CHULETAS PYTHONERAS

CREACIÓN Y MANIPULACIÓN ARRAYS NUMPY	
import numpy as np	**Importar** numpy
v1 = np.asarray([0, 1, 2, 3, 4]) v1 = np.array([0, 1, 2, 3, 4]) v1 = np.asanarray(lista1)	Conversión de una **lista en array**.
v1 = np.arange(0, 10, 0.1)	Array de valores comprendidos entre 0 y 10, con un **intervalo de 0.0 a 9.9**. 100 valores.
v1.ndim	**Dimensiones** del array.
v1.shape	Forma **(filas, columnas)**.
v1.size	Número **elementos totales**.
v1.dtype	**Tipo** de elementos.
v2= v1.reshape((1, 8))	Crea un nuevo array con una **forma distinta** (1, 8) con los **elementos de v1**.
v2= np.reshape(v1, [1, 8])	*Igual que el anterior.
np.ones(10)	**Array de unos** de dimensión (1, 10)
np.zeros([2, 2])	**Array de ceros** de dimensión (2, 2)
np.eyes(10)	Array **identidad**.
np.empty([2,2])	Array **aleatorio**.
np.concatenate(v1, v2)	**Concatenar** dos arrays. Ojo con las shapes!
np.split(v1, ndiv)	**Divide** el array v1 en ndiv trozos. Ojo con la división!
np.append(v1, [1, 3])	**Añade** los elementos [1, 3] **al final** del array.
np.max(v1) np.min(v1)	**Máximo y mínimo** de v1.

SLICING ARRAYS NUMPY	
v1[0]	**Primer** elemento de v1
v1[-1]	**Último** elemento de v1.
v1[:]	**Todos** los elementos de v1.
v2[0][3]	**Cuarto** elemento de la **primera fila**.
v2[0][1:3]	Elementos en el **intervalo [1, 3)** de la **primera fila**.

FUNCIONES UNIVERSALES	
np.mean(v1) np.mean(v1, axis = 0) np.mean(v1, axis = 1)	**Media** de los elementos de v1. **Media** de los elementos de v1, **considerando las columnas**. **Media** de los elementos de v1, **considerando las filas**.
np.std(v1), np.sqrt(v1), np.cos(v1), np.sin(v1)	**Desviación típica, raíz cuadrada, coseno y seno de v1**.
np.vectorize(func)	**"Vectorización"** de la función func.
DATOS DESDE UN ARCHIVO	
Datos = np.loadtxt("archivo.txt", delimiter = ',')	**Obtener los datos del archivo.**
np.savetxt("resultados.txt", datos, delimiter= ',')	**Guardar un array en un archivo.**

Realizado por Daniel Gutiérrez d.gutierrez.reina@gmail.com

CHULETAS PYTHONERAS

MATPLOTLIB	
Import matplotlib.pyplot as plt	**Importar** submódulo pyplot.
plt.figure(figsize= (10,10))	Crear **figura** y **dimensiones**.
plt.plot(x,y, color= 'r', marker = '*', linestyle = '- -')	**Y Vs X**, definimos el color, el marcador y el estilo de línea. Sólo son **obligatorios x e y.**
plt.xlabel("Valores X")	**Etiqueta eje X.**
plt.ylabel("Valores Y")	**Etiqueta eje Y.**
plt.xlim([0, 10])	**Límites eje X.**
plt.ylim([0, 20]))	**Límites eje Y**
plt.title(u"Mi gráfica")	**Título.**
plt.legend([serie1, serie2], loc = "upper center")	**Legenda, lista de legenda y posición.**
plt.grid(True)	**Activar rejilla.**
plt.save("mi_figura.jpg")	**Guardar figura** con **formato** en el **nombre.**
plt.hist(y)	**Histograma.**
plt.scatter(x,y)	**Dispersión.**
plt.bar(pos, valores)	**Diagrama de barras**
plt.xticks(pos, valores) plt.yticks(pos, valores)	Redefinir los **valores de los ejes.** Pos y valores son secuencias.

plt.subplot(221)
# definir figura 1 [0][0] 	
plt.subplot(222)
# definir figura 2 [0][1] 	
plt.subplot(223)
# definir figura 3 [1][0] 	
plt.subplot(224)
# definir figura 4 [1][1] 	
	Subplot (2x2)

ARGUMENTOS

Keyword argument	Description
color or c	Sets the color of the line; accepts any Matplotlib color format.
linestyle	Sets the line style; accepts the line styles seen previously.
linewidth	Sets the line width; accepts a float value in points.
marker	Sets the line marker style.
markeredgecolor	Sets the marker edge color; accepts any Matplotlib color format.
markeredgewidth	Sets the marker edge width; accepts float value in points.
markerfacecolor	Sets the marker face color; accepts any Matplotlib color format.
markersize	Sets the marker size in points; accepts float values.

COLORES

Color abbreviation	Color Name
b	blue
c	cyan
g	green
k	black
m	magenta
r	red
w	white
y	yellow

ESTILO DE LÍNEAS

Style abbreviation	Style
-	solid line
--	dashed line
-.	dash-dot line
:	dotted line

Realizado por Daniel Gutiérrez d.gutierrez.reina@gmail.com

CHULETAS PYTHONERAS

FILTRADO Y SLICING EN PANDAS	
S1[S1 > 10]	Elementos de la Serie S1 que sean **mayor que 10**.
S1[0]	**Primer elemento** de la Serie.
S1['a']	Elemento de la Serie que tiene un **índice 'a'**.
Sensores['S1']	Acceso a la **columna S1** del DataFrame sensores.
Sensores['S1'][2]	Tercer **elemento de la columna** S1.
Sensores['S1'][-1]	**Último elemento** de la columna S1.
Sensores ['S1'][1:3]	Elementos **[1, 3) de la columna S1**.
Sensores[Sensores['S1'] > 10]	Devuelve **DataFrame** incluyendo los **valores que cumplen la condición**.
Sensores['S1'][Sensores['S1'] > 10]	*Igual que la primera fila.
Sensores.filter(['S1', 'S2'])	Devuelve **DataFrame** incluyendo las **columnas seleccionadas**.
Sensores.iloc(0:1, 0:3)	Funciona como el **slicing de numpy** (filas, columnas). Intervalos ([0, 1), [0, 3))
Sensores.loc(index)	Seleccionar filas por **índice**.

Realizado por Daniel Gutiérrez d.gutierrez.reina@gmail.com

PANDAS	
import pandas as pd	**Importa** librería
S1= pd.Series(["Pacho", "Miguel", "Jorge"]) S1= pd.Series(np.arange(0,10))	**Crear** un objeto **Serie**.
S1.value	Devuelve los **valores** de la Serie como un array numpy.
S1.index	Devuelve los **índices** utilizados.
S1.append([u"Peña", "Navegante"])	**Concatenar** Series.
S2= S1.apply(lambda x: x.count('a'))	Devuelve una Serie resultado de **aplicar una función** sobre los **valores**.
Sensores = pd.read_csv("medidas.csv", sep =";", names = ["S1", "S2", "S3"])	**Crea un Dataframe** desde archivo csv *buscar el tipo de archivo que se quiere leer.
Sensores.shape	**Dimensiones** DataFrame.
Sensores.columns	**Etiquetas** de las columnas
Sensores.info()	**Información** sobre las **Series** que componen el DataFrame.
Sensores.heap(x)	Devuelve las **primeras x filas** del DataFrame
Sensores.tail(y)	Devuelve las **últimas y filas** del DataFrame.
Sensores.describe()	**Estadísticas**
Sensores["S4"] = Sensores["S1"] + Sensores["S2"]	Crea una **nueva columna** como resultado
Sensores.to_csv("resultados.csv")	**Escribe** el DataFrame en el archivo **csv**. *buscar el tipo de archivo que se quiere escribir.