

The background is a solid red color. It features several decorative elements: a large, faint white circle on the left side; a cluster of small white dots in the bottom left corner; and several semi-transparent white circles of varying sizes scattered across the upper and middle sections. Some of these circles contain small white dots. The overall aesthetic is modern and tech-oriented.

EANT

Introducción a Data Analytics con Python

Quienes somos?



Karina Bartolome
 @karbartolome



Juan Gago
 @Juan_MGago



Romina Mendez
 @r0mymendez



Por qué es importante?

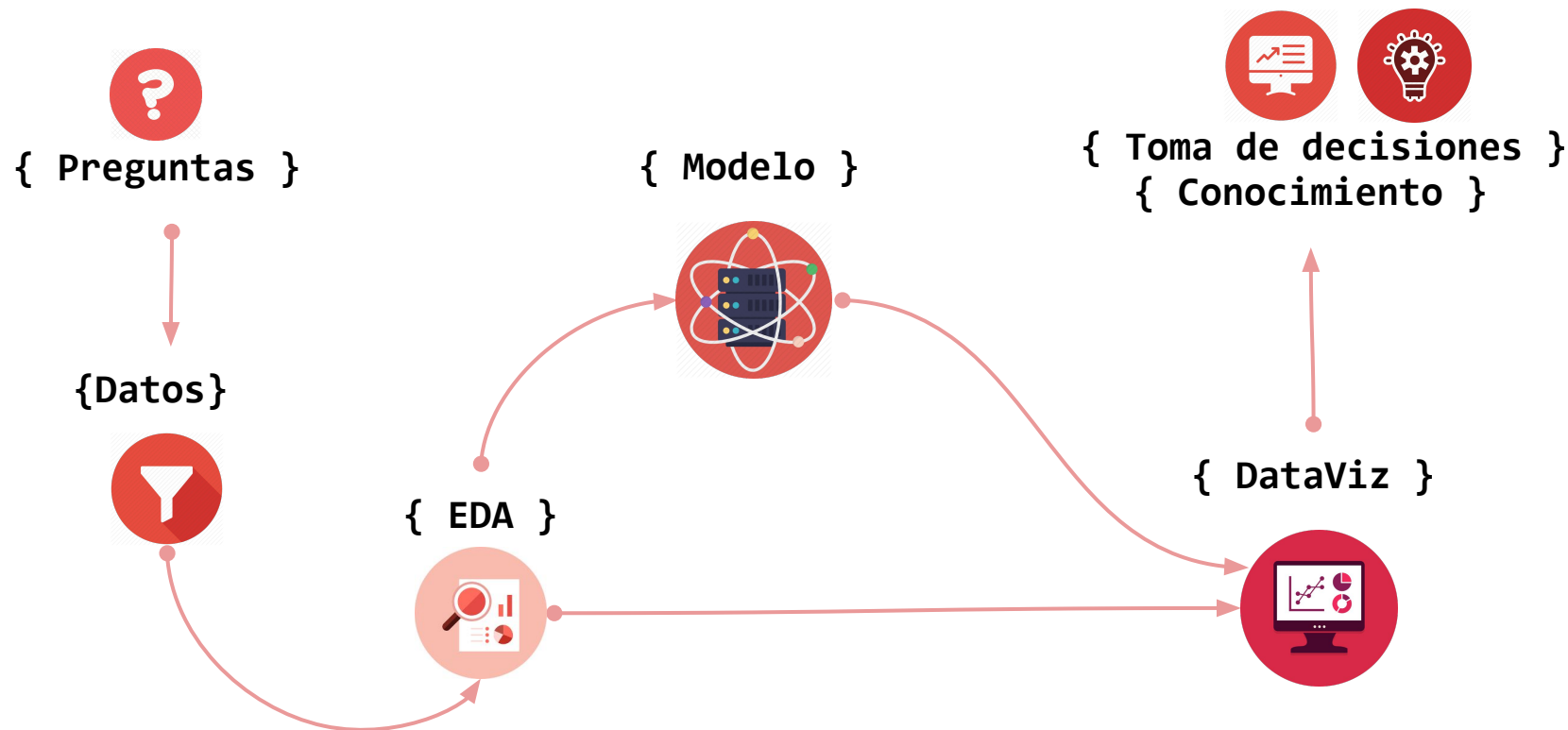
La visualización genera la posibilidad de comprender una gran cantidad de información muy rápidamente. Por lo cual la visualización...

- Permite identificar patrones y comunicar relaciones y significado.
- Puede inspirar a generar nuevas preguntas y una mayor exploración.
- Ayuda a identificar subproblemas.
- Es realmente buena para identificar tendencias y valores atípicos, descubrir o buscando puntos de datos interesantes o específicos en un campo más amplio.





Proceso de análisis de datos



¿Qué es Python?



Python es un lenguaje de programación de código abierto, orientado a objetos, muy simple y fácil de entender.

Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único.

Actualmente es uno de los lenguajes más utilizados para análisis de datos.

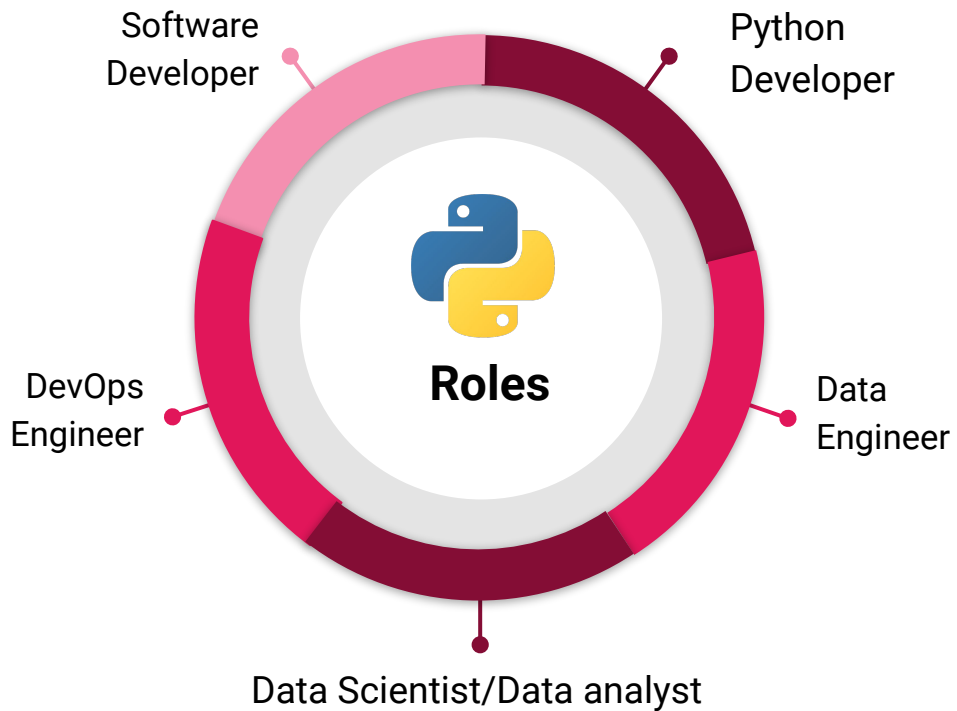
Historia de Python

La primera publicación de **Python** se produce hace ya 30 años.

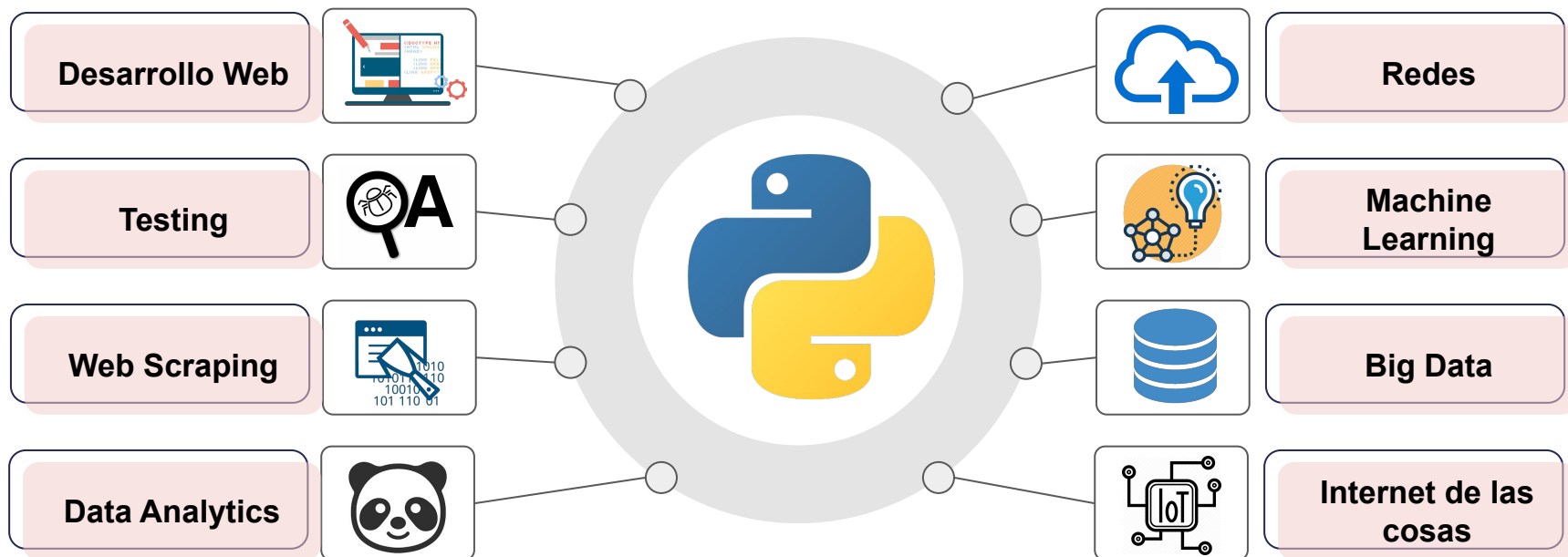
En 1989 Guido Van Rossum comenzó el desarrollo de un nuevo lenguaje de programación casi por hobby y tomando su nombre de el grupo humorístico británico Monty Python.



Roles que utilizan Python



Aplicaciones con Python



Google Colab

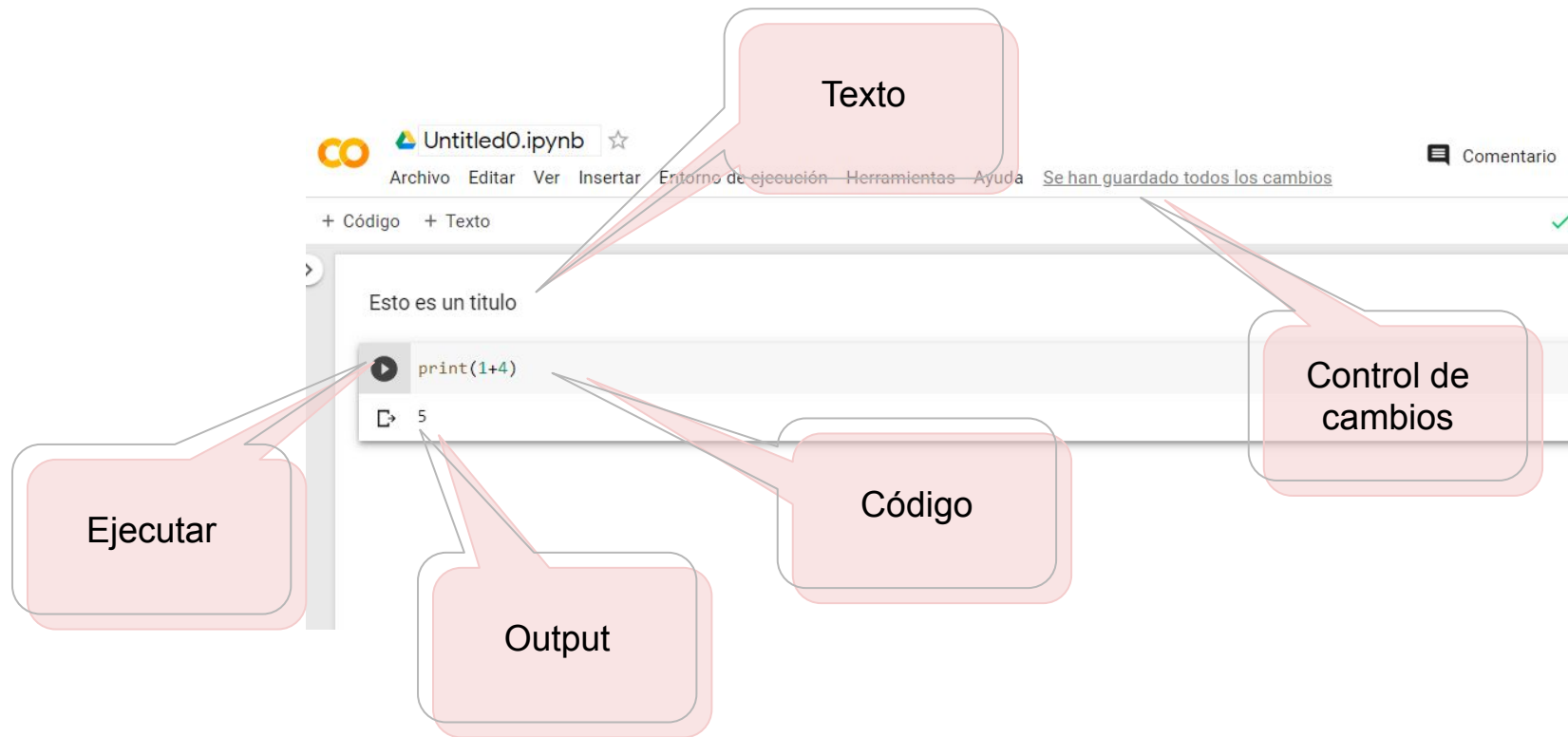
IDE: Su siglas hacen referencia a entorno de desarrollo integrado.

Es un marco de desarrollo amigable, el cual permite:

1. Autocompletado
2. Reconocimiento de sintaxis de programación
3. Depurador de errores



Colab notebooks



```
import seaborn as sns
```



Este paquete de visualización de datos de Python basada en matplotlib .

Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos.

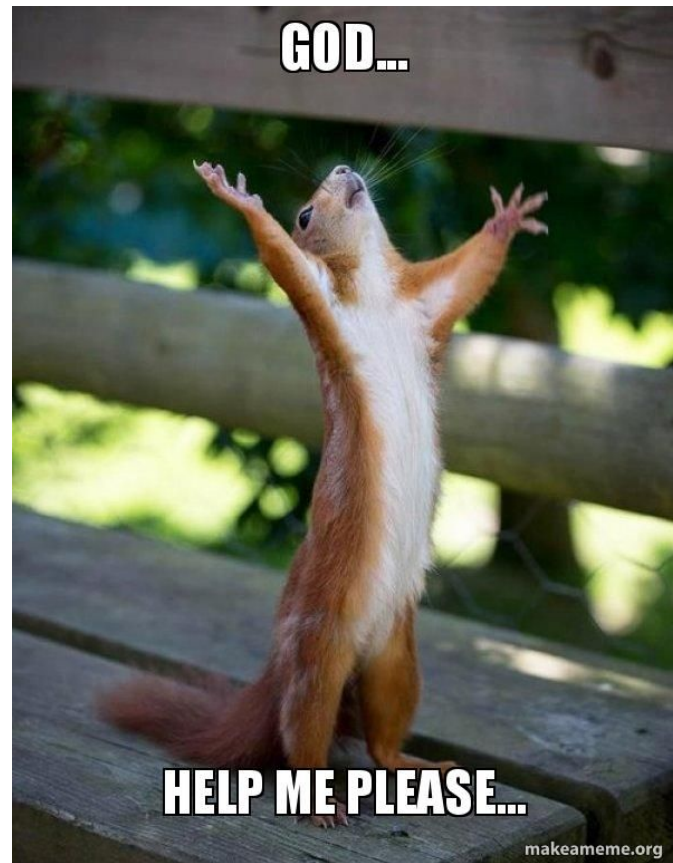
El paquete fue creado por Michael Waskom, que es investigador postdoctoral en el Centro de Ciencias Neurales de la Universidad de Nueva York

```
import plotly.express as px
```

Este paquete proporciona herramientas de gráficos, análisis y estadísticas, así como paquetes de código abierto con implementación de gráficos interactivos para Python , R , MATLAB , Perl , Julia, Arduino y REST .

Para conocer más de la implementación de Plotly en python: <https://plotly.com/python/>





HELP! Cheat Sheet

PANDAS: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

SEABORN: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf

[illegible]

Summarize Data

`df[["x", "value_counts()]]`
Count number of rows with each unique value of variable `x`

`df["x"].value_counts()`
of rows in `df`

`df["x"].nunique()`
of distinct values in a column

`df.describe()`
Basic descriptive statistics for each column (or Groupby)

pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (Dataframe columns, Series, Groupby, Grouping and rolling (see below) and produce single line **summary** of the groups. When applied to Dataframe, the result is returned as a pandas Series for each column. Examples:

<code>sum()</code> Sum total of each object.	<code>min()</code> Minimum value in each object.
<code>count()</code> Count non-Na/Null values in each object.	<code>max()</code> Maximum value in each object.
<code>mean()</code> Mean value of each object.	<code>name()</code> Name of each object.
<code>median()</code> Median value of each object.	<code>var()</code> Variance of each object.
<code>std()</code> Standard deviation of each object.	<code>quantile()</code> Quantiles of each object.

Group Data

`df.groupby("year")`
Returns a Groupby object.
`df.groupby("year").sum()`
Returns a Groupby object with column names in index

`df.groupby("year").size()`
Returns a Groupby object, grouped by values in index level (year)

All of the summary functions listed above can be applied to a group. Also, the following functions are available:

<code>size()</code> Size of each group	<code>agg(function)</code> Aggregate group using function
---	--

Windows

`df.rolling()`
Returns an `RollingObject` allowing summary functions to be applied cumulatively.

`df.rolling(window=5)`
Returns a rolling object allowing summary functions to be applied over windows of length 5.

Handling Missing Data

`df.dropna()`
Drop rows with any columns having NA/Null data.

`df.fillna(value)`
Replace all NA/Null data with value.

Make New Columns

`df.assign(x=lambda df: df.length*weight)`
Compute and assign new or new columns.

`df["weight"] = df["length"]*df["depth"]`
Add single column.

`df["label"] = df["x"].apply(lambda x: "label1" if x < 100 else "label2")`
fill column into 2 buckets.

pandas provides a large set of vector methods that operate on a Series (or Dataframe or a single selected column) in pandas. Examples: `df["weight"]` returns a Series of the column `weight`, or a single Series for the individual Series. Examples:

<code>min()</code> Minimum value max.	<code>minn()</code> Minimum value min.
<code>max()</code> Maximum value max.	<code>maxn()</code> Maximum value min.
<code>abs()</code> This value of abs thresholds.	<code>absvalue()</code> Absolute value.

The examples below can also be applied to Groups. In this case, the functions `df.groupby()` on a upper group-basis, and the returned outputs are of the length of the original Dataframe.

<code>shift()</code> Data with columns shifted by 1.	<code>shift(-1)</code> Data with values lagged by 1.
<code>rank("dense")</code> Ranks with no tie-break	<code>rank()</code> Cumulative rank
<code>rank("method")</code> Ranks using min-tie-break	<code>rank("min")</code> Cumulative min
<code>rank("pct")</code> Ranks based on percent rank	<code>rank("pct")</code> Cumulative pct
<code>rank("first")</code> Ranks. Ties go to first rank	<code>rank("first")</code> Cumulative first

Plotting

`df.plot.hist()`
Plot histogram for each column

`df.plot.scatter("x", "y")`
Plot scatter scatter using pairs of columns

Combine Data Sets

Standard Index

<code>df1</code>	<code>df2</code>	+	=
A	B		
B	C		
C	D		
D	E		

`pd.merge(df1, df2, how="left", on="x")`
Join matching rows on `df1` to `df2`.

`pd.merge(df1, df2, how="right", on="x")`
Join matching rows on `df2` to `df1`.

`pd.merge(df1, df2, how="inner", on="x")`
Join data. Retain only rows in both sets.

`pd.merge(df1, df2, how="outer", on="x")`
Join data. Retain values of rows in both sets.

Matching Index

<code>df1</code>	<code>df2</code>	=
A	B	
B	C	
C	D	

`df1[df1["x"].isin(df2["x"])]`
Rows in `df1` that have a match in `df2`.

`df1[df1["x"].isin(df2["x"])]`
Rows in `df1` that do not have a match in `df2`.

Index

<code>df1</code>	<code>df2</code>	=
A	B	
B	C	
C	D	

`pd.merge(df1, df2, how="left", on="x")`
Rows that appear in `df1` only and `df2` (Intersection).

`pd.merge(df1, df2, how="outer", on="x")`
Rows that appear in either `df1` or `df2` (Union).

`pd.merge(df1, df2, how="inner", on="x")`
Rows that appear in `df1` only (Setdiff).

`pd.merge(df1, df2, how="outer", on="x", indicator=True)`
Query, e.g. `df1["only"]`
`df1["only"]`
`df1["only"]`
Rows that appear in `df1` only (Setdiff).

HELP!



stack overflow



**CIENCIA
DE DATOS**

EANT

HELP!

seaborn

0.9.0

Gallery

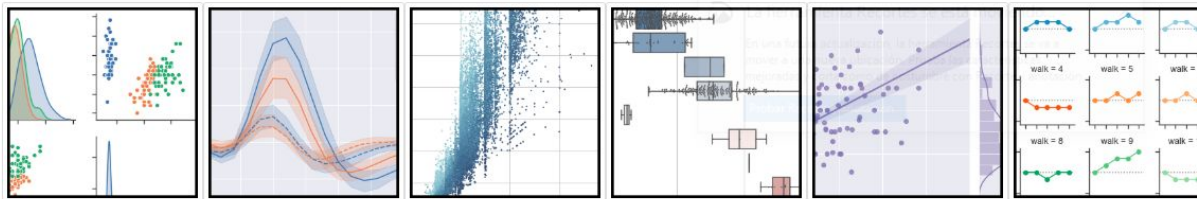
Tutorial

API

Site ▼

Page ▼

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on `matplotlib`. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the [introductory notes](#). Visit the [installation page](#) to see how you can download the package. You can browse the [example gallery](#) to see what you can do with seaborn, and then check out the [tutorial](#) and [API reference](#) to find out how.

To see the code or report a bug, please visit the [github repository](#). General support issues are most at home on [stackoverflow](#), where there is a seaborn tag.

Contents

- [Introduction](#)
- [Release notes](#)
- [Installing](#)
- [Example gallery](#)
- [Tutorial](#)
- [API reference](#)

Features

- Relational: [API](#) | [Tutorial](#)
- Categorical: [API](#) | [Tutorial](#)
- Distributions: [API](#) | [Tutorial](#)
- Regressions: [API](#) | [Tutorial](#)
- Multiples: [API](#) | [Tutorial](#)
- Style: [API](#) | [Tutorial](#)
- Color: [API](#) | [Tutorial](#)

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

EANT

ESCUELA ARGENTINA
DE NUEVAS TECNOLOGIAS

Estate atento a más eventos!



EANT



eanttech



@eanttech