

DATA SCIENCE

Data Science

DS-BUE-8/Clase 04



Hoja de ruta de hoy

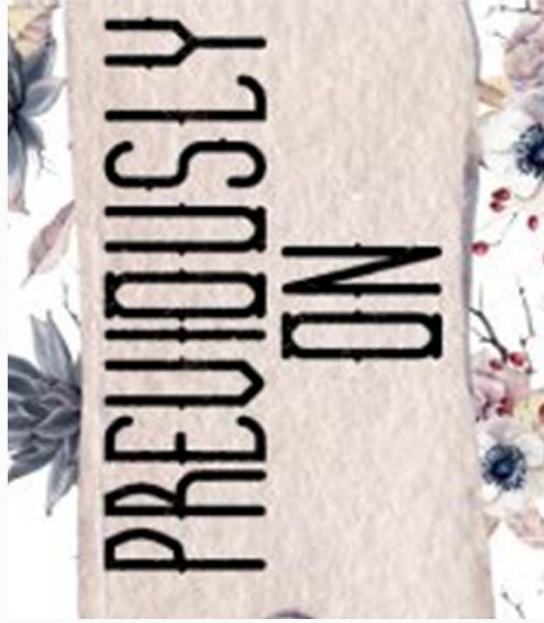
19:00	Repaso general
19:50	Librería Pandas
20:30	Break
20:45	Hands on: Pandas
21:30	Puesta en común
21:55	Cierre y próximo encuentro

DS-BUE-8/Clase 04



¿Comentarios?





¿Qué es Data Science?

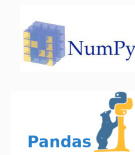
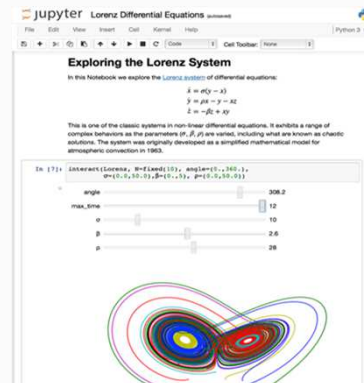


¿Qué es Data Science?



Conocimientos ¿previos?

- Probabilidad y Estadística
- Programación (cualquier lenguaje)
- Uso de la terminal (Windows, Mac, Ubuntu)
- [Recomendación] Entender algo de inglés



Metodología, entregas y proyectos

ADQUISICIÓN Y EXPLORACIÓN

1

Exploración de datos



Pandas

matplotlib

2

Feature engineering



3

Regresión

4

Optimización de parámetros

5

Procesamiento del lenguaje natural

6

Sistema de recomendación

7

Publicación de modelos



IBM Cloud

MODELADO

B-Learning:

- Aprender haciendo
- Aprendizaje basado en proyectos



ENTREGA 1

Duración estimada: 4 semanas

ENTREGA 2

Duración estimada: 2 semanas

ENTREGA 3

Duración estimada: 4 semanas

ENTREGA 4

Duración estimada: 2 semanas

ENTREGA 5

Duración estimada: 5 semanas

ENTREGA 6

Duración estimada: 4 semanas

ENTREGA 7

Duración estimada: 3 semanas





Definiciones básicas

Probabilidad	Estadística
¿Qué tan posible es que ocurra un evento determinado?	Análisis de los eventos gobernados por la probabilidad.
	Descriptiva
	Obtiene, organiza, presenta y describe a un conjunto de datos.
	Inferencial
	Métodos y procedimientos que por medio de la inducción determina propiedades de una población estadística, a partir de una parte de esta.



Medidas de tendencia central

Moda

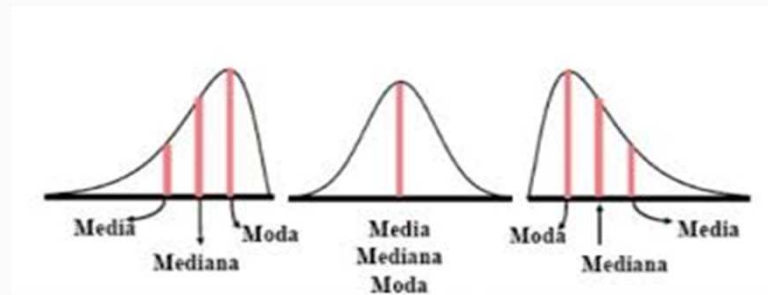
Número con mayor frecuencia en conjunto de datos

Media

Promedio de conjunto de datos numéricos

Mediana

Centro de conjunto de datos numérico



Nos sirven para describir características básicas de un estudio con datos cuantitativos.

Las medidas de tendencia central son medidas estadísticas que pretenden resumir en un solo valor a un conjunto de valores.



Medidas de dispersión o variabilidad

Varianza

Mide la mayor o menor dispersión de los valores de la variable respecto a la media aritmética

$$Var(X) = \frac{\sum_1^n (x_i - \bar{X})^2}{n}$$

Rango inter - cuartil

Se mide como la diferencia entre el tercer y primer cuartil de un conjunto de datos

$$RI = P_{75} - P_{25} = Q_3 - Q_1$$

Coefficiente de variación

Permite determinar la razón existente entre la desviación estándar y la media.

$$CV = \frac{\sigma}{\bar{X}}$$

Parámetros estadísticos que indican como se alejan los datos respecto de la media aritmética. Sirven como indicador de la variabilidad de los datos.

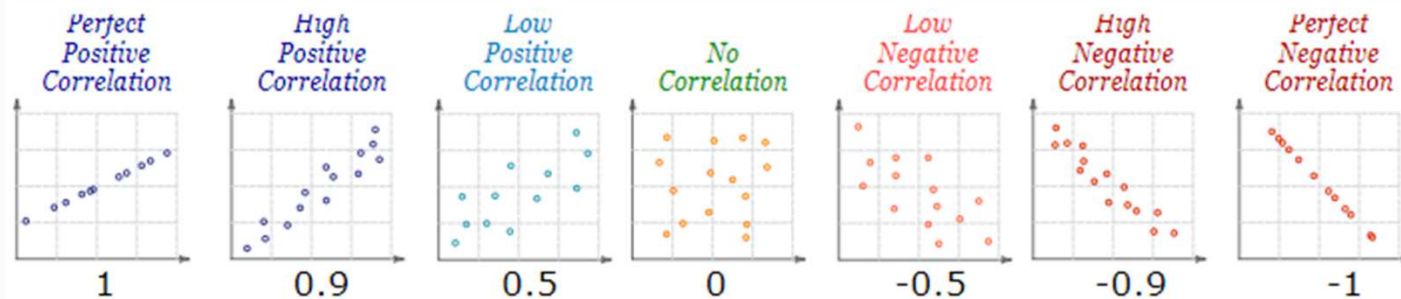


Coeficiente de correlación

Medida de relación lineal entre variables

$$r = \frac{n \cdot \sum x_i \cdot y_i - \sum x_i \cdot \sum y_i}{\sqrt{[n \cdot \sum x_i^2 - (\sum x_i)^2] \cdot [n \cdot \sum y_i^2 - (\sum y_i)^2]}}$$

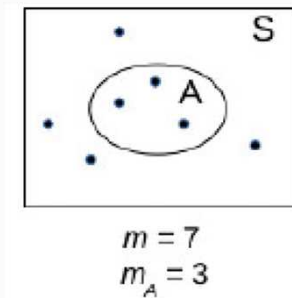
$$-1 \leq r \leq 1$$



Teoría de la Probabilidad

- 1. **S** espacio muestral: conjunto con los posibles resultados de un experimento.
- 2. **A, B, C**: eventos a los cuales vamos a asignarles probabilidad
- 3. **P** función de probabilidad

$$\frac{m_A}{m} \rightarrow \mathbb{P}(A)$$



P(A) representa el porcentaje de veces que esperamos que A ocurra en infinitas repeticiones



Variables Aleatorias

Una variable aleatoria **X** es una función definida sobre el espacio muestral que toma valores en los reales:

$$X: S \rightarrow R$$

Ejemplo:

Sea X la variable aleatoria que representa el número de sellos al lanzar dos monedas

$$S = \{(c,s), (s,c), (c,c), (s,s)\}$$

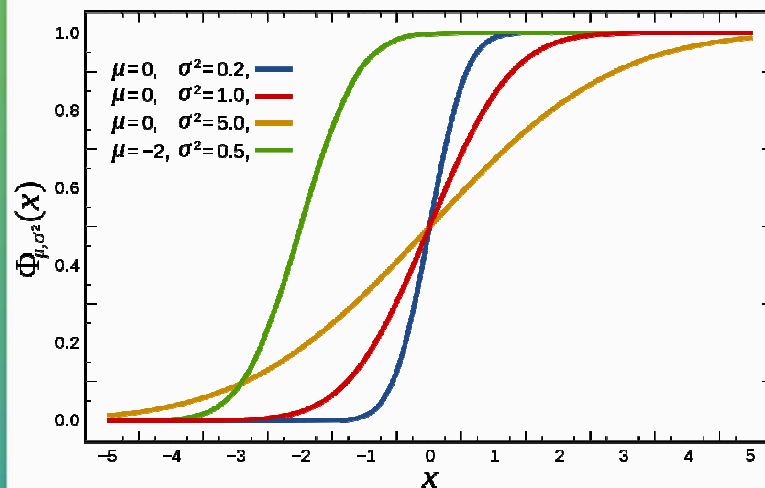
Entonces, ¿quién es X ?

Pista: Responda esto pensando que una variable aleatoria es un valor numérico que corresponde a un resultado de un experimento aleatorio.



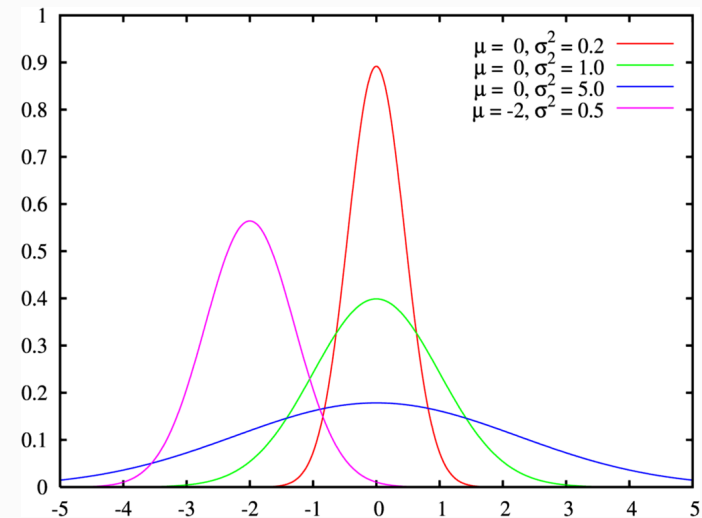
Funciones de X

Función de Distribución



La distribución de probabilidad de una variable aleatoria es una función que asigna a cada suceso definido sobre la variable la probabilidad de que dicho suceso ocurra

Función de Densidad



La función de densidad de probabilidad, función de densidad, o, simplemente, densidad de una variable aleatoria continua describe la probabilidad relativa según la cual dicha variable aleatoria tomará determinado valor.

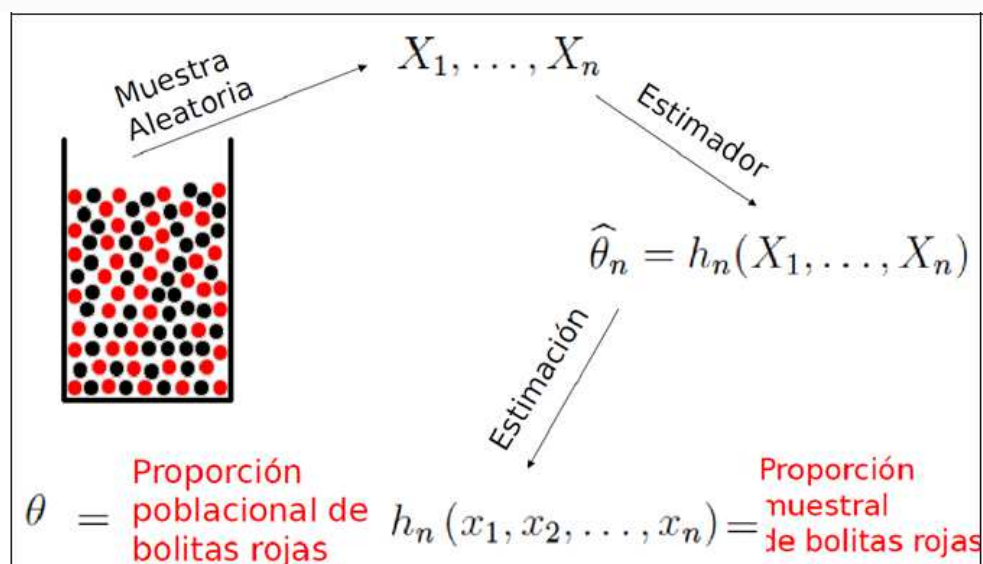


Probabilidad e Inferencia

Tratar de estimar o inferir mediante una muestra (aleatoria) el valor (desconocido) de un parámetro poblacional

Población: conjunto de individuos, objetos, elementos o fenómenos en los cuales puede presentarse determinada característica

Muestra: Subconjunto de unidades provenientes de la población, que con algún criterio o sin él, son seleccionadas a los efectos de ser estudiada en una o más características

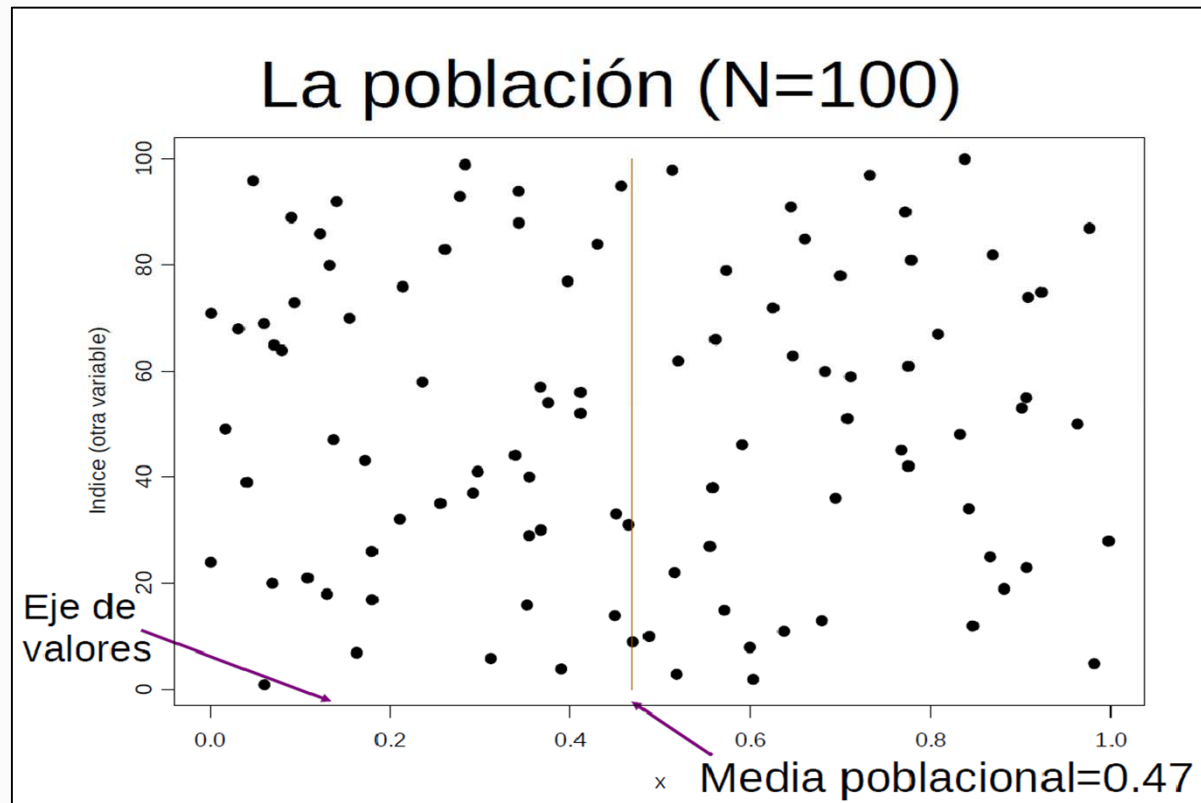


Error Cuadrático Medio

estimador mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima

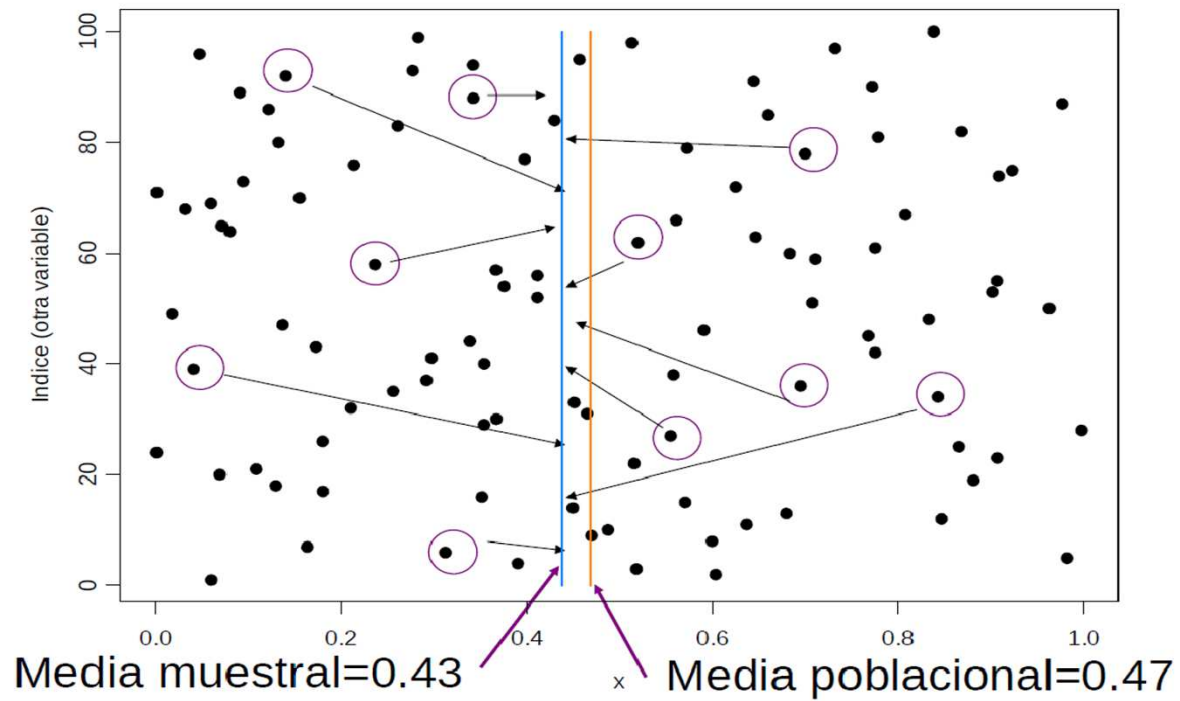


Ejemplo



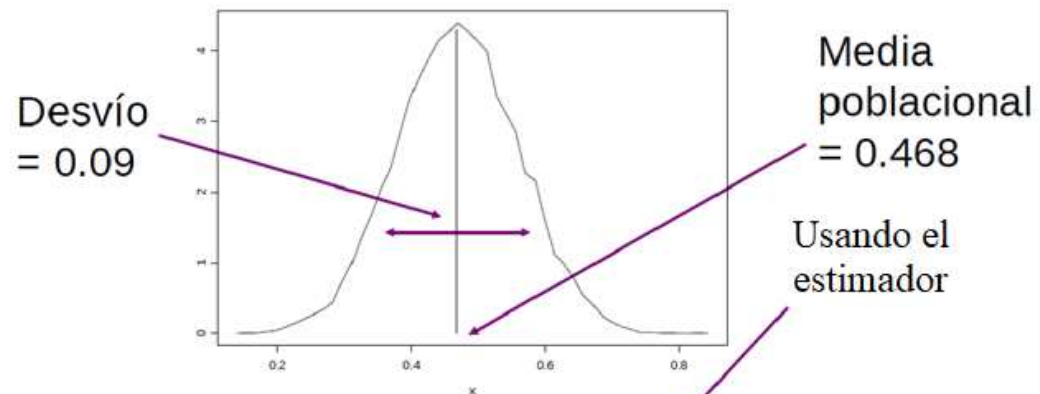
Ejemplo

La muestra (n=10)



Ejemplo

Repito el experimento 10000 veces



Min.	st Qu.	Median	Mean	3rd Qu.	Max.
0.15705	0.40697	0.46900	0.46879	0.53011	0.82568





Resumen de Python

- Lenguaje de programación de alto nivel
- Diseñado para ser fácil de leer y simple de implementar
- Es código abierto
- Lenguaje de tipo imperativo
- Orientado a objetos
- Interpretado
- Posee una gran comunidad de desarrolladores
- Puede ejecutarse en Mac, Windows y sistemas Unix; también ha sido portado a máquinas virtual JAVA y .NET.



Tipos de datos



Enteros

Son los números que usamos para contar, el 0 y los negativos

- -1
- 0
- 1
- 2

Floats

Son los números enteros, pero con coma.

En python se introducen usando puntos.

- 5.1
- -1.3
- 5.8
- 10.0

Strings

Cualquier carácter que podemos escribir con el teclado de la compu.

Se introducen entre comillas "" o ''.

- 'a'
- ''
- '&'
- 'Hola mundo'

Booleanos

Variables de verdad

Se introducen con una sentencia o escribiendolas

- True
- False
- 1 == 2
- 1==1



Elementos de Python

Listas: es una estructura de datos que contiene una colección o secuencia de datos. Los datos o elementos de una lista deben ir separados con una coma y todo el conjunto entre corchetes.

```
listaEstaciones = ["Invierno", "Primavera", "Verano", "Otoño"]
```



Elementos de Python

Tuplas: permite tener agrupados un conjunto inmutable de elementos, es decir, en una tupla no es posible agregar ni eliminar elementos.

```
dias = ("Lunes", "Martes", "Miércoles", "Jueves", "Viernes",  
        "Sábado", "Domingo")
```



Elementos de Python

Diccionarios: son objetos que contienen una lista de parejas de elementos. De cada pareja un elemento es la clave, que no puede repetirse, y, el otro, un valor asociado.

```
capitales = {"Chile" : "Santiago", "España" : "Madrid",  
            "Francia" : "París"}
```



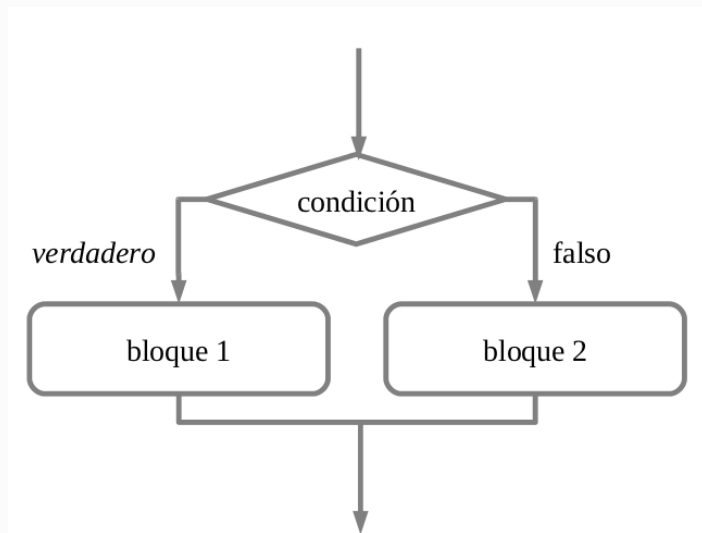
Operadores Aritméticos

Operación	Operador	Ejemplo
Suma	+	$3 + 5.5 = 8.5$
Resta	-	$4 - 1 = 3$
Multiplicación	*	$3 * 6 = 18$
Potencia	**	$3 ** 2 = 9$
División (cociente)	/	$15.0 / 2.0 = 7.5$
División (parte entera)	//	$15.0 // 2.0 = 7$
División (resto)	%	$7 \% 2 = 1$

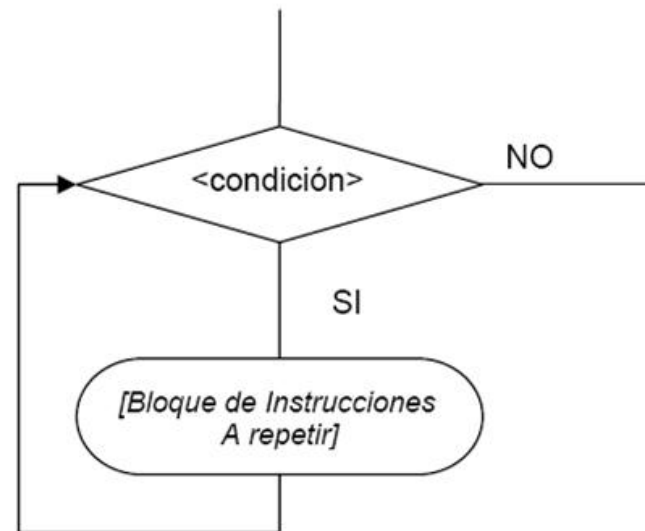


Estructuras de Control

Selección



Iteración



Ejemplo

Tenemos un módulo de una app de IoT que controla parámetros de un climatizador que solo se instalará provisionalmente en una habitación por siete días continuos. Este módulo recibe, por hora, información "en tiempo real" de Internet acerca de la temperatura exterior actual y valores de luminosidad a través de un sensor de luz externo. La misión es mantener la habitación 5°C por debajo de la temperatura exterior si es un día soleado y 3°C por debajo si es un día no-soleado o de noche. Suponga que el módulo IoT posee un reloj interno y un capturador para el sensor de luz que procesa la señal y la categoriza como: "S" si es un día soleado, "T" para un día no soleado y "N" para la noche.

NOTA: Por simplicidad para la solución, suponga que se le proporciona una función llamada `paso(intervalo)` que puede ser utilizada como función de espera, una función llamada `temperatura_exterior()` que se conecta a un endpoint de temperaturas en Internet y otra función llamada `luminosidad()` que se conecta a un sensor de luz externo.



Ejemplo

Solución:

Entrada: Información de temperatura del exterior en °C y luminosidad, digamos X y L , respectivamente.

Salida: Temperatura controlada por hora durante 7 días continuos, digamos y .

El valor de y puede variar en el intervalo $X - 5 \leq y \leq X - 3$, de acuerdo a las siguientes condiciones:

- Cuando $L = "S"$, entonces $y = X - 5$
- Cuando $L = "T"$ o $L = "N"$, entonces $y = X - 3$

```
def temperatura_exterior():  
    a = np.arange(10,40)  
    return np.random.choice(a)  
def luminosidad():  
    l = ["S", "T", "N"]  
    a = np.random.choice(np.arange(3))  
    return l[a]  
def paso(intervalo):  
    time.sleep(3600*intervalo)
```

```
dias = 7  
intervalo = 1  
fin = dias * (24 / intervalo)  
for i in range(int(fin)):  
    X = temperatura_exterior()  
    print("Temperatura exterior = ", X)  
    L = luminosidad()  
    print("Luminosidad = ", L)  
    if (L == "S"):  
        y = X - 5  
    elif (L == "T" or L == "N"):  
        y = X - 3  
    else:  
        None  
    print("Temperatura ajustada = ", y)  
    paso(intervalo)
```





Descripción

- Librería de Python que agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel.



Numpy vs Listas

**Menos
memoria**



Rápido



Conveniente

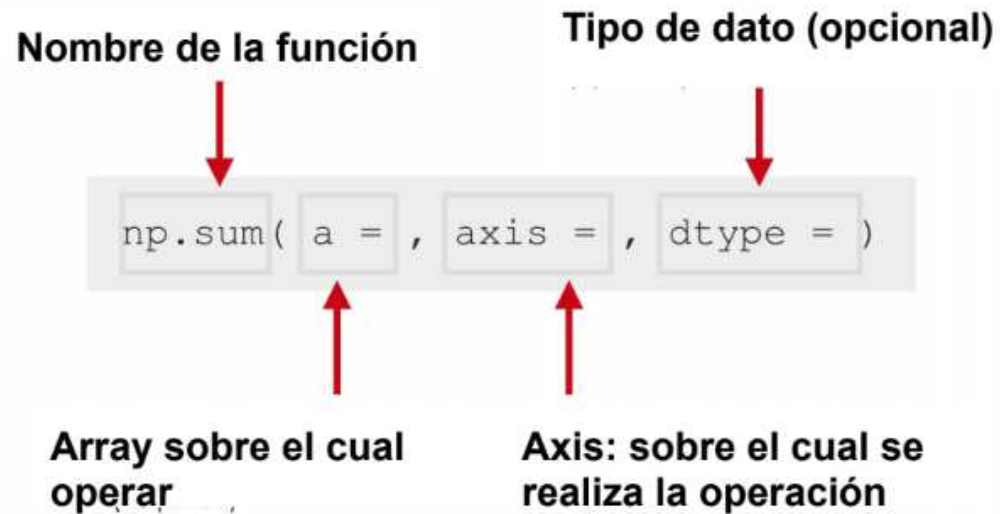


Funciones NumPy

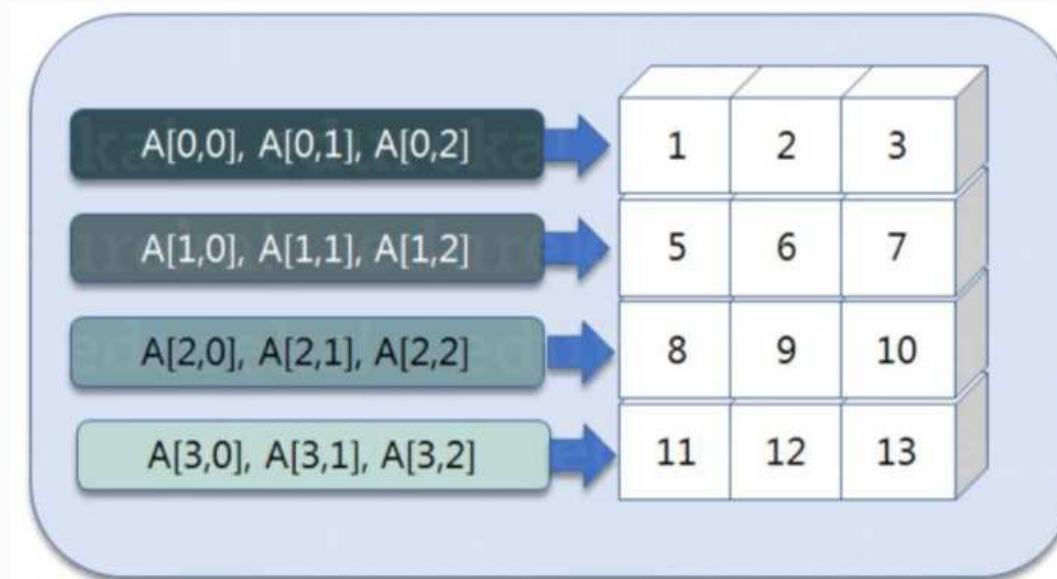
Nombre de la función **Tipo de dato (opcional)**

```
np.sum( a = , axis = , dtype = )
```

**Array sobre el cual
operar** **Axis: sobre el cual se
realiza la operación**



Cómo se indexa



Operaciones con arreglos

Metadatos

```
a = np.array([6, 1, 3, 9, 8])
```

```
a.dtype
```

```
dtype('int32')
```

```
a.shape
```

```
(5,)
```

```
a.size
```

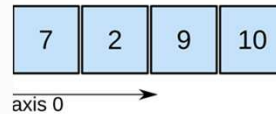
```
5
```



Operaciones con arreglos

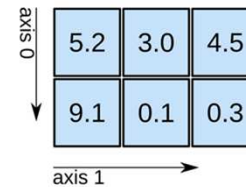
Ejemplos

1D array



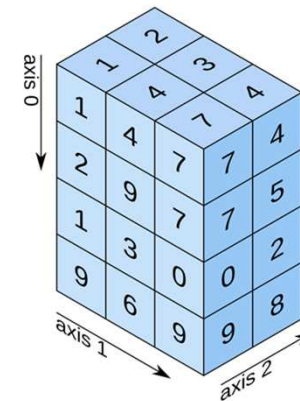
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

```
b = np.array([[5.2, 3.0, 4.5],[9.1, 0.1, 0.3]])
```

```
c = np.array([[[1,2], [4,3], [7,4]],  
              [[2,3], [9,4],[7,5]],  
              [[1,3], [3,4],[0,2]],  
              [[9,3], [6,4],[9,8]])
```



Cortes

```
>>> a[0,3:5]  
array([3,4])
```

```
>>> a[4:,4:]  
array([[44, 45],  
       [54, 55]])
```

```
>>> a[:,2]  
array([2,12,22,32,42,52])
```

```
>>> a[2::2,::2]  
array([[20,22,24]  
       [40,42,44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55



Cortes y Máscaras

```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]  
array([1, 12, 23, 34, 45])
```

```
>>> a[3:, [0,2,5]]  
array([[30, 32, 35],  
       [40, 42, 45],  
       [50, 52, 55]])
```

```
>>> mask = np.array([1,0,1,0,0,1], dtype=bool)  
>>> a[mask, 2]  
array([2, 22, 52])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55



Filtros Booleanos



```
B = np.array([[42,56,89,65],  
              [99,88,42,12],  
              [55,42,17,18]])
```

```
print(B>=42)
```

```
[[ True  True  True  True]
```

```
 [ True  True  True False]
```

```
 [ True  True False False]]
```

```
C = np.array([123,188,190,99,77,88,100])
```

```
A = np.array([4,7,2,8,6,9,5])
```

```
R = C[A<=5]
```

```
print(R)
```

```
[123 190 100]
```



Tipos de arreglos

Ejemplos:

```
p1 = np.zeros(6)
p1
array([0., 0., 0., 0., 0., 0.])
```

```
p2 = np.zeros([6,6])
p2
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.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

```
p3 = np.ones(6, dtype='int64')
p3
array([1, 1, 1, 1, 1, 1], dtype=int64)
```

```
p5 = np.linspace(1,2,5)
p5
array([1.   , 1.25, 1.5  , 1.75, 2.   ])
```

```
p4 = np.arange(3.0, 15.0, 2)
p4
array([ 3.,  5.,  7.,  9., 11., 13.])
```

```
p6 = np.random.random(3)
p6
array([0.8500509 , 0.30139955, 0.99669799])
```



Operaciones con arreglos

```
o1 = np.array([43, 22, 65, 21, 9])  
o2 = np.array([12, -9, 41, 23, -9])
```

```
o1 + o2
```

```
array([ 55,  13, 106,  44,   0])
```

```
o1 - o2
```

```
array([31, 31, 24, -2, 18])
```

```
o1 * o2
```

```
array([ 516, -198, 2665,  483, -81])
```

```
o1 / o2
```

```
array([ 3.58333333, -2.44444444,  1.58536585,  0.91304348, -1.        ])
```

```
o1 < o2
```

```
array([False, False, False,  True, False])
```

```
o1 ** 2
```

```
array([1849,  484, 4225,  441,   81], dtype=int32)
```



Operaciones con arreglos

```
m = np.array([[43, 22, 65, 21, 9], [12, -9, 41, 23, -9]])
```

```
print(m)
np.sum(m, axis = 0)

[[43 22 65 21  9]
 [12 -9 41 23 -9]]

array([ 55,  13, 106,  44,   0])
```

```
print(m)
np.sum(m, axis = 1)

[[43 22 65 21  9]
 [12 -9 41 23 -9]]

array([160,  58])
```

```
m.reshape(5,2)

array([[43, 22],
       [65, 21],
       [ 9, 12],
       [-9, 41],
       [23, -9]])
```

```
m.ravel()

array([43, 22, 65, 21,  9, 12, -9, 41, 23, -9])
```



Operaciones con arreglos

```
a = np.array([1.0, 5.0, np.pi, -0.1, 9.0, 3.14])
```

```
a
```

```
array([ 1.      ,  5.      ,  3.14159265, -0.1      ,  9.      ,  
        3.14      ])
```

```
print("Mínimo: ", a.min())
```

```
print("Máximo: ", a.max())
```

```
print("Índice del elemento mínimo: ", a.argmin())
```

```
print("Índice del elemento máximo: ", a.argmax())
```

```
Mínimo: -0.1
```

```
Máximo: 9.0
```

```
Índice del elemento mínimo: 3
```

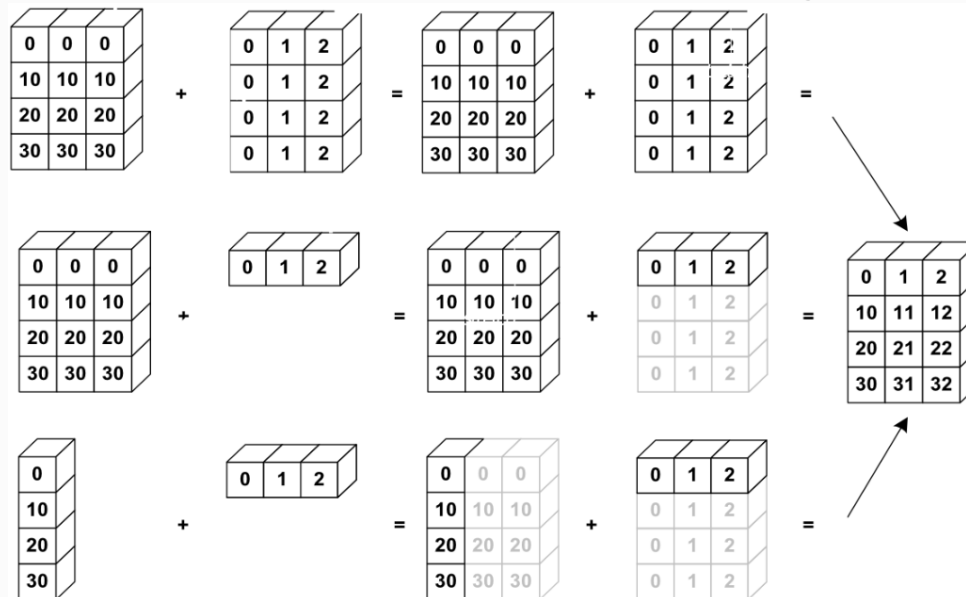
```
Índice del elemento máximo: 4
```



Broadcasting

Operaciones básicas en arreglos numpy (suma, resta, etc) **son elemento a elemento**.
Esto funciona con arreglos del **mismo tamaño**.

No obstante!, también es posible hacer operaciones con matrices de diferentes tamaños si Numpy puede transformar estos arreglos en arreglos del mismo tamaño:
esta conversión se llama **broadcasting**.



Valores especiales

numpy provee de varios valores especiales: `np.nan`, `np.Inf`, `np.Infinity`, `np.inf`, `np.infty`,...

numpy usa el estándar IEEE de números flotantes para aritmética (IEEE 754). Esto significa que *Not a Number* no es equivalente a *infinity*. También, *positive infinity* no es equivalente a *negative infinity*. Pero *infinity* es equivalente a *positive infinity*.



¿Dudas?





Pandas

Descripción

- Librería de Python que está construida como una extensión de NumPy para manipulación y análisis de datos.
- Permite representar, explorar y visualizar los datasets.
- Posee tres tipos de estructuras:

Series

Dataframes

~~Panel, Panel4D, PanelND~~



Series

- Un objeto "Series" es un arreglo unidimensional con datos indexados

X = pd.Series([10, 20, 14, 11])

- Pero, ¿qué los diferencia de los array de NumPy?

Reflexione...



Series

- La clave está en la *indexación*:
 - ❑ Si no especificamos una etiqueta para los índices, Pandas indexará los elementos de X con índices **[0, 1, ... len(X) -1]**. Igual que un array numpy.
 - ❑ Si queremos proporcionar las etiquetas de los índices:
X = pd.Series([6,3,4,6], index=['a', 'b', 'c', 'd'])



Series

- La clave está en la *indexación*:
 - ❑ Una serie contiene los datos (values) en un objeto de tipo array de numpy

```
X.values  
  
array([6, 3, 4, 6], dtype=int64)
```

- ❑ Una serie contiene los índices (index) en un objeto del propio Pandas

```
X.index  
  
Index(['a', 'b', 'c', 'd'], dtype='object')
```



Series

- ...pero hay algo más importante...
 - ❑ **Las operaciones** que pueden ser aplicadas a un array de numpy no son las mismas que pueden ser aplicadas a una serie de pandas.
 - ❑ Los arrays funcionan muy bien cuando tenemos datos numéricos y limpios.
 - ❑ Son poco útiles cuando tenemos datos faltantes o poco estructurados (clases, continuos, discretos, etc).



Series

- ...Ahora...

☐ Revisemos el notebook:

[intro-series-pandas.ipynb](#)



Dataframes

- Un objeto "Dataframe" es un arreglo bidimensional que puede tener columnas con diferentes tipos de datos.
- Técnicamente es un conjunto de series.

The diagram shows a DataFrame table with 10 rows and 8 columns. A blue bracket on the left side of the rows is labeled "index labels". A red bracket above the columns is labeled "column names". An orange bracket on the right side of the data cells is labeled "data".

	Mountain	Height (m)	Range	Coordinates	Parent mountain	First ascent	Ascents bef. 2004	Failed attempts bef. 2004
0	Mount Everest / Sagarmatha / Chomolungma	8848	Mahalangur Himalaya	27°59'17"N 86°55'31"E	NaN	1953	>>145	121.0
1	K2 / Qogir / Godwin Austen	8611	Baltoro Karakoram	35°52'53"N 76°30'48"E	Mount Everest	1954	45	44.0
2	Kangchenjunga	8586	Kangchenjunga Himalaya	27°42'12"N 88°08'51"E	Mount Everest	1955	38	24.0
3	Lhotse	8516	Mahalangur Himalaya	27°57'42"N 86°55'59"E	Mount Everest	1956	26	26.0
4	Makalu	8485	Mahalangur Himalaya	27°53'23"N 87°05'20"E	Mount Everest	1955	45	52.0
5	Cho Oyu	8188	Mahalangur Himalaya	28°05'39"N 86°39'39"E	Mount Everest	1954	79	28.0
6	Dhaulagiri I	8167	Dhaulagiri Himalaya	28°41'48"N 83°29'35"E	K2	1960	51	39.0
7	Manaslu	8163	Manaslu Himalaya	28°33'00"N 84°33'35"E	Cho Oyu	1956	49	45.0
8	Nanga Parbat	8126	Nanga Parbat Himalaya	35°14'14"N 74°35'21"E	Dhaulagiri	1953	52	67.0
9	Annapurna I	8091	Annapurna Himalaya	28°35'44"N 83°49'13"E	Cho Oyu	1950	36	47.0



Ejemplo

```
# Importamos Pandas
import pandas as pd
```

Creando un "dataframe" a partir de un diccionario

```
data = {'Nombre' : ["Luis", "Maria", "Florecia", "Dayana"],
        'Pais'   : ["Mexico", "Venezuela", "Argentina", "Costa Rica"],
        'Edad'   : [28, 25, 19, 23]}

df = pd.DataFrame(data)
df
```

	Nombre	Pais	Edad
0	Luis	Mexico	28
1	Maria	Venezuela	25
2	Florecia	Argentina	19
3	Dayana	Costa Rica	23



Importar datos

pd.read_csv(filename) - De un archivo CSV

pd.read_table(filename) - Desde un archivo de texto delimitado (como TSV)

pd.read_excel(filename) - De un archivo Excel

pd.read_sql(query, connection_object) - Lee desde una BaseDeDatos/Tabla SQL

pd.read_json(json_string) - Lee desde una cadena, URL o archivo con formato JSON

pd.read_html(url) - Analiza una URL html, una cadena o un archivo y extrae tablas a una lista

pd.read_clipboard() - Toma el contenido del porta papeles

pd.DataFrame(dict) - Desde un diccionario



Exportar datos

df.to_csv(filename) - Escribir en un archivo CSV

df.to_excel(filename) - Escribir en un archivo Excel

df.to_sql(table_name, connection_object) - Escribir en una tabla SQL

df.to_json(filename) - Escribir en un archivo con formato JSON



Inspeccionar datos

df.head(n) - Primeras n filas del DataFrame

df.tail(n) - Las últimas n filas del DataFrame

df.shape() - Número de filas y columnas

df.info() - Índice, tipo de datos y memoria

df.describe() - Estadísticas resumidas de columnas numéricas

df.value_counts(dropna=False) - Ver valores y recuentos únicos



Seleccionar datos

df[col] - Devuelve la columna con la etiqueta col como Serie

df[[col1, col2]] - Devuelve columnas como un nuevo DataFrame

df.iloc[0] - Selección por posición

df.iloc[0,:] - Primera fila

df.iloc[0,0] - Primer elemento de la primera columna

df.loc['index_one'] - Selección por índice



Limpieza de datos

df.columns = ['a', 'b', 'c'] - Renombrar columnas

pd.isnull() - Comprueba valores nulos, devuelve Boolean Arrays

pd.notnull() - El opuesto a **pd.isnull()**

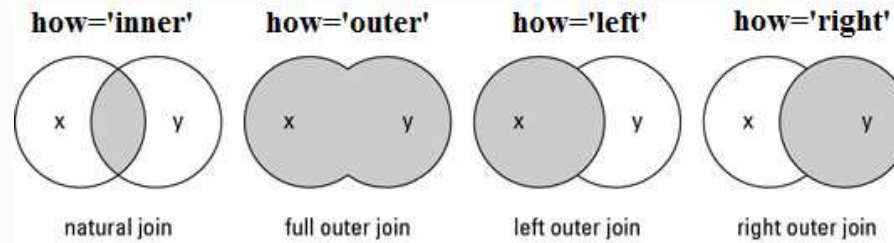
df.dropna() - Elimina todas las filas que contienen valores nulos

df.dropna(axis=1) - Elimina todas las columnas que contienen valores nulos

df.dropna(axis=1, thresh=n) - Elimina todas las filas que tienen menos de n valores no nulos



Uniones



adf			bdf		
x1	x2		x1	x3	
A	1	+	A	T	=
B	2		B	F	
C	3		D	T	

x1	x2	x3	
A	1	T	pd.merge(adf, bdf, how='left', on='x1')
B	2	F	
C	3	NaN	

x1	x2	x3	
A	1.0	T	pd.merge(adf, bdf, how='right', on='x1')
B	2.0	F	
D	NaN	T	

x1	x2	x3	
A	1	T	pd.merge(adf, bdf, how='inner', on='x1')
B	2	F	

x1	x2	x3	
A	1	T	pd.merge(adf, bdf, how='outer', on='x1')
B	2	F	
C	3	NaN	
D	NaN	T	



Filtro, orden y agrupamiento

`df[df[col]> 0.5]` - Filas donde la columna col es mayor que 0,5

`df[(df[col] > 0.5) & (df[col] < 0.7)]` - Filas donde $0.7 > \text{col} > 0.5$

`df.sort_values(col1)` - Ordenar los valores por la col1 en orden ascendente

`df.sort_values(col2, ascending=False)` - Ordena los valores por col2 en orden descendente

`df.sort_values([col1, col2], ascending=[True, False])` - Ordena los valores por la col1 de forma ascendente y luego por la col2 en orden descendente

`df.groupby(col)` - Devuelve un objeto groupby para los valores de una columna

`df.groupby([col1, col2])` - Devuelve un objeto groupby para valores de múltiples columnas

`df.groupby(col1)[col2].mean()` - Devuelve la media de los valores en col2, agrupados por los valores en col1 (la media se puede remplazar con casi cualquier función de la sección Estadísticas)

`df.apply(np.mean)` - Aplica la función `np.mean()` en cada columna



Estadísticas

Todas estas funciones también se pueden aplicar a una serie

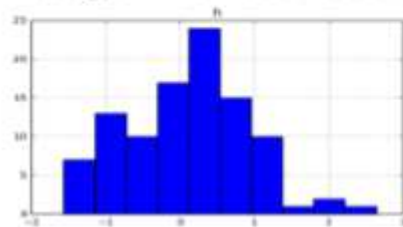
df.describe() - Resumen de estadísticas para columnas numéricas
df.mean() - Devuelve la media de todas las columnas
df.corr() - Devuelve la correlación entre columnas en un DataFrame
df.count() - Devuelve el número de valores no nulos en cada columna DataFrame
df.max() - Devuelve el valor más alto en cada columna
df.min() - Devuelve el valor más bajo en cada columna
df.median() - Devuelve la media de cada columna
df.std() - Devuelve la desviación estándar de cada columna



Gráficos

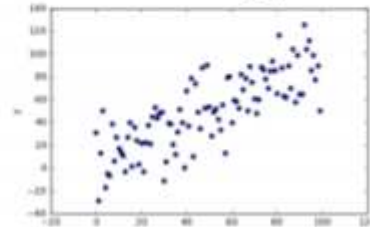
```
df.plot.hist()
```

Histogram for each column



```
df.plot.scatter(x='w',y='h')
```

Scatter chart using pairs of points



¿Dudas?

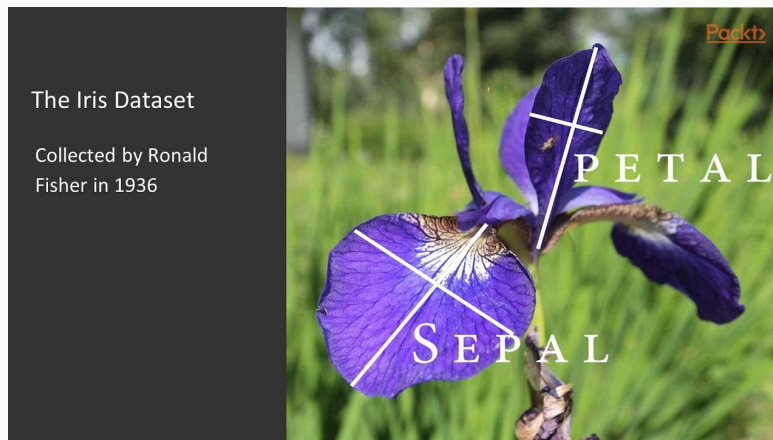


Desafíos Pandas

Para estos desafíos puede descargar los dos Pandas_Cheat_Sheet que están en Trello:

- Pandas_Cheat_Sheet.pdf
- Pandas_DataFrame_Notes.pdf

También puede revisar un resumen de la clase en el notebook llamado **Pandas_Tutorial.ipynb** que utiliza el dataset **RegularSeasonCompactResults.csv**:



Tenemos un dataset de **iris** y queremos identificar y eliminar las instancias que no tengan ningún valor en algunos de los campos: largo y ancho del pétalo y sépalo.

Diseñar una estrategia para reemplazar valores faltantes en el atributo “largo del pétalo”



Para el próximo encuentro...



matplotlib
Version 3.0.3

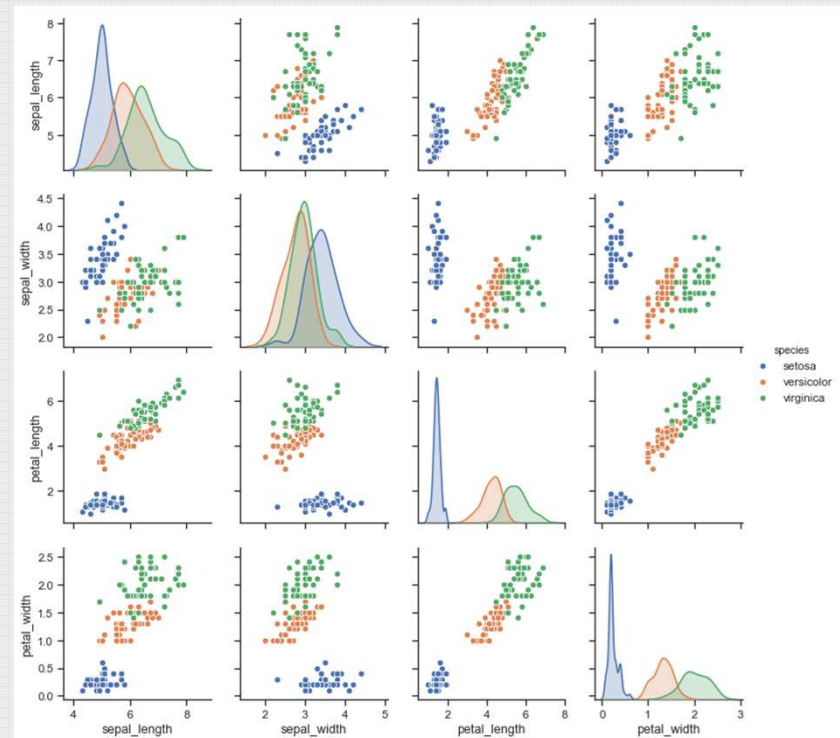


seaborn

¡Lindos Gráficos!

<https://matplotlib.org/>

<https://seaborn.pydata.org/>



Para la próxima...

- Ver los videos de la plataforma de Acámica de Matplotlib

