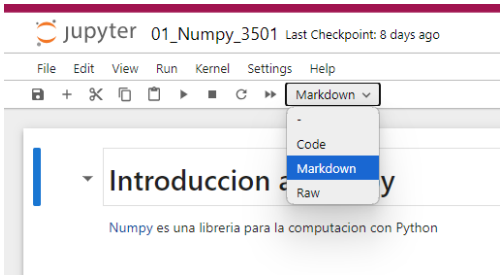


Nos abrirá esta pantalla donde podremos navegar libremente, crear archivos, modificar y eliminarlos.



Crearemos un nuevo archivo para comenzar con la librería “Numpy”.

En este apartado principal tendremos las herramientas que podemos utilizar en nuestro archivo, tenemos las opciones que se muestran en la imagen



- **Code:** Este apartado nos sirve para poder realizar instrucciones que queramos ejecutarlas después, que cumplan con alguna condición o no, cualquier actividad que tengamos que codificar se verá reflejada en este espacio.
- **Markdown:** Este nos permite la escritura de cualquier información, ya sean formulas o códigos para ejemplificar, no funciona si queremos mostrar o imprimir algún dato o valor.
- **Raw:**

Comenzaremos con una línea que nos permitirá enlazar con la página principal de la librería, en la cual podemos acceder para conocer información acerca de la misma, cualquier duda que tengamos podemos encontrarla en la página oficial.

Introduccion a Numpy

Numpy es una libreria para la computacion con Python

- Proporciona Arrays N-Dimensiones.
- Implemeta funciones matemáticas sofisticadas.
- Proporciona herramientas para integrar C/C++ y Frontan.
- Proporciona mecanismos para facilitar la realizacion de tareas relacionadas con álgebra lineas o numeros aleatorios.

Para comenzar a trabajar, necesariamente debemos importar la paquetería de Numpy.

```
[]: import numpy as np
```

Como primera actividad en clase, comenzamos con la creación de arrays como ejemplo para posteriormente conocer sus características como las mostramos en pantalla.

```
[2]: #Array cuyos valores sean todos 0.
     a = np.zeros((2,4))
     a

[2]: array([[0., 0., 0., 0.],
            [0., 0., 0., 0.]])
```

a es un array:

- Con dos **axis**, el primero de longitud 2 y el segundo de longitud 4.
- Con un **rang** igual a 2.
- Con un **shape** igual (2,4).
- Con un **size** igual a 8.

```
[3]: a.shape

[3]: (2, 4)

[4]: a.ndim

[4]: 2

[5]: a.size

[5]: 8
```

El primer tema será **"Creación de Arrays"** este tema estará enfocado en trabajar la creación de arreglos, modificación, operaciones en ellos, valores máximos y mínimos e inicialización.

Todo esto se lo mostramos en las siguientes pantallas:

```
[6]: # Array cuyos valores sean todos 0,
     np.zeros((2, 3, 4))

[6]: array([[[0., 0., 0., 0.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.]],

            [[0., 0., 0., 0.],
             [0., 0., 0., 0.],
             [0., 0., 0., 0.]]])

[7]: # Array cuyos valores sean todos 1
     np.ones((2, 3, 4))

[7]: array([[[1., 1., 1., 1.],
             [1., 1., 1., 1.],
             [1., 1., 1., 1.]],

            [[1., 1., 1., 1.],
             [1., 1., 1., 1.],
             [1., 1., 1., 1.]]])
```

```
[8]: # Array cuyos valores son todos el valor indicado como segundo parametro de la funcion
     np.full((2, 3, 4), 8)
```

```
[8]: array([[[8, 8, 8, 8],
             [8, 8, 8, 8],
             [8, 8, 8, 8]],

            [[8, 8, 8, 8],
             [8, 8, 8, 8],
             [8, 8, 8, 8]]])
```

```
[9]: # El resultado de np.emty no es redecible
     # Se inicializa con los valores de array con lo que haya en memoria en ese momento
     np.empty((2,3,9))
```

```
[9]: array([[[1.84115564e-314,             nan, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
            [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
            [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000]],

           [[0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
            [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
            [0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
             0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
                        nan, 0.00000000e+000, 1.48219694e-323]]])
```

```
[10]: # Inicializar el array utilizando un array de python.
      b= np.array ([[1,2,3],[4,5,6]])
      b
```

```
[10]: array([[1, 2, 3],
            [4, 5, 6]])
```

```
[11]: b.shape
```

```
[11]: (2, 3)
```

```
[12]: # Crear un array utilizandi una funcion basada en rangos
      # (minimo, maximo, numero elementos del array)
      print(np.linspace(0,6,10))
```

```
[0.         0.66666667 1.33333333 2.         2.66666667 3.33333333
 4.         4.66666667 5.33333333 6.        ]
```

```
[12]: # Crear un array utilizandi una funcion basada en rangos
      # (minimo, maximo, numero elementos del array)
      print(np.linspace(0,6,10))
```

```
[0.         0.66666667 1.33333333 2.         2.66666667 3.33333333
 4.         4.66666667 5.33333333 6.        ]
```

```
[13]: # Inicializar el array con valorea aleatoreos
      np.random.rand(3,3,4)
```

```
[13]: array([[[0.41293036, 0.54371091, 0.77587257, 0.95983259],
              [0.06019597, 0.03871104, 0.07840317, 0.10704618],
              [0.51671952, 0.34153451, 0.5604085 , 0.87307191]],

             [[0.62464635, 0.6341071 , 0.76395578, 0.19393566],
              [0.95959796, 0.82317932, 0.73390214, 0.56942981],
              [0.18536062, 0.52116856, 0.58163634, 0.8648931 ]],

             [[0.58574545, 0.53503759, 0.49688342, 0.50312515],
              [0.63812376, 0.78982076, 0.54333177, 0.31718685],
              [0.12426138, 0.90522124, 0.47387268, 0.67995374]]])
```

```
[14]: # inicializar array con alores aleatorios conforme a una distribucion normal.
      np.random.randn(2,4)
```

```
[14]: array([[-0.33749626,  0.9288522 , -1.78278607, -0.05693868],
             [-0.127037  ,  1.08167394,  0.90361916, -0.22692818]])
```

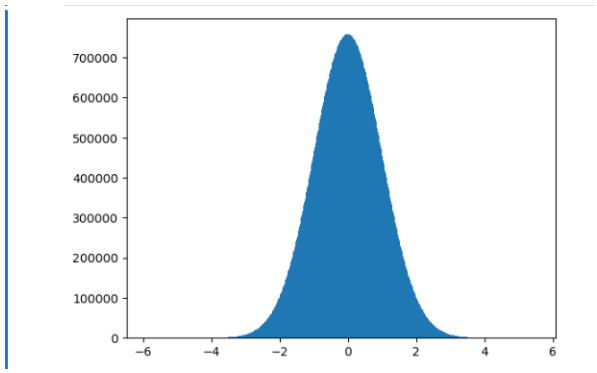
Una vez que tenemos nuestro arrays y trabajamos con diferentes ejemplos y ejercicios, también podemos graficarlos para analizar su comportamiento mediante las siguientes líneas.

Estas líneas prácticamente indican la importación de una nueva librería llamada Matplotlib que nos permitirá graficar funciones que necesitemos.

```
%matplotlib inline
import matplotlib.pyplot as plt

c= np.random.randn(100000000)
plt.hist(c, bins=600)
plt.show()
```

El resultado de nuestra grafica es el siguiente:



El siguiente tema que vamos a abordar dentro de esta librería de Numpy será:

Acceso a los elementos de un array (Array unidimensional).

Acceso a los elementos de un array

Array unidimencional

```
[17]: # Acceder a los elementos de un array
      array_uni = np.array( [1 ,3 ,5 ,7 ,9 ,11 ])
      print("Shape:", array_uni.shape)
      print("Array_uni:", array_uni)

      Shape: (6,)
      Array_uni: [ 1  3  5  7  9 11]

[18]: # Accedio al quinto elemento de array
      array_uni[4]

[18]: 9

[19]: #Acceder al tercer y cuarto elemento del array
      array_uni[2:4]

[19]: array([5, 7])
```

Array Multidimencional

```
[0]: # Crear una array multidimencional
array_multi = np.array ([[1, 2 , 3, 4],[5, 6, 7, 8]])
print("shape:",array_multi.shape)
print("Array_multi:\n",array_multi)
```

```
shape: (2, 4)
Array_multi:
 [[1 2 3 4]
 [5 6 7 8]]
```

```
[1]: #Acceder al cuarto elemento del array
array_multi[0,3]
```

```
[1]: 4
```

```
[2]: #Aceder a la segunda fila del array
array_multi[1,:]
```

```
[2]: array([5, 6, 7, 8])
```

```
[3]: ## acceder al primer elemento de las dos primeras filas del array
array_multi[0:2,2]
```

```
[3]: array([3, 7])
```

Modificacion de un array

```
# crear un arreglo unidimencional e inicializarlo con un rango
# de elementos 0-27
array1 = np.arange(28)
print("shape:",array1.shape)
print("Array:\n",array1)
```

```
shape: (28,)
Array:
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27]
```

```
# cambiar lasdimenciones del arry y sus longitudes
array1.shape=(7,4)
print("shape:",array1.shape)
print("Array:\n",array1)
```

```
shape: (7, 4)
Array:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]
 [24 25 26 27]]
```

[27]:
```
# Modificacion del nuevo array devuelto
array2[1,3] = 30
print("Array2:\n",array2)
```

```
Array2:
 [[ 0  1  2  3  4  5  6]
 [ 7  8  9 30 11 12 13]
 [14 15 16 17 18 19 20]
 [21 22 23 24 25 26 27]]
```

[28]:
```
print("Array1:\n",array1)
```

```
Array1:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 30 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]
 [24 25 26 27]]
```

[29]:
```
# Devolver el array a su estado original
print("Array1:",array1.ravel())
```

```
Array1: [ 0  1  2  3  4  5  6  7  8  9 30 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27]
```

Operaciones aritmeticas con Arrays

[30]:
```
array1 = np.arange (2,18,2)
array2 = np.arange(8)
print("Array 1:\n",array1)
print("Array 2:\n",array2)
```

```
Array 1:
 [ 2  4  6  8 10 12 14 16]
Array 2:
 [0 1 2 3 4 5 6 7]
```

[31]:
```
# Suma
print(array1+array2)
```

```
[ 2  5  8 11 14 17 20 23]
```

[32]:
```
# Resta
print(array1-array2)
```

```
[2 3 4 5 6 7 8 9]
```

[33]:
```
# Multiplicacion
# Nota : no es una multiplicacionde matrices.
print(array1*array2)
```

```
[  0   4  12  24  40  60  84 112]
```

Broadcasting

Si se aplican operaciones aritmeticas sobre arrays que no tienen la misma forma (shape), Numpy aplica una propiedad que se llama Broadcasting.

[34]:
```
array1 = np.arange(5)
array2 = np.array([3])
print("shape:",array1.shape)
print("Array1:\n",array1)
print("\n")
print("shape:",array2.shape)
print("Array:\n",array2)
```

```
shape: (5,)
Array1:
 [0 1 2 3 4]


shape: (1,)
Array:
 [3]
```

[35]:
```
# Suma de amos Arrays
array1+array2
```

[35]:
```
array([3, 4, 5, 6, 7])
```

Funciones estadisticas sobre arrays

[37]:
```
# Creacion de un array unidimensional
array1 = np.arange(1,20,2)
print("Array1:\n",array1)
```

```
Array1:
 [ 1  3  5  7  9 11 13 15 17 19]
```

[38]:
```
# Media de los elementos del array
array1.mean()
```

[38]:
```
10.0
```

[39]:
```
# Suma de los elementos del Array
array1.sum()
```

[39]:
```
100
```

Y como tema final “Funciones Universales Proporcionadas por numpt: ufunc”

Funciones universales proporcionadas por numpt:ufunc

```
[40]: # Cuadrado de los elementos del array
      np.square(array1)
```

```
[40]: array([  1,   9,  25,  49,  81, 121, 169, 225, 289, 361])
```

```
[41]: # Raiz cuadrada de los elementos del array
      np.sqrt(array1)
```

```
[41]: array([1.        , 1.73205081, 2.23606798, 2.64575131, 3.        ,
             3.31662479, 3.60555128, 3.87298335, 4.12310563, 4.35889894])
```

```
[42]: # Exponencial de los elementos del array.
      np.exp(array1)
```

```
[42]: array([2.71828183e+00, 2.00855369e+01, 1.48413159e+02, 1.09663316e+03,
             8.10308393e+03, 5.98741417e+04, 4.42413392e+05, 3.26901737e+06,
             2.41549528e+07, 1.78482301e+08])
```

```
[43]: # log de los elementos del array
      np.log(array1)
```

```
[43]: array([0.        , 1.09861229, 1.60943791, 1.94591015, 2.19722458,
             2.39789527, 2.56494936, 2.7080502 , 2.83321334, 2.94443898])
```

FIN

IV. Conclusiones:

La verdad este tema que vimos durante las clases nos ayudó a fortalecer mucho nuestro conocimiento sobre Anaconda, trabajando con esta librería, cabe recalcar que semestres pasados el docente aprovechaba un rato para impartirnos este conocimiento en clase y eso fue clave fundamental para poder facilitar este trabajo en el semestre presente, sinceramente esperamos seguir trabajando de esta manera con el docente ya que es una forma muy manejable y muy buena para entender y comprender las clases, algo ameno que como estudiantes hace que podamos comprender cada tema y así aprovechar cualquier momento para preguntar al maestro si en algún momento tenemos dudas sobre el tema.

El uso de las herramientas que nos proporciona el docente para trabajar facilitan la comprensión del tema en su extensión, mediante Anaconda Navigator y el uso de distintas librerías podemos analizar y comprender el movimiento sobre los datos que queramos analizar, todo eso con el fin de recopilar información útil para cada situación.