



Descubre Python: La Llave Maestra para la Ciencia de Datos

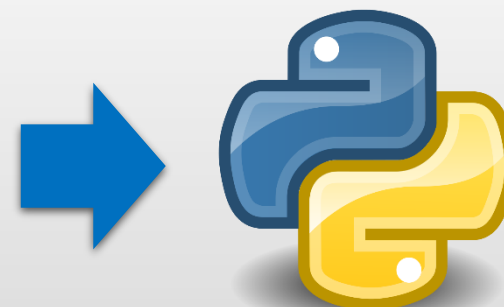
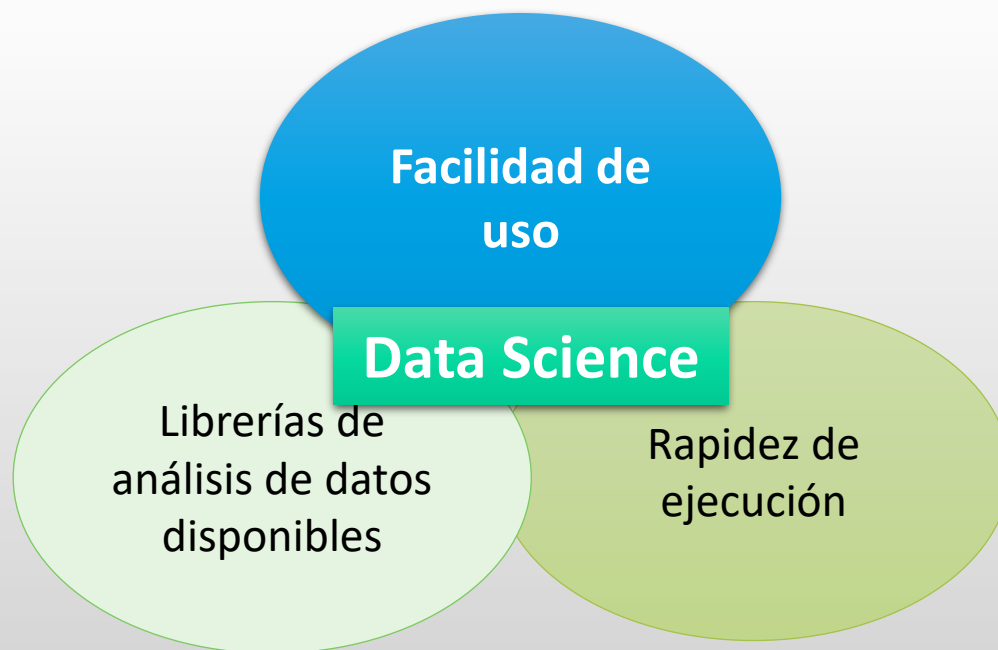
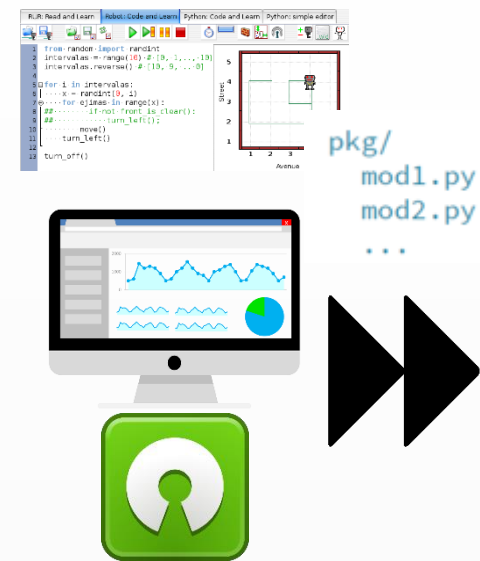
IVÁN PINAR DOMÍNGUEZ ®



BLOQUE 1: Introducción al Análisis de Datos con Python

¿Qué es Python y qué nos proporciona para el análisis de datos?

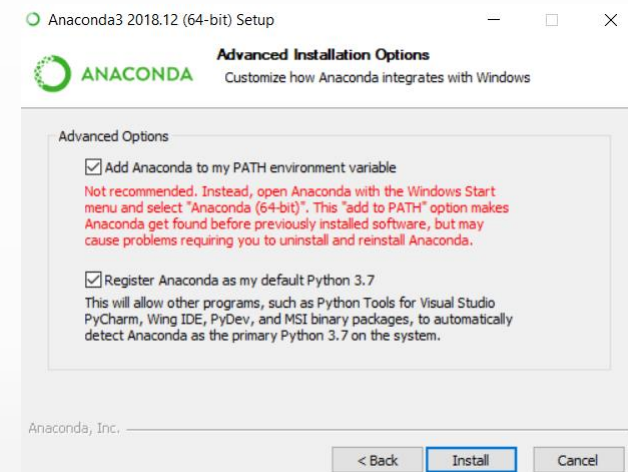
- Python es un lenguaje de programación orientado a objetos de **alto nivel** y de propósito general.
- Python se basa en la **simpleza** y en la facilidad de lectura de código disminuyendo costes de mantenimiento y **facilitando su aprendizaje**.
- Soporta módulos y librerías que favorecen la **reutilización de código**.
- Dispone de potentes estructuras de datos y **librerías enfocadas al análisis de datos**.
- Es un lenguaje **interpretado**, se ejecuta el código línea a línea. No hay un paso de compilación como en otros lenguajes → Rapidez de ejecución.
- Python es **open source y multiplataforma** (Windows, Linux, Mac,...).



Instalación Python + Jupyter

- **ANACONDA:** Distribución de Python con múltiples librerías pre-instaladas para Data Science.

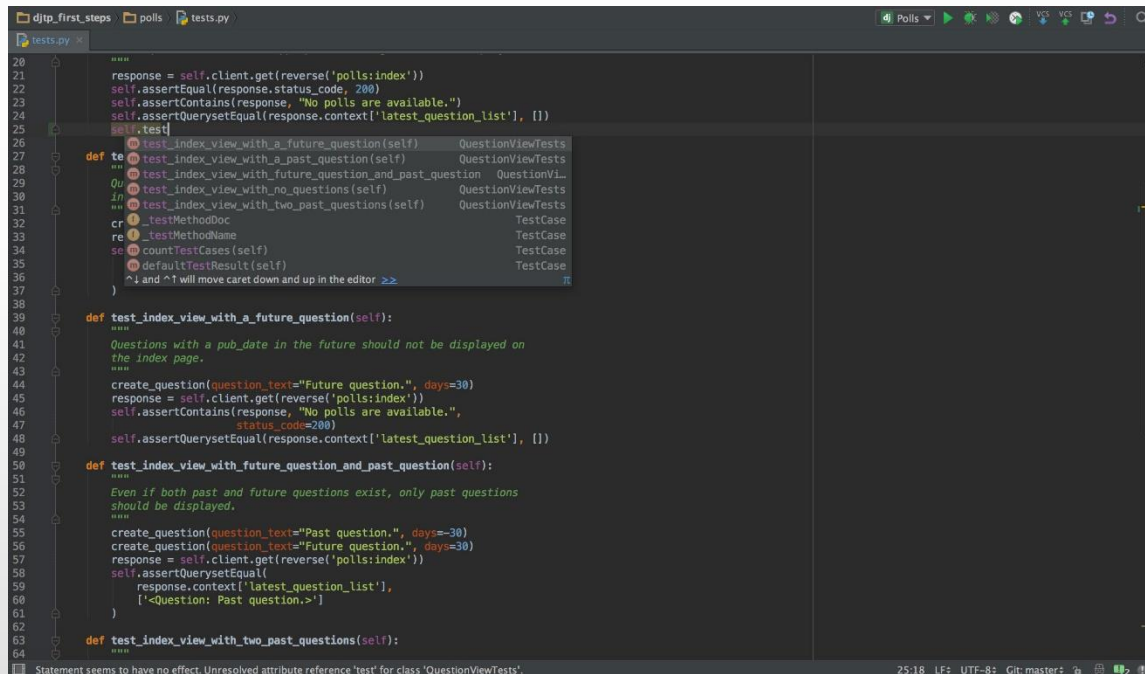
<https://www.anaconda.com/distribution/#download-section>



- **Definir el PATH:** Definir la ruta de la instalación (Python.exe / conda.exe) para mayor facilidad de uso a la hora de instalar cualquier nueva librería:
 - **Ruta habitual de Python:**
 - C:\Users\tu_nombre_de_usuario\Anaconda3\Scripts
 - **Ruta habitual de “conda”** (actualización de paquetes):
 - C:\Users\tu_nombre_de_usuario\Anaconda3\Scripts
- Panel de Control > system > advanced > | Variables de Entorno | > system variables
→ Añadir a la variable PATH la ruta concreta con “;”

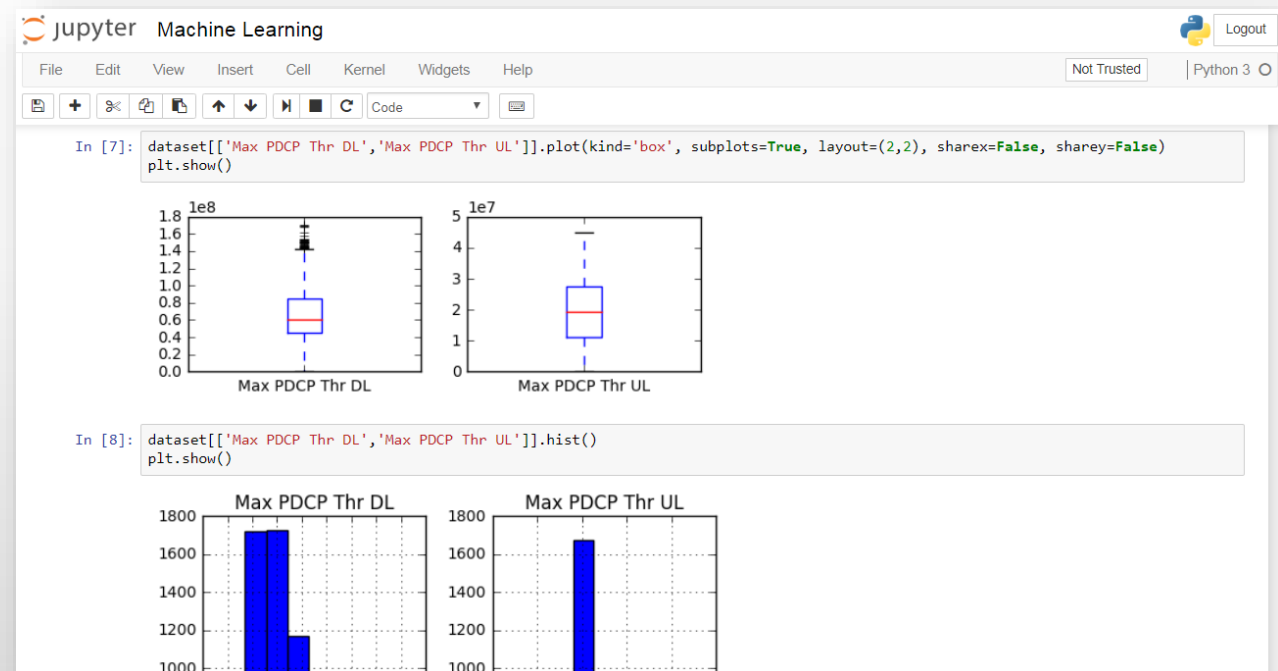
Instalación Python + Jupyter

IDE (Programación enfocada a scripts y desarrollos extensos)



```
20
21
22 response = self.client.get(reverse('polls:index'))
23 self.assertEqual(response.status_code, 200)
24 self.assertContains(response, "No polls are available.")
25 self.assertQuerysetEqual(response.context['latest_question_list'], [])
26
27 @test_index_view_with_a_future_question(self)
28 def test_index_view_with_a_past_question(self):
29     """
30     test_index_view_with_future_question_and_past_question
31     """
32     create_question(question_text="Future question.", days=30)
33     response = self.client.get(reverse('polls:index'))
34     self.assertContains(response, "No polls are available.",
35                         status_code=200)
36     self.assertQuerysetEqual(response.context['latest_question_list'], [])
37
38 @test_index_view_with_future_question_and_past_question(self)
39 def test_index_view_with_future_question_and_past_question(self):
40     """
41     Even if both past and future questions exist, only past questions
42     should be displayed.
43     """
44     create_question(question_text="Past question.", days=-30)
45     create_question(question_text="Future question.", days=30)
46     response = self.client.get(reverse('polls:index'))
47     self.assertQuerysetEqual(
48         response.context['latest_question_list'],
49         ['<Question: Past question.>']
50     )
51
52 @test_index_view_with_two_past_questions(self)
53 def test_index_view_with_two_past_questions(self):
54     """
55     """
56     create_question(question_text="Past question 1.", days=-30)
57     create_question(question_text="Past question 2.", days=-30)
58     response = self.client.get(reverse('polls:index'))
59     self.assertQuerysetEqual(
60         response.context['latest_question_list'],
61         ['<Question: Past question 1.>',
62          '<Question: Past question 2.>']
63     )
64
65 Statement seems to have no effect. Unresolved attribute reference 'test' for class 'QuestionViewTests'.
```

JUPYTER NOTEBOOK (Programación interactiva)



Importar librerías y fuentes de datos

- **Librerías en Python:**

- Librería = Directorio de Scripts en Python (cada script un modulo con funciones, métodos,..)
- Inmensa variedad de librerías ya definidas para el análisis de datos (Numpy, Pandas, Matplotlib, Scikit-learn,...)
- Para instalar librería: **conda install <librería>**
- Para importar un paquete:

```
pkg/  
mod1.py  
mod2.py  
...
```

```
In [1]: import numpy as np  
  
In [2]: array_ejemplo = np.array([1,2,3])  
  
In [3]: array_ejemplo  
Out[3]: array([1, 2, 3])
```

- **Ruta habitual librerías:**

- C:\Users\tu_nombre_de_usuario\Anaconda3\Lib\site-packages

Importar librerías y fuentes de datos

Librería Pandas

- Elemento clave: Dataframe

	Pais	Poblacion	Area
0	Mexico	129	1973.0
1	Espana	46	505.0
2	Venezuela	32	916.0

Tipo: Objeto int64 float64

- Convención: `import pandas as pd`



Importar de un CSV

```
df = pd.read_csv(r'file.csv', index_col = 0, nrows=5,
encoding = "ISO-8859-1", delimiter=';')
```

Ejercicio:

- Importar fichero csv con la información de población, esperanza de vida y renta per cápita de cada país:

	País	Poblacion	Renta per capita	Esperanza de vida
0	United States	325084756	59939	78,9
1	China	1421021791	8612	76,7
2	Japan	127502725	38214	84,5
3	Germany	82658409	44680	81,2
4	India	1338676785	1980	69,4

Visualización básica con Matplotlib

- Librería de visualización
- Todo tipo de gráficos con amplia configuración

- **Line Plot**

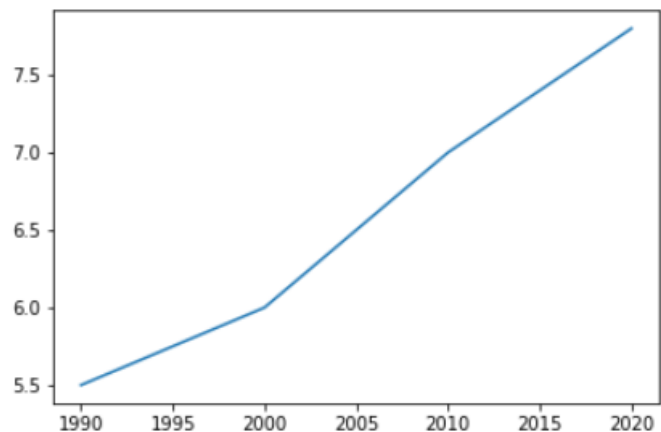
```
In [1]: import matplotlib.pyplot as plt
```

```
In [2]: año = [1990, 2000, 2010, 2020]  
pob = [5.5, 6, 7, 7.8]
```

```
In [3]: plt.plot(año, pob)
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x912b208>]
```

```
In [4]:
```



- **Scatter Plot**

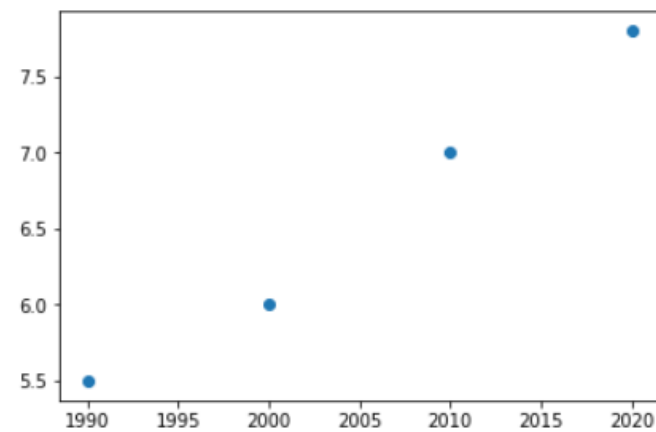
```
In [1]: import matplotlib.pyplot as plt
```

```
In [2]: año = [1990, 2000, 2010, 2020]  
pob = [5.5, 6, 7, 7.8]
```

```
In [5]: plt.scatter(año, pob)
```

```
Out[5]: <matplotlib.collections.PathCollection at 0x95d0a58>
```

```
In [6]:
```



Visualización básica con Matplotlib

- ¿Cómo visualizamos variables de un dataframe de Pandas?

```
In [34]: import pandas as pd  
df = pd.DataFrame(dic)
```

```
In [35]: df
```

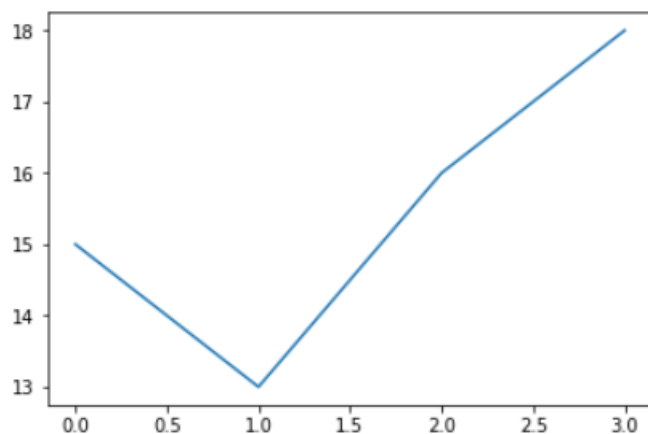
```
Out[35]:
```

	hum	temp
0	60	15
1	80	13
2	45	16
3	55	18

```
In [36]: plt.plot(df["temp"])
```

```
Out[36]: [matplotlib.lines.Line2D at 0xc0f2780]
```

```
In [37]: plt.show()
```

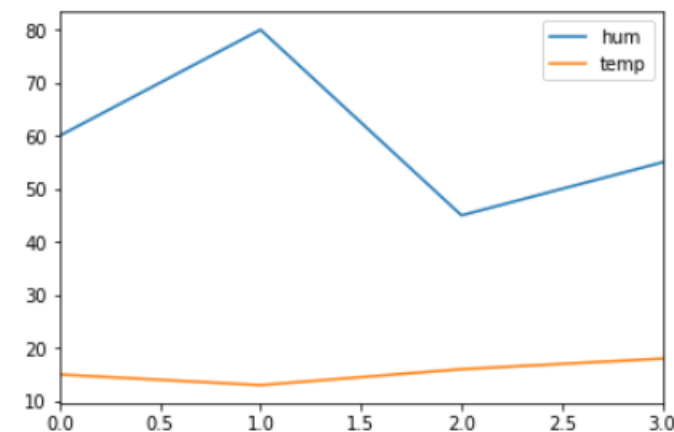


`Df["nombre_columna"].plot()`

```
In [38]: df.plot()
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0xc130438>
```

```
In [39]: plt.show()
```



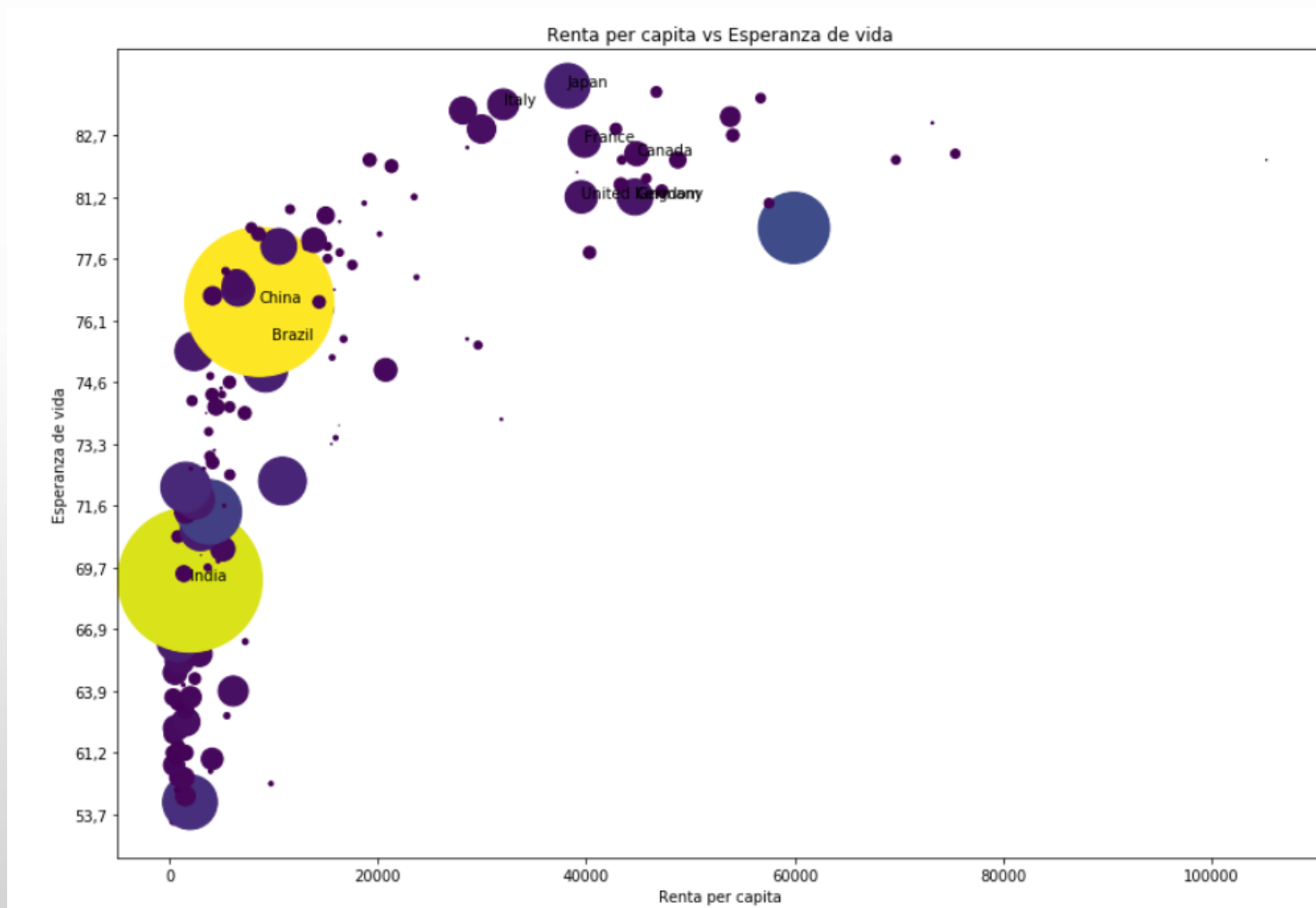
<https://matplotlib.org/gallery/index.html>



-
- The graph illustrates the exponential growth of Twitter activity over time. The x-axis, labeled 'Date', spans from 1927-12-30 to 2007-08-27. The y-axis represents the number of tweets per second, ranging from 0 to 3500. The data shows a period of low activity until approximately 1987, followed by a rapid increase. A notable peak occurs around 2007, reaching nearly 3500 tweets per second, before a sharp decline and subsequent recovery.

Visualización básica con Matplotlib – Caso práctico

- **Ejercicio:** Visualizar un gráfico de Esperanza de Vida frente a Renta per capita:
 - **¿Existe una correlación entre estas variables?**



Flujograma de un proyecto Data Science

1.Importación de datos

Obtener datos desde fuentes heterogéneas.



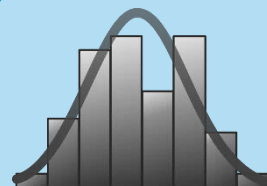
2.Limpieza de inconsistencias

Eliminar información errónea, redundante, transformación de tipos de datos.



3.Limpieza estadística y transformación

Resumen estadístico, búsqueda de outliers (boxplots) e interpolación de datos.

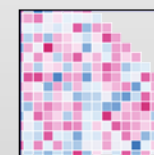


4.Visualización de datos

Mostrar la distribución de nuestra información con diferentes visualizaciones como histogramas, CDFs, correlaciones, series temporales,...



matplotlib



Seaborn



5.Análisis y conclusiones

Análisis de datos y generación de conclusiones para la toma de decisiones.

