

NumPy

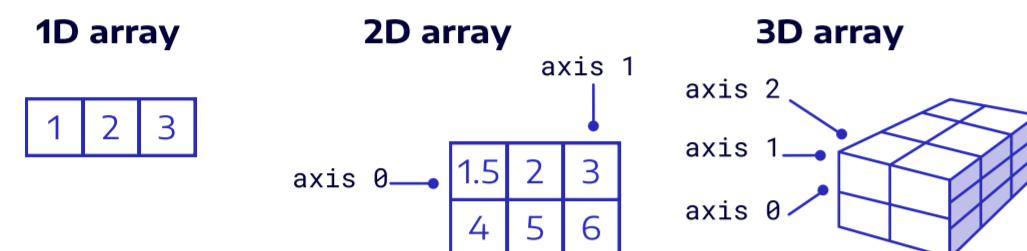


NumPy es la librería principal para cálculos científicos en Python. Proporciona potentes estructuras de datos, implementando arreglos (arrays) multidimensionales y herramientas para trabajar con estos.

Utiliza la siguiente convención de importación:

```
>>> import numpy as np
```

Arreglos NumPy



Creando Arreglos

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]),
      dtype = float)
```

Placeholders Iniciales

Crear un arreglo de ceros	<code>>>> np.zeros((3,4))</code>
Crear un arreglo de unos	<code>>>> np.ones((2,3,4), dtype=np.int16)</code>
Crear un arreglo de valores espaciados uniformemente (step value)	<code>>>> d = np.arange(10,25,5)</code>
Crear un arreglo de valores espaciados uniformemente (número de muestra)	<code>>>> np.linspace(0,2,9)</code>
Crear un arreglo constante	<code>>>> e = np.full((2,2),7)</code>
Crear una matriz de identidad 2x2	<code>>>> f = np.eye(2)</code>
Crear un arreglo con valores aleatorios	<code>>>> np.random.random((2,2))</code>
Crear un arreglo vacío	<code>>>> np.empty((3,2))</code>

Seleccionando, cortando e indexando

Seleccionar el elemento en el segundo index (tercera posición)	Subsetting <code>>>> a[2]</code> 3
Seleccionar elemento en fila 1 (segunda fila), columna 2 (tercera columna)	<code>>>> b[1,2]</code> 6.0
Seleccionar items de fila 0 y 1 (primera y segunda) y columna 1 (segunda columna)	Slicing <code>>>> a[0:2]</code> array([1, 2])
Seleccionar ítems en la fila 0 y 1 en columna 1	<code>>>> b[0:2,1]</code> array([2., 5.])
Seleccionar todos los ítems en la fila 0 (equivalente a <code>b[0:1,:]</code>)	<code>>>> b[:1]</code> array([[1.5, 2., 3.]])
Igual a <code>[1, :, :]</code>	<code>>>> c[1,...]</code> array([[3., 2., 1.], [4., 5., 6.]])
Invertir arreglo	<code>>>> a[: :-1]</code> array([3, 2, 1])
Seleccionar elementos de a menor a 2	Boolean Indexing <code>>>> a[a<2]</code> array([1])
Seleccionar elementos (1,0),(0,1),(1,2) y (0,0)	Fancy Indexing <code>>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]</code> array([4., 2., 6., 1.5])
Seleccionar un subconjunto de filas y columnas de la matriz	<code>>>> b[[1, 0, 1, 0]][:, [0, 1, 2, 0]]</code> array([[4., 5., 6., 4.], [1.5, 2., 3., 1.5], [4., 5., 6., 4.], [1.5, 2., 3., 1.5]])

NumPy Basics

Pedir Ayuda

```
>>> np.info(np.ndarray.dtype)
```

Arreglos (array) Matemáticos

Operaciones Aritméticas

Resta	<code>>>> g = a - b</code> array([-0.5, 0., 0., [-3., -3., -3.]])
Resta	<code>>>> np.subtract(a,b)</code>
Suma	<code>>>> b + a</code> array([2.5, 4., 6., [5., 7., 9.]])
Suma	<code>>>> np.add(b,a)</code>
División	<code>>>> a / b</code> array([[0.66666667, 1., 1.], [0.25, 0.4, 0.5]])
División	<code>>>> np.divide(a,b)</code>
Multiplicación	<code>>>> a * b</code> array([[1.5, 4., 9.], [4., 10., 18.]])
Multiplicación	<code>>>> np.multiply(a,b)</code>
Exponencial	<code>>>> np.exp(b)</code>
Raíz Cuadrada	<code>>>> np.sqrt(b)</code>
Seno de cada elemento del arreglo	<code>>>> np.sin(a)</code>
Coseno de cada elemento del arreglo	<code>>>> np.cos(b)</code>
Logaritmo natural de cada elemento del arreglo	<code>>>> np.log(a)</code>
Producto punto entre dos arreglos (e y f)	<code>>>> e.dot(f)</code> array([[7., 7.], [7., 7.]])

Comparación

Comparación de dos Arreglos	<code>>>> a == b</code> array([[False, True, True], [False, False, False]], dtype=bool)
Comparación de Elementos	<code>>>> a < 2</code> array([True, False, False], dtype=bool)
Comparación de Arreglos	<code>>>> np.array_equal(a, b)</code>

Funciones agregadas

Suma de elementos del arreglo	<code>>>> a.sum()</code>
Valor mínimo del arreglo	<code>>>> a.min()</code>
Máximo valor de fila de un arreglo	<code>>>> b.max(axis=0)</code>
Suma acumulada de los elementos	<code>>>> b.cumsum(axis=1)</code>
Media	<code>>>> a.mean()</code>
Mediana	<code>>>> b.median()</code>
Coeficiente de correlación	<code>>>> a.corrcoef()</code>
Desviación Standard	<code>>>> np.std(b)</code>

Copiando Arreglos

Crear una vista del arreglo con la misma data	<code>>>> h = a.view()</code>
Crear una copia del arreglo	<code>>>> np.copy(a)</code>
Crear una copia profunda del arreglo	<code>>>> h = a.copy()</code>

Ordenando Arreglos

Ordenar un arreglo	<code>>>> a.sort()</code>
Ordenar elementos según un eje del arreglo	<code>>>> c.sort(axis=0)</code>

Data Types

Signed 64-bit tipos enteros	<code>>>> np.int64</code>
Estandar de doble-precisión floating point	<code>>>> np.float32</code>
Números complejos representados por float128	<code>>>> np.complex</code>
Tipo Boolean que almacena valores TRUE y FALSE	<code>>>> np.bool</code>
Objeto de Python	<code>>>> np.object</code>
String de longitud fija	<code>>>> np.string_</code>
Unicode de longitud fija	<code>>>> np.unicode_</code>

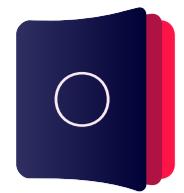
Inspeccionando Arreglos

Dimensiones del arreglo	<code>>>> a.shape</code>
Longitud del arreglo	<code>>>> len(a)</code>
Dimensiones del arreglo	<code>>>> b.ndim</code>
Número de elementos del arreglo	<code>>>> e.size</code>
Data type de los elementos del arreglo	<code>>>> b.dtype</code>
Nombre del tipo de dato	<code>>>> b.dtype.name</code>
Convertir un arreglo a un tipo diferente	<code>>>> b.astype(int)</code>

Manipulando Arreglos

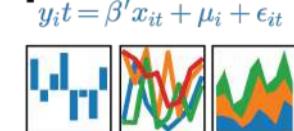
Permutar las dimensiones del arreglo	Transponer arreglos <code>>>> i = np.transpose(b)</code>
Transponer un arreglo	<code>>>> i.T</code>
Aplanar el arreglo	Cambiar tamaño de arreglos <code>>>> b.ravel()</code>
Remodelar, pero no cambiar la data	<code>>>> g.reshape(3,-2)</code>
Retornar un nuevo arreglo con forma (2,6)	Agregar/eliminar elementos <code>>>> h.resize((2,6))</code>
Añadir elementos a un arreglo	<code>>>> np.append(h,g)</code>
Insertar elementos a un arreglo	<code>>>> np.insert(a, 1, 5)</code>
Borrar elementos de un arreglo	<code>>>> np.delete(a,[1])</code>
Concatenar arreglos	Combinar arreglos <code>>>> np.concatenate((a,d),axis=0)</code> array([1, 2, 3, 10, 15, 20])
Apilar arreglos verticalmente (por filas)	<code>>>> np.vstack((a,b))</code> array([[1., 2., 3.], [1.5, 2., 3.], [4., 5., 6.]])
Apilar arreglos verticalmente (por filas)	<code>>>> np.r_[e,f]</code>
Apilar arreglos horizontalmente (por columnas)	<code>>>> np.hstack((e,f))</code> array([[7., 7., 1., 0.], [7., 7., 0., 1.]])
Crear arreglos apilados por columnas	<code>>>> np.column_stack((a,d))</code> array([[1, 10], [2, 15], [3, 20]])
Crear arreglos apilados por columnas	<code>>>> np.c_[a,d]</code>
Dividir el arreglo horizontalmente en el 3er índice	Dividir arreglos <code>>>> np.hsplit(a,3)</code> [array([1]),array([2]),array([3])]
Dividir el arreglo verticalmente en el 2do índice	<code>>>> np.vsplit(c,2)</code> [array([[1.5, 2., 1.], [4., 5., 6.]]), array([[3., 2., 3.], [4., 5., 6.]])]

Descubre más en www.option.cl



Pandas

pandas



Pandas es una librería escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

Utiliza la siguiente convención de importación:

```
>>> import pandas as pd
```

Estructura de Datos de Pandas

Series

a	3
b	-5
c	7
d	4

Un arreglo unidimensional que puede contener cualquier tipo de datos

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Una etiqueta bidimensional de estructura de datos con columnas de tipos potencialmente diferentes

	Country	Capital	Population
Index	Belgium	Brussels	11190846
	India	New Delhi	1303171035
	Brazil	Brasilia	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
   'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
   'Population': [11190846, 1303171035, 207847528]}
>>> df = pd.DataFrame(data,
   columns=['Country', 'Capital', 'Population'])
```

I/O

Leer y Escribir en CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Leer y Escribir en Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Leer múltiples hojas del mismo archivo
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Aplicar funciones

Aplicar la función	>>> f = lambda x: x*2
Aplicar la función a cada elemento	>>> df.apply(f)
	>>> df.applymap(f)

Python For Data Science Cheat Sheet

Pandas Basics

Pedir ayuda

```
>>> help(pd.Series.loc)
```

Seleccionar

Obtener un elemento

Obtener un elemento	>>> s['b'] -5
---------------------	------------------

Obtener conjunto de DataFrame

Obtener conjunto de DataFrame	>>> df[1:] Country Capital Population 1 India New Delhi 1303171035 2 Brazil Brasilia 207847528
-------------------------------	---

Seleccionar, Boolean, Indexar y Configurar

Seleccionar un solo valor por fila y columna

Por Posición	>>> df.iloc[[0],[0]] 'Belgium'
--------------	-----------------------------------

Seleccionar un solo valor por etiquetas de fila y columna

Por Etiqueta	>>> df.loc[[0], ['Country']] 'Belgium'
--------------	---

Seleccionar una sola columna de subconjunto de filas

Por Etiqueta/Posición	>>> df.ix[2] Country Brazil Capital Brasilia Population 207847528
-----------------------	--

Seleccionar una sola columna de subconjunto de columnas

Por Etiqueta/Posición	>>> df.ix[:, 'Capital'] 0 Brussels 1 New Delhi 2 Brasilia
-----------------------	--

Seleccionar por fila y columna

Por Etiqueta/Posición	>>> df.ix[1, 'Capital'] 'New Delhi'
-----------------------	--

Serie s donde el valor no es mayor a 1

Boolean Indexing	>>> s[~(s > 1)]
------------------	-----------------

S donde el valor es < -1 o > 2

Boolean Indexing	>>> s[(s < -1) (s > 2)]
------------------	---------------------------

Usar filtro para ajustar DataFrame

Boolean Indexing	>>> df[df['Population'] > 1200000000]
------------------	---------------------------------------

Configurar index a de la Series s a 6

Configuración	>>> s['a'] = 6
---------------	----------------

Ordenar y Rango

Ordenar por etiquetas a lo largo de un eje

Ordenar por los valores a lo largo de un eje

Asignar rangos a las entradas

```
>>> df.sort_index()
```

```
>>> df.sort_values(by='Country')
```

```
>>> df.rank()
```

Recuperar información de series/dataframe

Información Básica

(Filas, columnas)

```
>>> df.shape
```

Describir index

```
>>> df.index
```

Describir columnas de DataFrame

```
>>> df.columns
```

Información en DataFrame

```
>>> df.info()
```

Número de valores no-NA

```
>>> df.count()
```

Resumen

Suma de valores

```
>>> df.sum()
```

Suma acumulativa de valores

```
>>> df.cumsum()
```

Valores mínimos / máximos

```
>>> df.min()/df.max()
```

Valor de index mínimo / máximo

```
>>> df.idxmin()/df.idxmax()
```

Resumen estadístico

```
>>> df.describe()
```

Media de valores

```
>>> df.mean()
```

Mediana de valores

```
>>> df.median()
```

Alineación de Data

Alineación de data interna

Los valores NA se introducen en los índices que no se superponen

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b    NaN
c     5.0
d     7.0
```

También se puede hacer la alineación interna de los datos con la ayuda de los métodos de relleno:

Operaciones aritméticas con métodos de relleno

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

Descubre más en www.option.cl

Leer y Escribir en SQL Query o Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

read_sql() es un conveniente envoltorio de read_sql_table() y read_sql_query()

```
>>> pd.to_sql('myDF', engine)
```

Aplicar funciones

Aplicar la función	>>> f = lambda x: x*2
Aplicar la función a cada elemento	>>> df.apply(f)
	>>> df.applymap(f)

Eliminar

Eliminar valores de las filas (ejes=0)	>>> s.drop(['a', 'c'])
Eliminar valores de la columna(eje=1)	>>> df.drop('Country', axis=1)

Python Basics

Variables y Strings

Las variables son usadas para guardar valores. Un string es una serie de caracteres conformado por comillas simples (' ') o dobles ("").

Hello world	<pre>>>> print("Hello world!")</pre>
Hello world con una variable	<pre>>>> msg= "Hello world!" >>> print(msg)</pre>
Concatenación (unión de strings)	<pre>>>> first_name = 'albert' >>> last_name = 'einstein' >>> full_name = first_name + ' ' + last_name >>> print(full_name)</pre>

Pidiendo Ayuda

```
>>> help(str)
```

Variables y Tipos de Data

Asignación Variable

```
>>> x=5  
>>> x  
5
```

Cálculos con Variables

Suma de variables	<pre>>>> x+2 7</pre>
Resta de variables	<pre>>>> x-2 3</pre>
Multiplicación de variables	<pre>>>> x*2 10</pre>
Exponencial de una variable	<pre>>>> x**2 25</pre>
Resto de una variable	<pre>>>> x%2 1</pre>
División de una variable	<pre>>>> x/float(2) 2.5</pre>

Tipos de variables y conversión

Variables a strings	<code>str ()</code>	'5', '3.45', 'True'
Variables a enteros	<code>int ()</code>	5, 3, 1
Variables a floats	<code>float ()</code>	5.0, 1.0
Variable a booleanos	<code>bool ()</code>	True, True, True

Strings

```
>>> my_string = 'thisStringIsAwesome'  
>>> my_string  
'thisStringIsAwesome'
```

Operaciones de String

```
>>> my_string * 2  
'thisStringIsAwesomethisStringIsAwesome'  
>>> my_string + 'Innit'  
'thisStringIsAwesomeInnit'  
>>> 'm' in my_string  
True
```

Listas

```
>>> a = 'is'  
>>> b = 'nice'  
>>> my_list = ['my', 'list', a, b]  
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Crear Lista

Recorrer una lista

Agregar ítems a la lista

Hacer listas numéricas

Seleccionando elementos de la Lista

Seleccionar item en index 1

Seleccionar 3º último item

Seleccionar items de index 1 y 2

Seleccionar items después de index 0

Seleccionar items antes del index 3

Copiar my_list

my_list[list][itemOfList]

Operaciones de Lista

```
>>> my_list + my_list  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list * 2  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list2 > 4  
True
```

Comparación

Obtener el index de un ítem

Contar un ítem

Añadir un ítem a la vez

Quitar un ítem

Invertir la lista

Añadir un ítem

Quitar un ítem

Insertar un ítem en cierta posición

Ordenar la lista

Operaciones de String

```
>>> my_string[3]  
>>> my_string[4:9]
```

Métodos de String

String a mayúscula

String a minúscula

Contar los elementos del string

Cambiar los elementos del string

Eliminar espacios en blanco

```
>>> my_string.upper()
```

```
>>> my_string.lower()
```

```
>>> my_string.count('w')
```

```
>>> my_string.replace('e', 'i')
```

```
>>> my_string.strip()
```

Diccionarios

Los diccionarios almacenan conexiones entre piezas de información. Cada elemento es un par llave - valor

Un simple Diccionario

Accediendo a un valor

Recorriendo todos los pares llave-valor

Recorriendo todos los pares llave

Recorriendo todos los valores

```
>>> alien = {'color': 'green',  
'points': 5}
```

```
>>> alien['x_position'] = 0
```

```
>>> fav_numbers = {'eric': 17, 'ever':  
4}  
>>> for name, number in  
>>> fav_numbers.items():  
>>> print(name + ' loves a  
number')  
>>> fav_numbers = {'eric': 17, 'ever':  
4}
```

```
>>> for number in  
>>> fav_numbers.values():  
>>> print(str(number) + ' is a  
favorite')
```

If Statements

Los If statements son usados para comprobar condiciones particulares y que respondan apropiadamente.

Pruebas Condicionales

Igual a

```
>>> x == 42
```

Mayor a

```
>>> x > 42
```

Mayor o igual a

```
>>> x >= 42
```

```
>>> 'trek' in bikes  
>>> 'surly' not in bikes
```

Pruebas Condicionales con listas

Asignando valores booleanos

```
>>> game_active = True  
>>> can_edit = False
```

```
>>> if age >= 4:  
ticket_price = 0
```

```
>>> elif age < 18:  
ticket_price = 10
```

```
>>> else:  
ticket_price = 15
```

Librerías

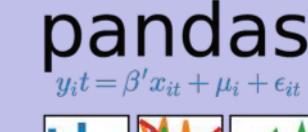
Importar librerías

```
>>> import numpy
```

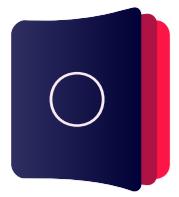
```
>>> import numpy as np
```

Importación Selectiva

```
>>> from math import pi
```



Descubre más en www.option.cl



Matplotlib



Es una librería de Python 2D que produce gráficas de calidad en una variedad de formatos impresos y entornos interactivos a través de plataformas.

1) Preparar la Data

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> X, Y = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2) Crear Plot

```
>>> import matplotlib.pyplot as plt
```

Figura

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Ejes

Todos los plots están hechos respecto a Axes (ejes). En la mayoría de los casos un subplot se ajustará a tus necesidades. Un subplot son ejes en un sistema de cuadrilla.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3) Rutinas de Plotting

1D Data

Dibujar puntos con líneas o marcadores conectándolos	<pre>>>> fig, ax = plt.subplots() >>> lines = ax.plot(x,y)</pre>
Dibujar puntos sin conectar, a escala o con colores	<pre>>>> ax.scatter(x,y)</pre>
Plot rectángulos verticales (ancho constante)	<pre>>>> axes[0,0].bar([1,2,3],[3,4,5])</pre>
Plot rectángulos horizontales (altura constante)	<pre>>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])</pre>
Dibujar línea horizontal a través de los ejes	<pre>>>> axes[1,1].axhline(0.45)</pre>
Dibujar línea vertical a través de los ejes	<pre>>>> axes[0,1].axvline(0.65)</pre>
Dibujar polígonos llenos	<pre>>>> ax.fill(x,y,color='blue')</pre>
Rellenar entre los valores de y y el eje x	<pre>>>> ax.fill_between(x,y,color='yellow')</pre>

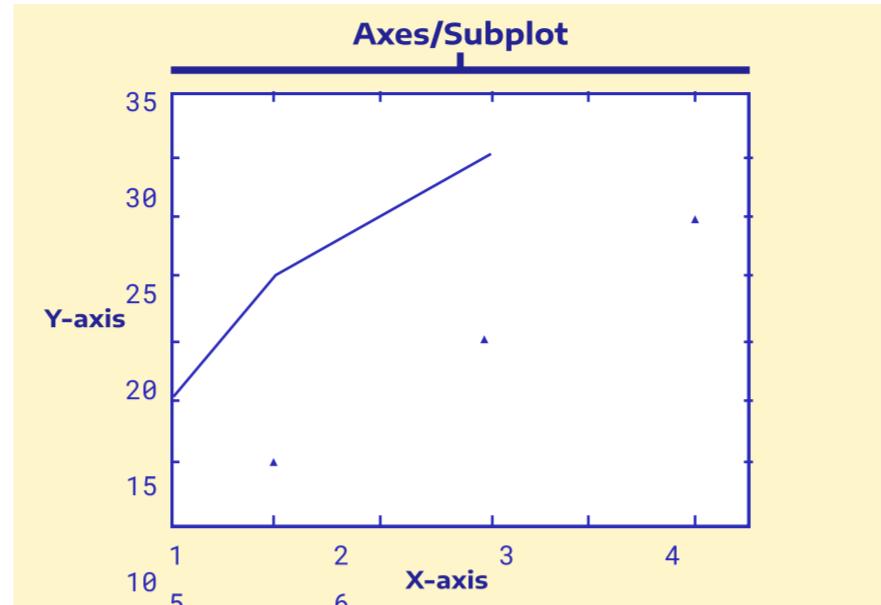
2D Data o Imágenes

Mapa de colores o arreglos RGB	<pre>>>> fig, ax = plt.subplots() >>> im = ax.imshow(img, >>> cmap='gist_earth', >>> interpolation='nearest', >>> vmin=-2, >>> vmax=2)</pre>
--------------------------------	--

Matplotlib

Anatomía y Flujo de Trabajo

Anatomía del Plot



4) Personalizar Plot

Colores, Barras de colores y Mapa de Colores

```
>>> plt.plot(x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
>>> cmap='seismic')
```

Estilo de Línea

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Texto Matemático

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

Texto y Anotaciones

```
>>> ax.text(1, -2.1, 'Example Graph', style='italic')
>>> ax.annotate("Sine", xy=(8, 0), xycoords='data', xytext=(10.5, 0), textcoords='data', arrowprops=dict(arrowstyle="->", connectionstyle="arc3"),)
```

Campos vectoriales

Agregar una flecha a los ejes	<pre>>>> axes[0,1].arrow(0,0,0.5,0.5)</pre>
Graficar un campo 2D de flechas	<pre>>>> axes[1,1].quiver(y,z)</pre>
Graficar un campo 2D de flechas	<pre>>>> axes[0,1].streamplot(X,Y,U,V)</pre>

Distribuciones de Datos

Graficar un histograma	<pre>>>> ax1.hist(y)</pre>
Graficar un boxplot o diagrama de caja y bigote	<pre>>>> ax3.boxplot(y)</pre>
Graficar un violin plot	<pre>>>> ax3.violinplot(z)</pre>

3D Data o Imágenes

Graficar pseudocolor plot de 2D array	<pre>>>> axes2[0].pcolor(data2)</pre>
Graficar Pseudocolor de 2D array	<pre>>>> axes2[0].pcolormesh(data)</pre>
Graficar una línea de contorno	<pre>>>> CS = plt.contour(Y,X,U)</pre>
Graficar una línea de contorno rellena	<pre>>>> axes2[2].contourf(data1)</pre>
Etiquetar una línea de contorno	<pre>>>> axes2[2]= ax.clabel(CS)</pre>

Anatomía del Plot

1 Preparar Data	2 Crear Plot	3 Plot	4 Personalizar Plot	5 Salvar Plot	6 Mostrar Plot
Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
<pre>>>> import matplotlib.pyplot as plt</pre>	<pre>>>> x = [1,2,3,4]</pre>	<pre>>>> y = [10,20,25,30]</pre>	<pre>>>> fig = plt.figure()</pre>	<pre>>>> ax = fig.add_subplot(111)</pre>	<pre>>>> ax.plot(x, y, color='lightblue', linewidth=3)</pre>
				<pre>>>> ax.scatter([2,4,6], [5,15,25], color='darkgreen', marker='^')</pre>	<pre>>>> ax.set_xlim(1, 6.5)</pre>
				<pre>>>> plt.savefig('foo.png')</pre>	<pre>>>> plt.show()</pre>

Límites, leyendas y layout

Agregar padding a plot	Límites y Autoescalamiento
Establecer la relación de aspecto de la trama en 1	<pre>>>> ax.margins(x=0.0,y=0.1)</pre>
Establecer límites para los ejes x e y	<pre>>>> ax.axis('equal')</pre>
Establecer límites para eje x	<pre>>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])</pre>
Establecer un título y etiquetas para ejes x e y	Leyendas
Establecer leyendas para ejes x e y	<pre>>>> ax.set(title='An Example Axes', ylabel='Y-Axis', xlabel='X-Axis')</pre>
Establecer manualmente x-ticks	Ticks
Alargar y-ticks, que estén dentro y fuera del eje	<pre>>>> ax.xaxis.set(ticks=range(1,5), ticklabels=[3,100,-12,"foo"])</pre>
Ajustar el espacio entre subplots	Espaciado de Subplots
Encajar subplot(s) en el área de la figura	<pre>>>> fig.subplots_adjust(wspace=0.5, hspace=0.3, left=0.125, right=0.9, top=0.9, bottom=0.1)</pre>
Hacer la línea del eje superior invisible para un plot	Axis Spines
Mover la línea del eje inferior hacia afuera	<pre>>>> ax1.spines['top'].set_visible(False)</pre>
Marcadores	<pre>>>> ax1.spines['bottom'].set_position(('outward',10))</pre>
	<pre>>>> fig, ax = plt.subplots() >>> ax.scatter(x,y,marker=".") >>> ax.plot(x,y,marker="o")</pre>

5) Guardar Plot

Guardar figuras	Guardar figuras
	<pre>>>> plt.savefig('foo.png')</pre>
Guardar figuras sin fondo	Guardar figuras sin fondo
	<pre>>>> plt.savefig('foo.png', transparent=True)</pre>

6) Mostrar Plot

```
>>> plt.show()
```

Cerrar y Limpiar

Limpiar un eje	<pre>>>> plt.cla() Clear an axis</pre>
Limpiar la figura completa	<pre>>>> plt.clf() Clear the entire figure</pre>
Cerrar ventana	<pre>>>> plt.close()</pre>

Descubre más en www.option.cl



Visualización de datos estadísticos con Seaborn

Seaborn es una librería para Python que está basada en matplotlib y proporciona una interfaz de alto nivel para dibujar atractivos gráficos estadísticos.

Utiliza la siguiente convención de importación:

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

Los pasos básicos para crear gráficos con Seaborn son:

1. Preparar la data
2. Controlar la estética de las figuras
3. Plot con Seaborn
4. Personalizar tu plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips")
>>> sns.set_style("whitegrid")           ← Paso 1
>>> g = sns.lmplot(x="tip",
                   y="total_bill",
                   data=tips,
                   aspect=2)                         ← Paso 2
>>> g.set_axis_labels("Tip", "Total bill(USD)").
set(xlim=(0,10), ylim=(0,100))          ← Paso 3
>>> plt.title("title")                  ← Paso 4
>>> plt.show(g)
```

1) Preparar la data

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({'x':np.arange(1,101),
                       'y':np.random.normal(0,4,100)})
```

Seaborn también ofrece sets de datos incorporados:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

2) Estética de la Figura

Crear una figura y un subplot

```
>>> f, ax = plt.subplots(figsize=(5,6))
```

Estilos de Seaborn

(Re) iniciar seaborn por defecto

```
>>> sns.set()
>>> sns.set_style("whitegrid")
```

Establecer los parámetros de matplotlib

```
>>> sns.set_style("ticks", Set the
                    matplotlib parameters
                    {"xtick.major.size":8,
                     "ytick.major.size":8})
>>> sns.axes_style("whitegrid")
```

Establecer los parámetros de matplotlib

Devuelve un dictado de parámetros o utiliza with para establecer temporalmente el estilo

Funciones de Contexto

Establecer contexto "talk"

```
>>> sns.set_context("talk")
```

Establecer contexto para "notebook", escalar elementos fuente y sobre-escribir parámetros de mapeo

```
>>> sns.set_context("notebook"
                    font_scale=1.5,
                    rc={"lines.linewidth":2.5})
```

Paletas de colores

Definir paleta de colores

```
>>> sns.set_palette("husl",3)
>>> sns.color_palette("husl")
>>> flatui =
["#9b59b6", "#3498db", "#95a5a6", "#e74"]
```

Usar with para establecer temporalmente la paleta

```
>>> sns.set_palette(flatui)
```

Establecer tu propia paleta de colores

Seaborn

3) Plotting con Seaborn

Cuadrillas de Ejes

Cuadrilla de Subplot para trazar relaciones condicionales

```
>>> g = sns.FacetGrid(titanic,
                      col="survived",
                      row="sex")
>>> g.map(plt.hist, "age")
```

Dibujar un plot categórico en un Facetgrid

```
>>> sns.factorplot(x="pclass",
                     y="survived",
                     hue="sex",
                     data=titanic)
```

Graficar data y ajuste de modelo de regresión usando un FacetGrid

```
>>> sns.lmplot(x="sepal_width",
                y="sepal_length",
                hue="species",
                data=iris)
```

Grilla de subplot para graficar relaciones por pares

```
>>> h = sns.PairGrid(iris)
```

Plot para pares de distribuciones bivariadas

```
>>> sns.pairplot(iris)
```

Grilla para gráfico bivariado con distribuciones marginales univariadas

```
>>> i = sns.JointGrid(x="x", y="y",
                      data=data)
```

Gráfico de distribución bivariada

```
>>> sns.jointplot("sepal_length",
                  "sepal_width",
                  data=iris,
                  kind='kde')
```

Plots de Regresión

Datos de plot y ajuste de modelo de regresión lineal

```
>>> sns.regplot(x="sepal_width",
                y="sepal_length",
                data=iris,
                ax=ax)
```

4) Personalizaciones Adicionales

Objetos de Axisgrid

Quitar columna izquierda

```
>>> g.despine(left=True)
```

Establecer las etiquetas del eje y

```
>>> g.set_ylabels("Survived")
```

Establecer las etiquetas de tick para x

```
>>> g.set_xticklabels(rotation=45)
```

Establecer las etiquetas de eje

```
>>> g.set_axis_labels("Survived",
                      "Sex")
```

Establecer el límite y ticks de los ejes x e y

```
>>> h.set_xlim=(0, 5),
        ylim=(0, 5),
        xticks=[0, 2.5, 5],
        yticks=[0, 2.5, 5])
```

5) Mostrar o Guardar Plot

Mostrar el plot

```
>>> plt.show()
```

Guarda el plot como una figura

```
>>> plt.savefig("foo.png")
```

Guarda una figura transparente

```
>>> plt.savefig("foo.png"
                 transparent=True)
```

Plot Categórico

Scatterplot con una variable categórica

```
>>> sns.stripplot(x="species",
                  y="petal_length",
                  data=iris)
```

Scatterplot categórico con puntos no superpuestos

```
>>> sns.swarmplot(x="species",
                  y="petal_length",
                  data=iris)
```

Mostrar estimaciones de puntos e intervalos de confianza con glifos de dispersión

```
>>> Bar Chart
>>> sns.barplot(x="sex",
                  y="survived",
                  hue="class",
                  data=titanic)
```

Mostrar recuento de observaciones

```
>>> sns.countplot(x="deck",
                  data=titanic, palette="Greens_d")
```

Mostrar estimaciones de puntos e intervalos de confianza como barras rectangulares

```
>>> Boxplot
>>> sns.boxplot(x="alive",
                  y="age",
                  hue="adult_male",
                  data=titanic)
```

Diagrama de caja con datos de formato ancho

```
>>> sns.boxplot(data=iris, orient="h")
```

Violin Plot

```
>>> sns.violinplot(x="age",
                  y="sex",
                  hue="survived",
                  data=titanic)
```

Plots de Distribución

Gráfico de distribución univariada

```
>>> plot = sns.distplot(data.y,
                        kde=False,
                        color="b")
```

Plot

Añadir título al plot

```
>>> plt.title("A Title")
```

Ajustar la etiqueta del eje y

```
>>> plt.ylabel("Survived")
```

Ajustar la etiqueta del eje x

```
>>> plt.xlabel("Sex")
```

Ajustar los límites del eje y

```
>>> plt.ylim(0,100)
```

Ajustar los límites del eje x

```
>>> plt.xlim(0,10)
```

Ajustar una propiedad del plot

```
>>> plt.setp(ax, yticks=[0,5])
```

Ajustar parámetros de subplot

```
>>> plt.tight_layout()
```

Cerrar y Limpiar

Limpiar un eje

```
>>> plt.cla() Clear an axis
```

Limpiar la figura completa

```
>>> plt.clf() Clear the entire figure
```

Cerrar ventana

```
>>> plt.close()
```

Descubre más en www.option.cl