Programación

Tema 8. Librerías

Javier González Villa

David Lázaro Urrutia

Departamento de Matemática Aplicada
y Ciencias de la Computación

Este material se publica bajo la siguiente licencia:
Creative Commons BY-NC-SA 4.0

Contenidos

1. Librerías
    1. Biblioteca Estándar
    2. Matemáticas - NumPy
    3. Visualización - Matplotlib
    4. Analítica de datos - Pandas

1. Librerías

- Son conjuntos de módulos que implementan funciones útiles para el usuario incrementando el nivel de abstracción y reutilización de código.
- El conjunto de librerías básico incluido en la distribución por defecto de Python se denomina Biblioteca Estándar.
- Las librerías específicas requieren de instalación individual o a través de paquetes de librerías que contienen múltiples a la vez.
- Algunas librerías pueden tener dependencias de otras si utilizan sus funcionalidades.
- El instalador por defecto de Python es Pip que es utilizado para instalar paquetes que se encuentren en el directorio Python Package Index.

https://pypi.org/

1.1. Biblioteca Estándar

random	math	cmath	statistics
Generación de números pseudoaleatorios	Funciones matemáticas	Funciones matemáticas con números complejos	Funciones estadísticas

```
[103]: import random
       random.random()
[103]: 0.8038393649984922

[109]: random.choices([1,2,3],weights=[0.3,0.3,0.4],k=2)
[109]: [3, 2]

[111]: random.normalvariate(0,1)
[111]: 0.8506009810536409

[113]: random.randint(10,20)
[113]: 11
```

```
[114]: import math
       math.factorial(5)
[114]: 120

[116]: math.cos(math.pi)
[116]: -1.0

[117]: math.floor(4.3)
[117]: 4
```

```
[119]: import statistics
       statistics.mean([1,2,3,4])
[119]: 2.5

[120]: statistics.stdev([1,2,3,4])
[120]: 1.2909944487358056

[121]: statistics.quantiles([1,2,3,4])
[121]: [1.25, 2.5, 3.75]
```

1.1. Biblioteca Estándar

os	sys	time
Acceso a funciones del sistema operativo	Parámetros y funciones específicas del sistema	Acceso a tiempo y conversiones

```
[122]: import os
       os.getcwd()

[122]: '/home/pyodide'

[124]: os.getlogin()

[124]: 'web_user'

[125]: os.uname()

[125]: posix.uname_result(sysname='Emscripten',
```

```
[127]: import sys
       sys.flags

[127]: sys.flags(debug=0, inspect=0, interactive=0, optimize
       iet=0, hash_randomization=1, isolated=0, dev_mode=Fal

[128]: sys.implementation

[128]: namespace(name='cpython',
                 cache_tag='cpython-310',
                 version=sys.version_info(major=3, minor=10,
                 hexversion=50987760,
                 _multiarch='wasm32-emscripten')

[129]: sys.maxsize

[129]: 2147483647
```

1.1. Biblioteca Estándar

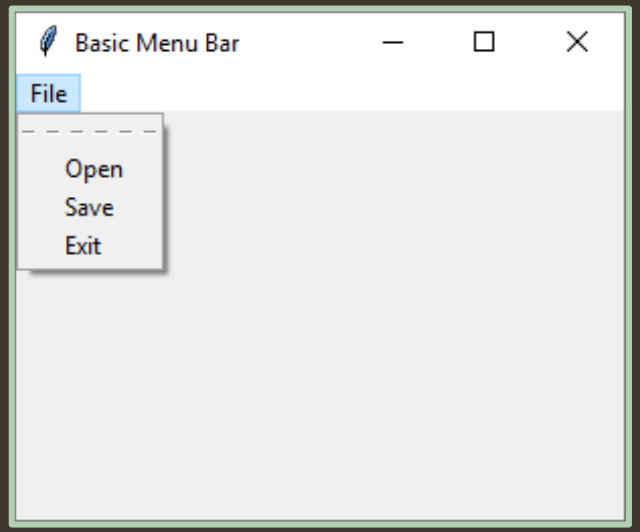
csv	json	xml
Manejo de ficheros .csv	Codificador y decodificador JSON	Manejo de ficheros .xml

- Todas estas librerías permiten manejar diferentes tipos de ficheros, tanto en lectura, escritura o interpretación de ficheros para ser almacenados en objetos de manera automática.

1.1. Biblioteca Estándar

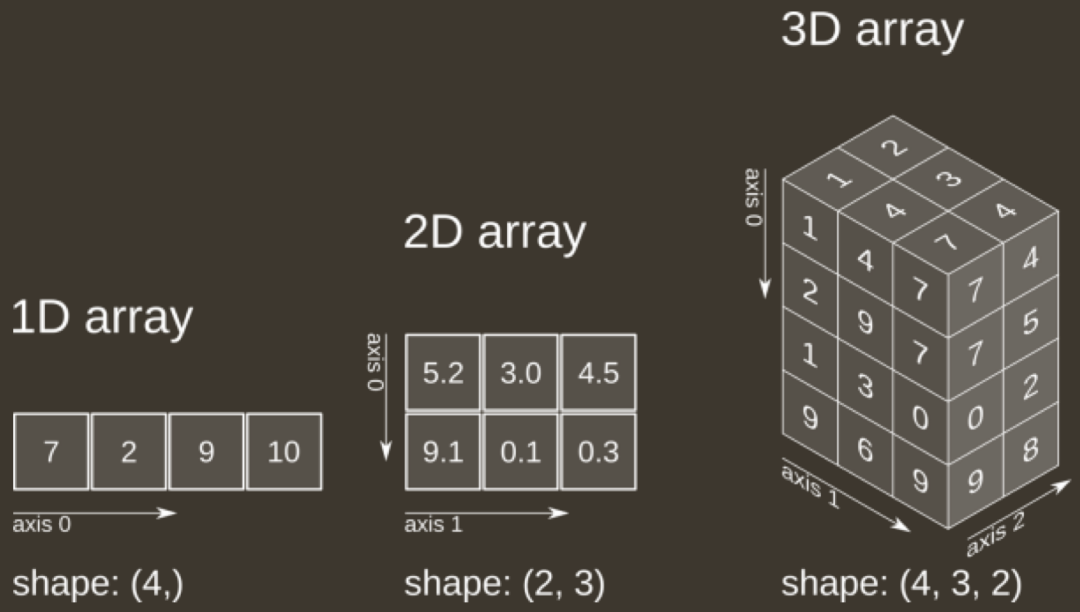
threading	tkinter	itertools
Paralelismo basado en hilos	Creación de interfaces	Creación de bucles eficientes

```
[145]: import itertools
       for v in itertools.repeat(10,3):
           print(v)

       10
       10
       10

[146]: for v in itertools.combinations([1,2,3],2):
           print(v)

       (1, 2)
       (1, 3)
       (2, 3)
```



1.2. Matemáticas - NumPy

- Utilización: import numpy as np
- Cálculo numérico y análisis de gran volumen de datos.
- Define una clase de objeto especial denominado Array para representación de datos en múltiples dimensiones.
- Velocidad de procesamiento hasta 50 veces superior que las listas.



1.2. Matemáticas - NumPy

- Inicialización básica: (mismo tipo de objetos)

  nombre_array = np.array([[1,2,…,n],[…],…,[…]])

- Otras inicializaciones:
  - np.empty(dimensión): Array vacío
  - np.zeros(dimensión): Array de ceros
  - np.full(dimensión, valor): Array de valores específicos
  - np.identity(dimensión): Array matriz identidad
  - np.random.random(dimensión): Array de valores aleatorios

```
[1]: import numpy as np

[5]: np.array([[1,2,3],[4,5,6],[7,8,9]])

[5]: array([[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]])

[12]: np.identity(3)

[12]: array([[1., 0., 0.],
             [0., 1., 0.],
             [0., 0., 1.]])

[10]: np.random.random((3,2))

[10]: array([[0.74493089, 0.17107606],
             [0.33718138, 0.78568468],
             [0.01714379, 0.94000157]])
```

1.2. Matemáticas - NumPy

- Atributos de un Array:

  array.ndim / .shape / .size / .dtype

- Consultar elementos:

  array[índice_dim_1][índice_dim_2]…[índice_dim_n]

```
[14]: array = np.array([[1,2,3],[4,5,6]])
      array

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

[17]: array.shape

[17]: (2, 3)

[19]: array[0][1]

[19]: 2
```

1.2. Matemáticas - NumPy

- Funciones específicas:
  - Operadores matemáticos: +, -, *, /, **
  - Álgebra matricial:
    - Producto escalar: array1.dot(array2)
    - Módulo: np.linalg.norm(array)
    - Producto de matrices: array1.dot(array2)
    - Matriz transpuesta: array.T
    - Determinante: np.linalg.det(array)
    - Inversa: np.linalg.inv(array)
    - Autovalores: np.linalg.eigvals(array)
    - Solución de sistemas: np.linalg.solve(array_a, array_b)
    - ...

```
[20]: import numpy as np
      array1 = np.array([1,2,3])
      array2 = np.array([2,3,4])

[21]: array1.dot(array2)

[21]: 20

[26]: a = np.array([[1,-2],[2,-1]])
      b = np.array([-4,1])

[27]: np.linalg.solve(a,b)

[27]: array([2., 3.])
```

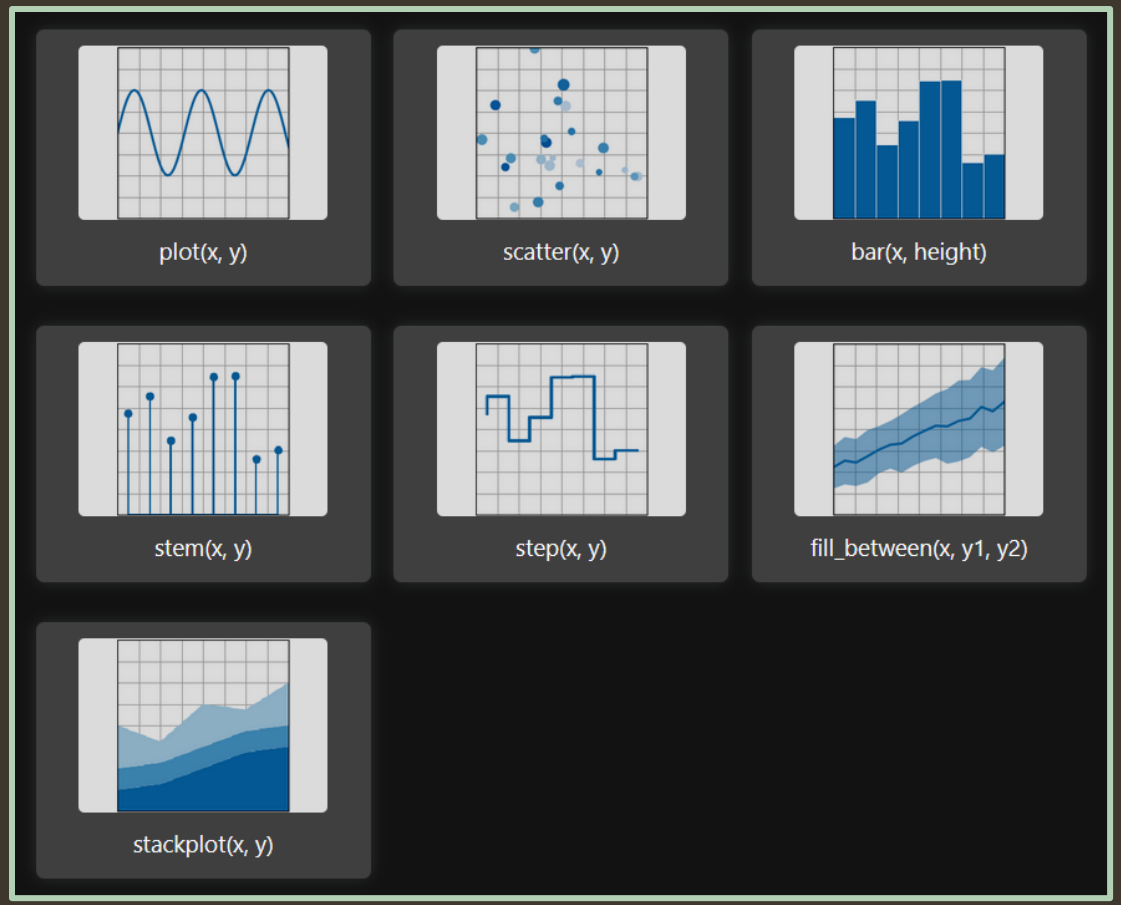
1.3. Visualización - Matplotlib

- Utilización: import matplotlib.pyplot as plt
- Es una librería especializada en la creación de gráficos en dos dimensiones.
- Permite crear los gráficos más comúnmente utilizados:
  - Histogramas.
  - Diagramas de barras.
  - Nubes de puntos.
  - Diagrama de caja.
  - …

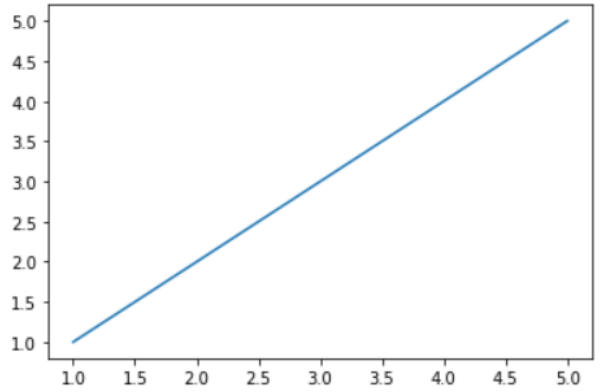
https://matplotlib.org/

1.3. Visualización - Matplotlib

Básicas



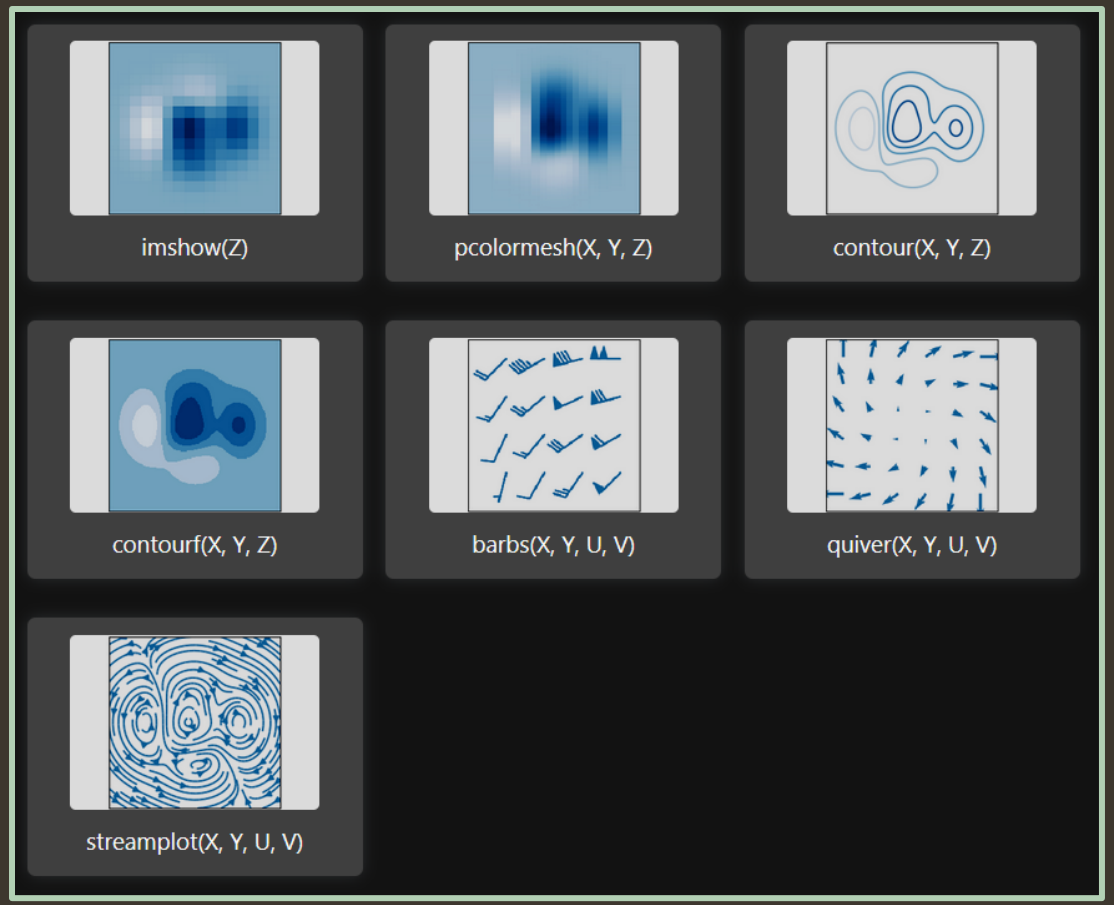

```
[230]: import matplotlib.pyplot as plt

[231]: x=[1,2,3,4,5]
       y=[1,2,3,4,5]
       plt.plot(x,y)
       plt.show()
```



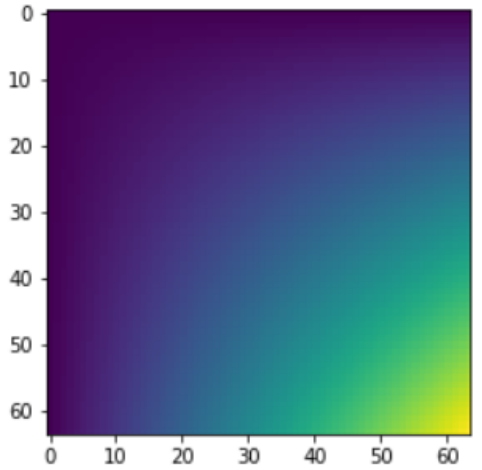

```
[232]: help(plt.plot)

Help on function plot in module matplotlib.pyplot:

plot(*args, scalex=True, scaley=True, data=None, **kwargs)
    Plot y versus x as lines and/or markers.

    Call signatures::

        plot([x], y, [fmt], *, data=None, **kwargs)
        plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

    The coordinates of the points or line nodes are given by *x*, *y*.
```

1.3. Visualización - Matplotlib

Matrices y campos

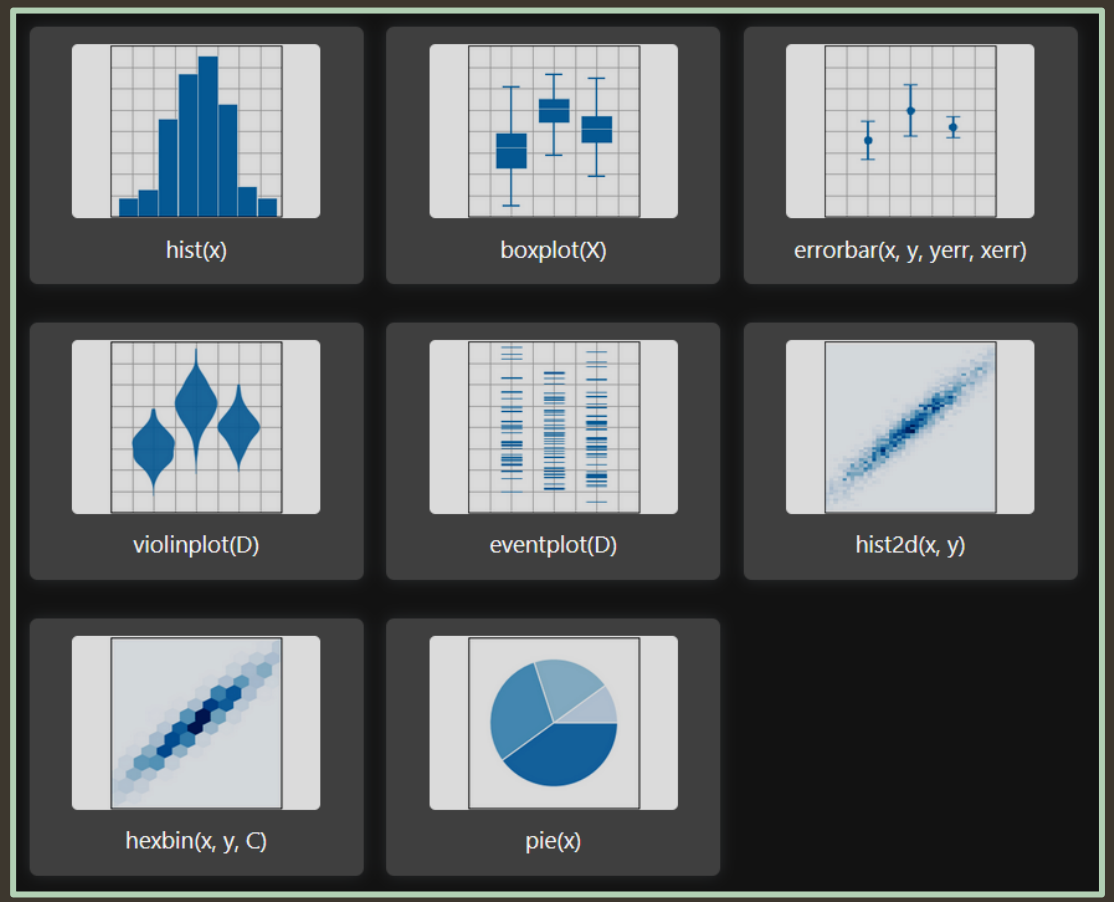



```
[265]: import matplotlib.pyplot as plt
       import numpy as np

[266]: X, Y = np.meshgrid(np.linspace(0, 16, 64), np.linspace(0, 16, 64))
       Z = X*Y

[267]: plt.imshow(Z, vmin=0, vmax=256)

[267]: <matplotlib.image.AxesImage at 0x1da84d96bb0>
```

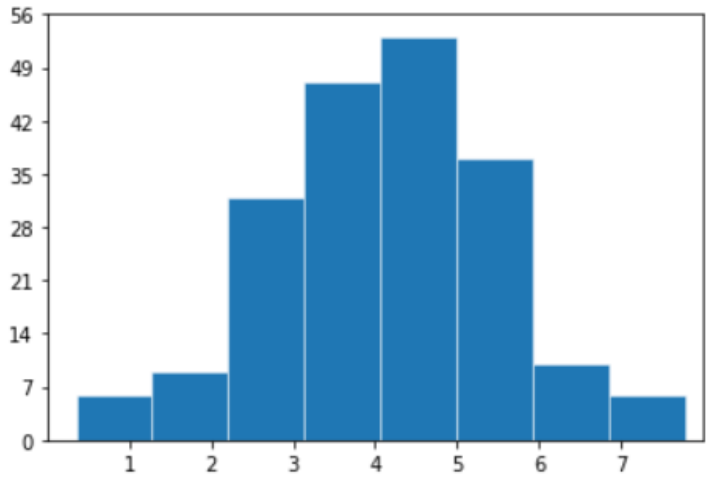


1.3. Visualización - Matplotlib

Estadísticas

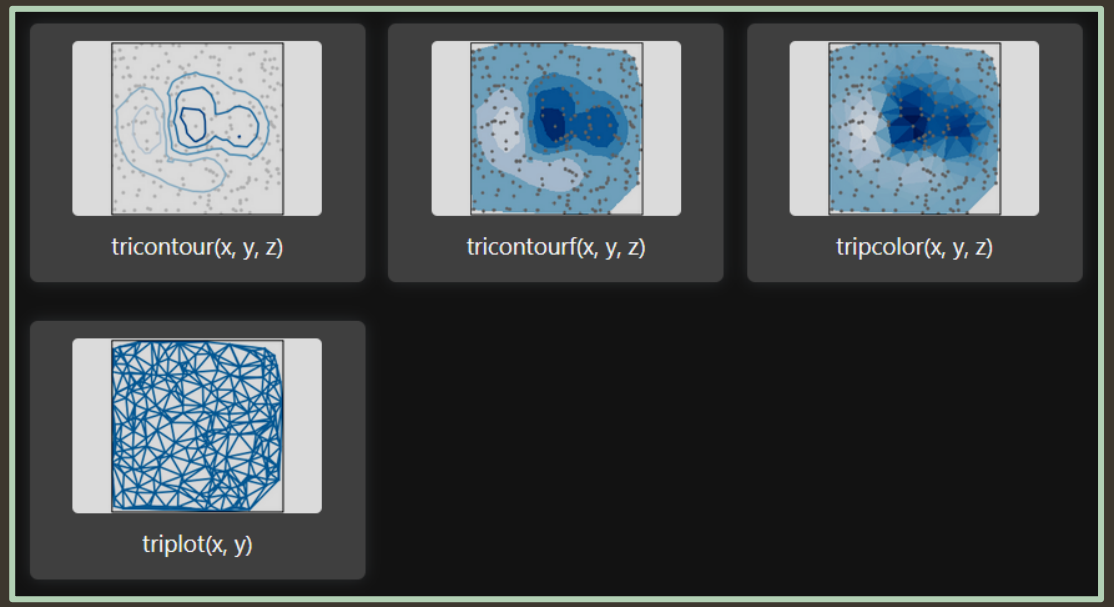



```
[275]: import matplotlib.pyplot as plt
       import numpy as np

[276]: np.random.seed(1)
       x = 4 + np.random.normal(0, 1.5, 200)

[277]: fig, ax = plt.subplots()
       ax.hist(x, bins=8, linewidth=0.5, edgecolor="white")
       ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 56), yticks=np.linspace(0, 56, 9))
       plt.show()
```

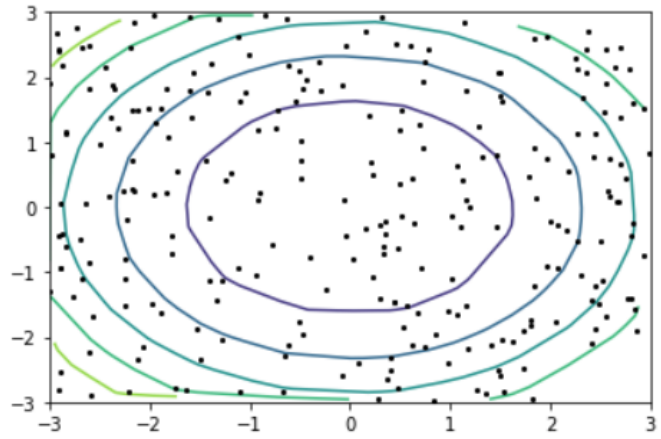


1.3. Visualización - Matplotlib

Datos no estructurados




```
[326]: import matplotlib.pyplot as plt
       import numpy as np

[327]: np.random.seed(1)
       x = np.random.uniform(-3, 3, 256)
       y = np.random.uniform(-3, 3, 256)
       z = x**2+y**2
       levels = np.linspace(z.min(), z.max(), 7)

[328]: fig, ax = plt.subplots()
       ax.plot(x, y, 'o', markersize=2, color='black')
       ax.tricontour(x, y, z, levels=levels)
       ax.set(xlim=(-3, 3), ylim=(-3, 3))
       plt.show()
```



1.4. Analítica de datos - Pandas

- Utilización: import pandas as pd
- Define nuevas estructuras y funcionalidades basándose en los Array de NumPy.
- Permite lectura y escritura sencilla de CSV, Excel, y bases de datos SQL.
- Implementa funcionalidades para trabajo con series temporales.
- Muy eficiente en el trabajo con grandes volúmenes de datos.

Series	**DataFrame**	~~**Panel**~~
Estructura de una dimensión.	Estructura en forma de tabla.	~~Estructura en forma de matriz tridimensional.~~

1.4. Analítica de datos - Pandas

- Series:
  - Se pueden crear a través de lista de valores o de diccionarios.
  - La consulta es como en listas a través del índice o por la etiqueta del diccionario asignada.
  - Funciones importantes:
    - serie.count(): Número de elementos.
    - serie.cumsum(): Suma acumulada de la serie.
    - serie.value_counts(): Frecuencia de valores.
    - serie.min() / .max(): Valores mínimo o máximo.
    - serie.mean(): Media de la serie.
    - serie.describe(): Descripción estadística de la serie.

```
[30]: import pandas as pd

[35]: s = pd.Series(['A','B','C'],dtype='string')
      s

[35]: 0    A
      1    B
      2    C
      dtype: string

[37]: s = pd.Series({'A':1.0,'B':2.0,'C':3.0})
      s

[37]: A    1.0
      B    2.0
      C    3.0
      dtype: float64

[38]: s[1:2]

[38]: B    2.0
      dtype: float64

[39]: s['C']

[39]: 3.0
```

1.4. Analítica de datos - Pandas

- Series:
  - Se les puede aplicar operaciones aritméticas básicas de Python.

+ - / % * //

  - Se les puede aplicar operaciones más complejas recogidas en la librería math.

serie.apply(math.función)

```
[11]: import pandas as pd
      import math

[9]:  s1 = pd.Series([1,2,3])
      s2 = pd.Series([1,2,3])

[10]: s1+s2

[10]: 0    2
      1    4
      2    6
      dtype: int64

[12]: s1.apply(math.cos)

[12]: 0    0.540302
      1   -0.416147
      2   -0.989992
      dtype: float64
```

1.4. Analítica de datos - Pandas

- Series:
  - También se puede realizar acciones de tratamiento de datos puramente específicas de Data Science como por ejemplo:
    - Filtrar valores:

      serie[condición]

    - Ordenar la serie:

      serie.sort_values(ascending=bool)

    - Limpiar valores nulos:

      serie.dropna()

```
[13]: import pandas as pd

[27]: s = pd.Series([2,4,5,3,None])

[28]: s[s < 3]

[28]: 0    2.0
      dtype: float64

[29]: s.sort_values(ascending=True)

[29]: 0    2.0
      3    3.0
      1    4.0
      2    5.0
      4    NaN
      dtype: float64

[30]: s.dropna()

[30]: 0    2.0
      1    4.0
      2    5.0
      3    3.0
      dtype: float64
```

1.4. Analítica de datos - Pandas

- DataFrame:
  - Pueden crearse a través de listas, diccionarios, Series, Arrays de NumPy u otros DataFrame.
  - Implementa una tabla donde cada columna es una Serie y como tal tiene dos índices para acceder a sus elementos.

```
[13]: import pandas as pd

[36]: datos = [['A',3],['B',2],['C',1]]

[39]: df = pd.DataFrame(datos, columns=['Nombre','Valor'])

[40]: df
```

[40]:

	Nombre	Valor
0	A	3
1	B	2
2	C	1

```
[13]: import pandas as pd

[36]: datos = [['A',3],['B',2],['C',1]]

[44]: df = pd.DataFrame(datos, columns=['Nombre','Valor'])

[48]: df['Nombre']

[48]: 0    A
      1    B
      2    C
      Name: Nombre, dtype: object

[49]: df['Nombre'][0]

[49]: 'A'
```

1.4. Analítica de datos - Pandas

- DataFrame:
  - Al tratarse de tablas las operaciones más comunes son las consultas, selección, adición y borrado tanto de filas como de columnas.

```
[2]: import pandas as pd

[1]: datos = [['Pedro',21,8.0],['Ana',22,7.4],['Luis',37,6.1]]

[4]: df = pd.DataFrame(datos, columns=['Nombre','Edad','Nota'])

[7]: df['Edad']

[7]: 0    21
     1    22
     2    37
     Name: Edad, dtype: int64

[10]: df.loc[0]

[10]: Nombre    Pedro
      Edad         21
      Nota        8.0
      Name: 0, dtype: object
```

```
[73]: import pandas as pd

[74]: datos = [['Pedro',21,8.0],['Ana',22,7.4],['Luis',37,4.1]]

[75]: df = pd.DataFrame(datos, columns=['Nombre','Edad','Nota'])

[76]: df['Aprobado']=pd.Series([True, True, False])
      df.pop('Nota')

[76]: 0    8.0
      1    7.4
      2    4.1
      Name: Nota, dtype: float64

[77]: nueva_linea = {'Nombre':'Julia', 'Edad':25, 'Aprobado':True}
      df = df.append(nueva_linea, ignore_index=True)
      df.drop(0)

      C:\Users\Javier\AppData\Local\Temp\ipykernel_16732\1032265573.py:2: FutureWarning: The frame.append method is deprec
      ated and will be removed from pandas in a future version. Use pandas.concat instead.
        df = df.append(nueva_linea, ignore_index=True)
```

[77]:

	Nombre	Edad	Aprobado
1	Ana	22	True
2	Luis	37	False
3	Julia	25	True

1.4. Analítica de datos - Pandas

- DataFrame:
  - Tiene las mismas funciones importantes que las series para cálculo de medias, máximo, desviaciones, etc.
  - Tiene los mismos operadores que las series, pero es necesario utilizar la referencia a la columna con la que se quiere operar:

    dataframe[columna]

  - Otras funciones útiles:

    dataframe.rename(columns=columnas, index=filas)

    dataframe.groupby(columna).groups / .agg(función)

    np.mean
    np.std
    np.sum
    np.min
    np.max

```
[99]: import pandas as pd
      import numpy as np

[100]: datos = [['Pedro',21,8.0,'H'],
                ['Ana',22,7.4,'M'],
                ['Luis',37,4.1,'H']]

[101]: df = pd.DataFrame(datos,
           columns=['Nombre','Edad','Nota','Sexo'])

[102]: df = df.rename(columns={'Nombre':'Name'}, index={0:10000})
       df
```

[102]:

	Name	Edad	Nota	Sexo
10000	Pedro	21	8.0	H
1	Ana	22	7.4	M
2	Luis	37	4.1	H

```
[103]: df.groupby('Sexo').groups

[103]: {'H': [10000, 2], 'M': [1]}

[104]: df.groupby('Sexo').agg(np.mean)
```

[104]:

	Edad	Nota
Sexo		
H	29.0	6.05
M	22.0	7.40