

Conceptos Básicos de Inteligencia Artificial y Aprendizaje Automático para la Investigación

**Clase 1: Introducción, Análisis y
Visualización de datos con Python**

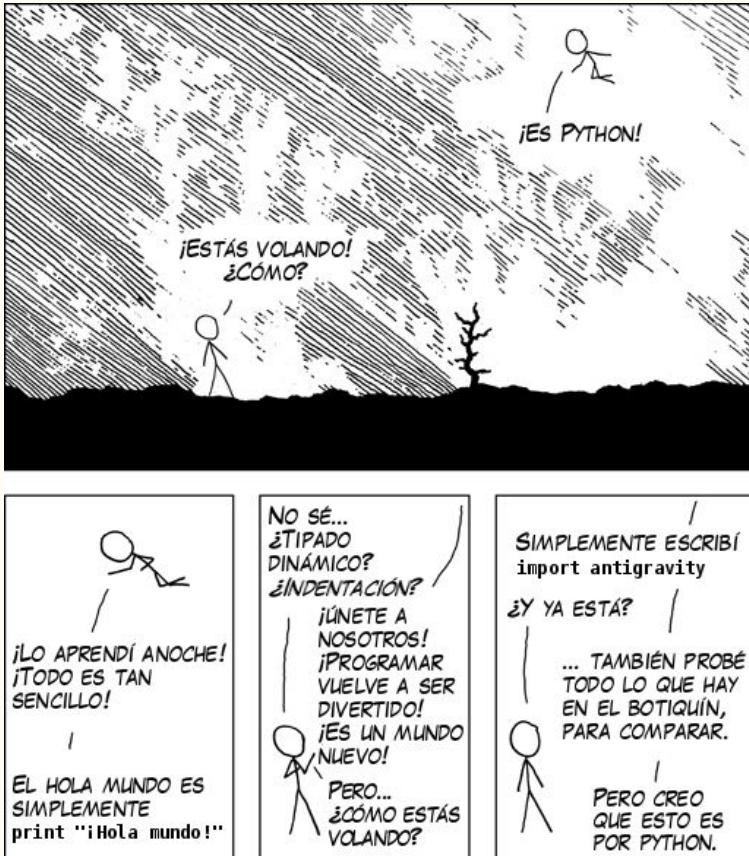
Dr. Juan Claudio Toledo Roy
Instituto de Ciencias Nucleares UNAM

¿Qué veremos en esta clase?

- Python como lenguaje de análisis y visualización de datos
- Librerías principales de Python para análisis de datos
- Análisis exploratorio de datos (EDA)
- Lectura y limpieza de datos
- Visualización de datos

Datos y ejemplos: https://github.com/meithan/Curso_IA_Investigacion

Generalidades de Python



Generalidades de Python

- Creado por Guido van Rossum en 1991 (¡anterior a Java!)
- Versión 2.0 en 2000, versión 3.0 en 2008; última: 3.13 (2024)
- Versión 3 no compatible hacia atrás; ¡no usar Python 2!
- Nombre inspirado en el grupo de comediantes británicos Monty Python



Generalidades de Python

Python es el lenguaje más usado actualmente.



Jan 2026	Jan 2025	Change	Programming Language	Ratings	Change
1	1		 Python	22.61%	-0.68%
2	4		 C	10.99%	+2.13%
3	3		 Java	8.71%	-1.44%
4	2		 C++	8.67%	-1.62%
5	5		 C#	7.39%	+2.94%
6	6		 JavaScript	3.03%	-1.17%
7	9		 Visual Basic	2.41%	+0.04%
8	8		 SQL	2.27%	-0.14%
9	11		 Delphi/Object Pascal	1.98%	+0.19%
10	18		 R	1.82%	+0.81%

Generalidades de Python

Características principales:

- Lenguaje de (muy) **alto nivel**
- **Dinámicamente tipado, bytecode interpretado**
- Sintaxis **ligera, intuitiva y poderosa**
- **Multi-paradigma:** procedural, orientado a objetos, funcional, etc.
- **Extensa librería estándar** (estructuras de datos, matemáticas, tipos de datos, etc)
- **Multi-plataforma** y tremadamente ubicuo
- **Ecosistema de terceros muy amplio**

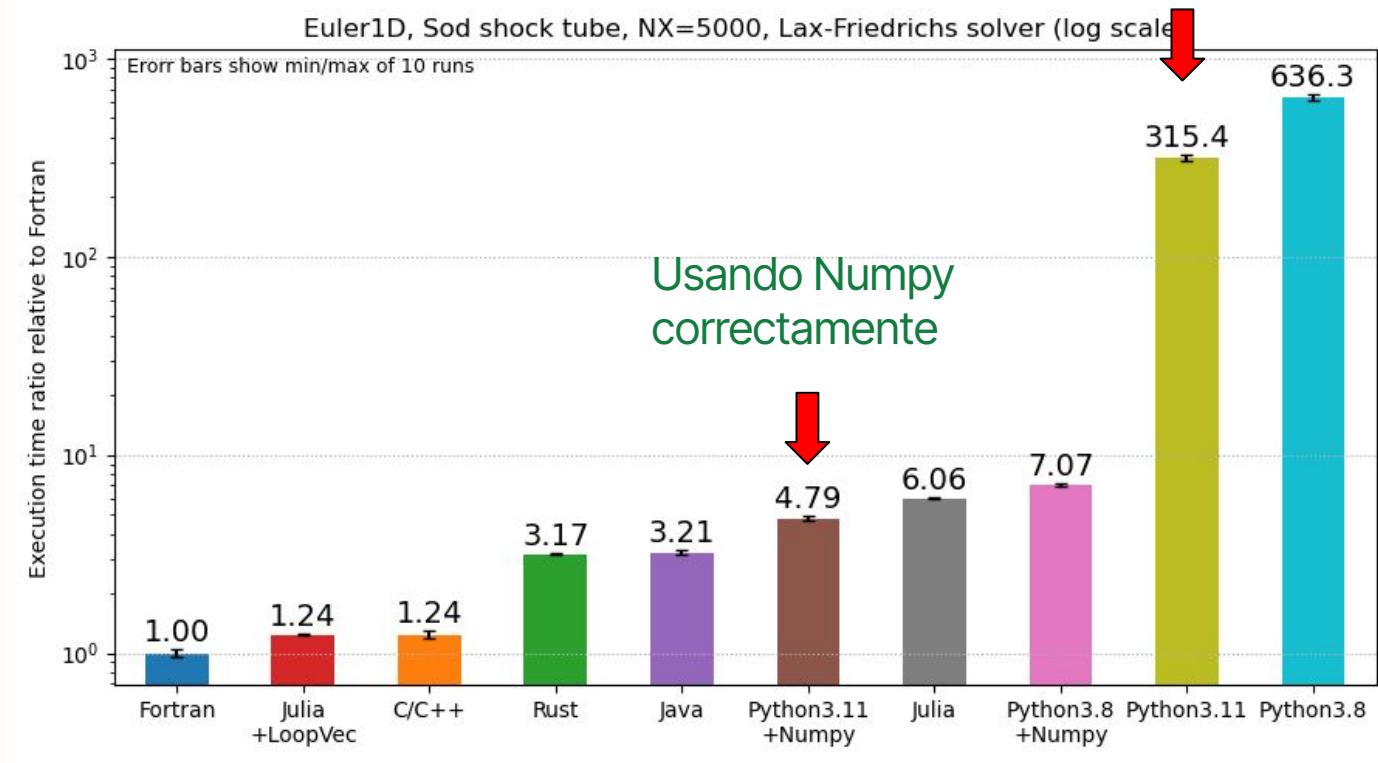
Generalidades de Python

Pero ...

- **Lento** en comparación de otros lenguajes, sobre todo compilados
 - Compilado a bytecode (aunque Java también y es bastante más rápido)
 - Tipado dinámico y “duck typing”
 - Todo es un objeto
- Está diseñado para ser **sintéticamente poderoso y ligero**, no para ser rápido
- Hay formas de acelerarlo, sobre todo vía librerías como **Numpy**
- Existen algunos **compiladores** avanzados (PyPy, Numba)

Generalidades de Python

Python nativo
¡Ouch!



Generalidades de Python

The Zen of Python

- Ejecutar `import this`

Bello es mejor que feo.
Explícito es mejor que implícito.
Simple es mejor que complejo.
Complejo es mejor que complicado.
Plano es mejor que anidado.
Espaciado es mejor que denso.
La legibilidad es importante.
...

El estilo “Pythónico” de programación.



```
for i in range(len(array)):  
    print(array[i])
```



```
for item in array:  
    print(item)
```

Instalación y uso de Python

Instalación

- Directamente desde la página oficial: <https://www.python.org/downloads/>
- Vía una distribución como Anaconda: <https://www.anaconda.com/download>

Uso

- Editor de código ([Notepad++](#), [Atom](#), [VS Code](#)) + terminal
- IDE, por ejemplo [PyCharm](#)
- IDE en web: [Jupyter Notebook](#), [Google Colab](#)



Python como lenguaje de análisis de datos

Capacidades para análisis y visualización de datos

- Python en sí tiene **pocas** herramientas de análisis/visualización de datos
- Pero amplio y muy activo ecosistema de **librerías de terceros**:
 - **NumPy**: cómputo numérico y herramientas matemáticas
 - **SciPy**: cómputo científico y herramientas avanzadas
 - **Matplotlib**: graficación y visualización de datos
 - **Seaborn, plotly**: graficación interactiva
 - **pandas**: especializada en análisis de datos
 - **scikit-learn**: machine learning
 - **NetworkX**: análisis de redes
 - **PyTorch, TensorFlow**: deep learning



Python como lenguaje de análisis de datos

- La gran mayoría de estas librerías de terceros están en **PyPI** (Python Package Index) y se pueden instalar muy fácilmente usando **pip**

A screenshot of the PyPI (Python Package Index) website showing the details for the matplotlib package. The page has a blue header with the PyPI logo, a search bar, and navigation links for Help, Docs, Sponsors, Log in, and Register. The main title is "matplotlib 3.10.5". Below it, there's a green button with a checkmark and the text "Latest version". To the right, it says "Released: Jul 31, 2025". A red arrow points to a red-bordered button containing the command "pip install matplotlib" with a copy icon. The page also features a large "matplotlib" logo at the bottom.

matplotlib 3.10.5

Type '!' to search projects

Help Docs Sponsors Log in Register

✓ Latest version

Released: Jul 31, 2025

pip install matplotlib ⚡

Python plotting package

Navigation

Project description

pypi v3.10.5 downloads inaccessible powered by NumFOCUS

help forum discourse chat on gitter issue tracking github PR Welcome

Tests passing Azure Pipelines succeeded build unknown codecov 88% version scheme EffVer

Verified details ✓ These details have been *verified by PyPI*

Project links Bug Tracker

matplotlib

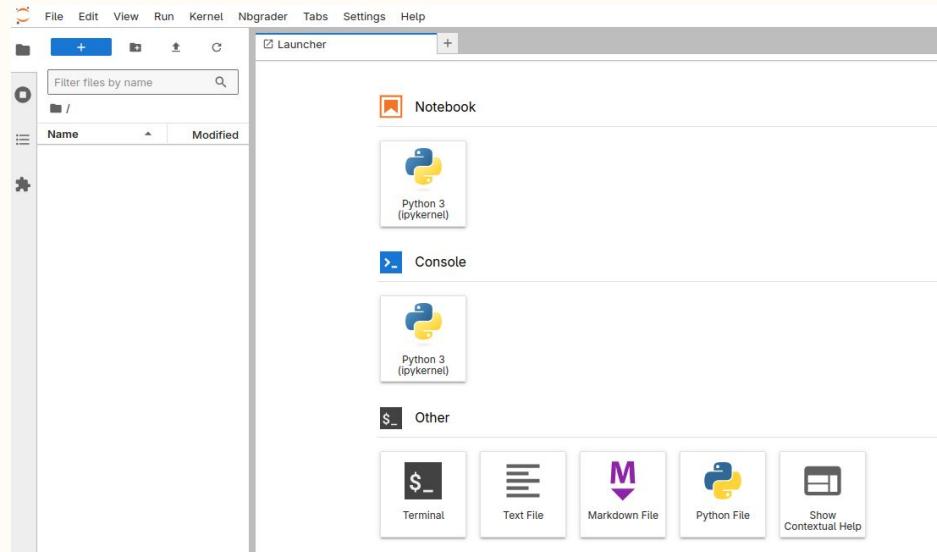
Python como lenguaje de análisis de datos

- Hoy en día, lo recomendado oficialmente es usar **virtual environments (venvs)**
 - Ambiente aislado
(independiente del sistema)
con paquetes específicos
 - Control fino de versión de
paquetes
 - Entorno reproducible
(se puede copiar el venv)
 - No requiere privilegios
elevados (administrador)
1. Crear venv nuevo (en cualquier carpeta del usuario)
`> python -m venv test`
 2. Activar el venv en la sesión actual (en terminal)
`> source test/bin/activate`
La terminal usualmente indicará el venv activo:
`test>`
 3. Instalar paquetes con pip
`test> pip install matplotlib`
 4. Se puede desactivar el venv:
`test> deactivate`

Python como lenguaje de análisis de datos

- Alternativamente, se pueden usar entornos de Python en web como **Google Colab** y **JupyterLab** (pero dependen de un servicio ofrecido por terceros)
- Para las demostraciones del curso, usaremos **MACTI**, una plataforma en línea para enseñanza en cómputo del ICN que incluye JupyterLab

<https://www.macti.unam.mx>



Análisis Exploratorio de Datos (EDA)

- **Explorar** un conjunto de **datos** para entender su **estructura, características** y potenciales **problemas** antes de aplicar métodos de análisis más formales

1. Aprender la estructura de los datos

- ¿Qué variables están presentes?
- ¿De qué tipo son (numéricas, categóricas, temporales)?
- ¿Cuántos datos (observaciones) hay?

2. Valorar la calidad de los datos

- Valores faltantes
- Valores duplicados
- Valores inconsistentes o imposibles
- Valores atípicos (outliers)

(continúa)

Análisis Exploratorio de Datos (EDA)

3. Visualizar características principales

- Tendencia central (media, mediana, etc)
- Variabilidad (desviación estándar, IQR, etc)
- Distribución (sesgos, centralidad, etc)

4. Identificar (visualmente) patrones y relaciones

- Correlaciones entre variables
- Diferencias entre grupos
- Tendencias y periodicidades temporales

5. Guía para análisis subsecuente

- ¿Qué herramientas de análisis se pueden usar?
- ¿Cuáles modelos o pruebas se pueden aplicar?

Regla de oro: antes de aplicar cualquier análisis
¡VISUALIZAR LOS DATOS!

Tipos de archivos de datos

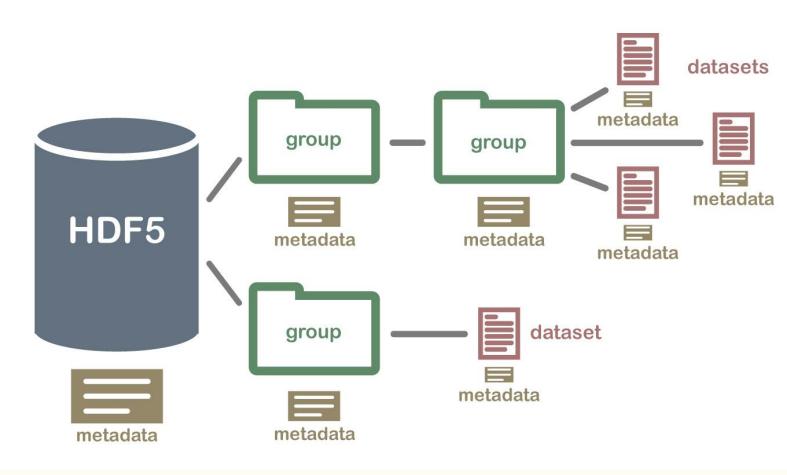
- **Texto plano:**

- Legible por humanos
- Simple, transparente, portable
- Poco eficiente (espacio)
- Necesario saber cómo interpretarlo
- Formatos comunes:
txt, csv, tsv
- También formatos de texto estructurado: JSON, XML, etc.

```
# Data from March 1958 through April 1974 have been obtained  
# of the Scripps Institution of Oceanography (SIO) and were ob-  
# tained from the Scripps website (scrippsc02.ucsd.edu).  
# Monthly mean CO2 constructed from daily mean values.  
# Scripps data downloaded from http://scrippsc02.ucsd.edu/datasets/co2/monthly/.  
# Monthly values are corrected to center of month based on av-  
# erage cycle. Missing days can be asymmetric which would produce a  
# bias. Missing months have been interpolated, for NOAA data indica-  
# ting uncertainty. We have no information for SIO data about un-  
# certainty so that they are also indicated by negative numbers  
#  
# NOTE: Due to the eruption of the Mauna Loa Volcano, measurements  
# were suspended as of Nov. 29, 2022 and resumed in July 2023.  
# Observations starting from December 2022 to July 4, 2023 are  
# taken at the Maunakea Observatories, approximately 21 miles north of the  
# volcano.  
year,month,decimal date,average,deseasonalized,nDays,sdev,unc  
1958,3,1958.2027,315.71,314.44,-1,-9.99,-0.99  
1958,4,1958.2877,317.45,315.16,-1,-9.99,-0.99  
1958,5,1958.3699,317.51,314.69,-1,-9.99,-0.99  
1958,6,1958.4548,317.27,315.15,-1,-9.99,-0.99  
1958,7,1958.5370,315.87,315.20,-1,-9.99,-0.99  
1958,8,1958.6219,314.93,316.21,-1,-9.99,-0.99  
1958,9,1958.7068,313.21,316.11,-1,-9.99,-0.99  
1958,10,1958.7890,312.42,315.41,-1,-9.99,-0.99  
1958,11,1958.8710,310.55,315.21,-1,-9.99,-0.99
```

Tipos de archivos de datos

- Formatos **binarios**:
 - No legible por humanos, sí por máquinas
 - Compacto (espacio), rápido de leer en la computadora
 - Necesario saber cómo interpretarlo, pero suelen contener metadata
 - Usualmente hay librerías para leerlos
 - Ejemplos: formatos estándar científicos (HDF5, NetCDF, FITS, MAT), formatos binarios ad hoc
 - Otros:
 - Hojas de cálculo (Excel)
 - Bases de datos (MySQL, SQLite)



Lectura de datos en Python

- Usando funciones básicas de Python (archivos de texto, CSVs, etc)

```
datos = []
with open("sunspots.dat") as f:
    line = f.readline()
    for line in f:
        x, y = [float(d) for d in line.split()]
        datos.append((x, y))
```

Sin argumentos, `split()` separa un string usando *whitespace* (espacios, tabulaciones, saltos de línea), uno o más.

También se le puede pasar un string como separador, por ejemplo `split(", ")`

sunspots.dat

1850.001	100
1850.004	133
1850.007	85
1850.010	114
1850.012	52
1850.015	57
1850.018	107
1850.021	75
1850.023	40
1850.026	50
1850.029	65
1850.031	85
1850.034	99
1850.037	77
1850.040	73
1850.042	34
1850.045	74
1850.048	25

Lectura de datos en Python

- Usando la librería nativa **csv** de Python (CSVs, TSVs)

```
import csv
datos = []
with open("CO2.csv") as f:
    reader = csv.reader(f)
    for row in reader:
        mes = float(row[0])
        CO2 = float(row[1])
        datos.append((mes, CO2))
```

¿En qué difiere de leerlo "a mano"?

Que **csv.reader** va a parsear correctamente texto que contiene comas (si es un CSV correcto)

CO2.csv

Year	CO2 (ppm)
1958.208,	315.71
1958.292,	317.45
1958.375,	317.50
1958.458,	317.10
1958.542,	315.86
1958.625,	314.93
1958.708,	313.20
1958.792,	312.66
1958.875,	313.33
1958.958,	314.67
1959.042,	315.62
1959.125,	316.38
1959.208,	316.71
1959.292,	317.72
1959.375,	318.29
1959.458,	318.15

Lectura de datos en Python

- Usando **Numpy** (datos en ASCII o binario, CSVs, etc)

```
import numpy as np
datos = np.loadtxt("CO2.csv", skiprows=1, delimiter=",")
print(datos)
print(datos.shape)
type(datos)
```

- Devuelve un array de Numpy (más poderosos y eficientes que las listas de Python)
- Convierte automáticamente a números (floats o ints)
- El archivo debe contener sólo números, y mismo número de valores por fila
- Se pueden saltar líneas de encabezado y comentarios
- Se puede cambiar el **delimiter** de las columnas (default es espacio)

Lectura de datos en Python

- Usando **pandas** (múltiples formatos)

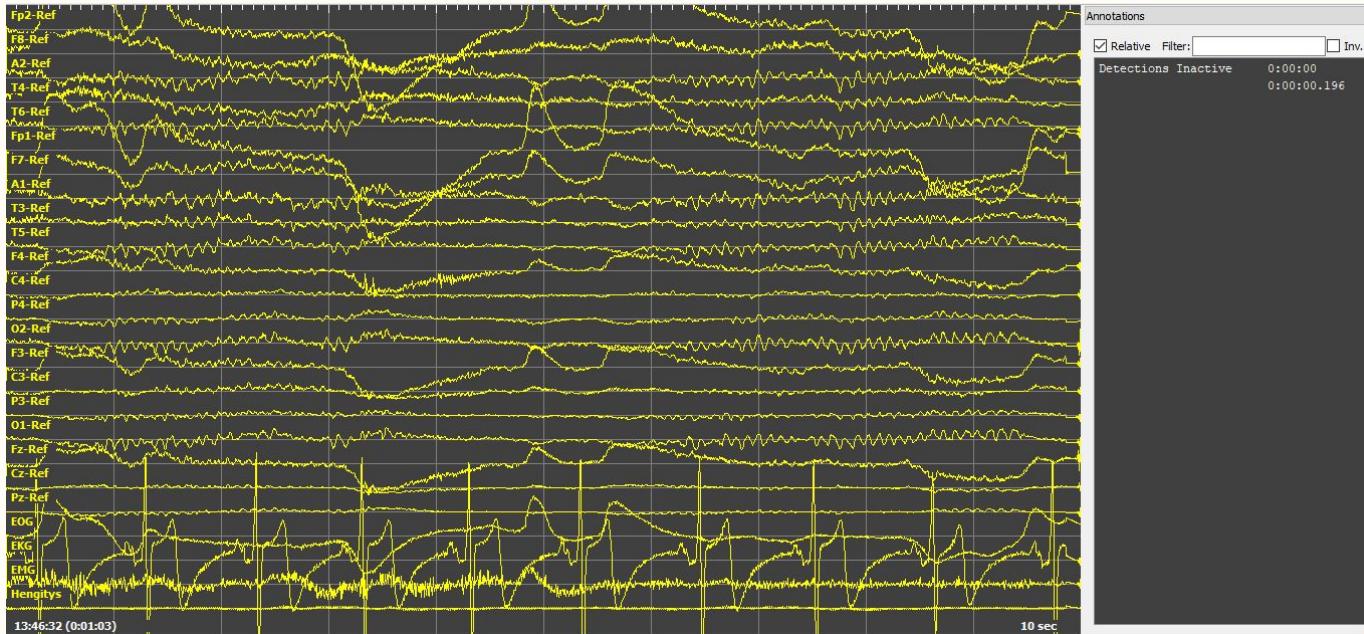
```
import pandas as pd
df = pd.read_csv("CO2.csv")
print(df)
print(type(df))
print(df.columns)
print(df["CO2 (ppm)"])
```

- Reconoce automáticamente encabezados, se brinca comentarios, etc.
- Devuelve un DataFrame de pandas (objeto de datos sofisticado)
- Columnas pueden ser de tipos de datos mixtos (p.ej. fechas formateadas)
- Un poco más opaco, hay que aprender a usarlo
- Un poco más pesado y lento

Lectura de datos en Python

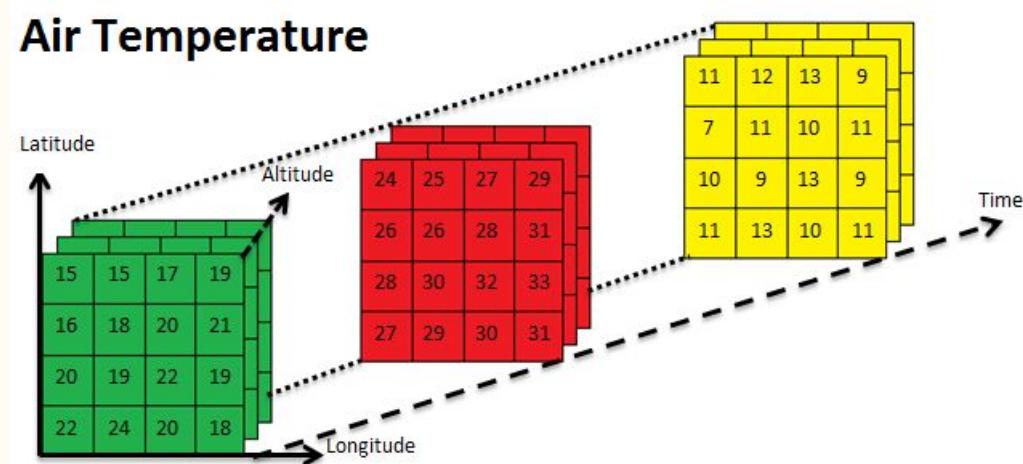
- Formatos especializados/propietarios: usualmente hay librerías de terceros en Python

Ejemplo: formato **EDF** (series de tiempo médicas), **PyEDFlib**



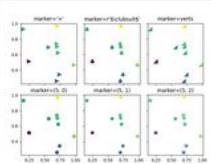
Lectura de datos en Python

- Formatos especializados: usualmente hay librerías de terceros en Python
- Ejemplo: formato **NetCDF** (series de tiempo y datos geoespaciales), **netCDF4**

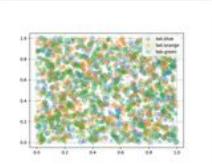


Visualización en Python con matplotlib

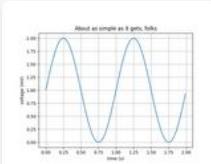
- **matplotlib** es una poderosa librería de graficación y visualización
- Muchos tipos de gráficos: líneas, scatters, barras, heatmaps, vectores, 3D, etc.
- Se usa mucho en conjunto con Numpy (de hecho, es una dependencia)



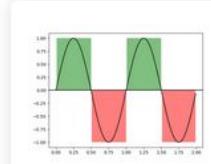
Marker examples



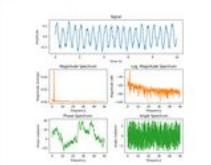
Scatter plot with a legend



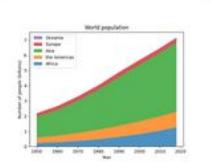
Line plot



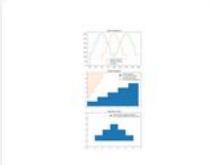
Shade regions defined by a logical mask using fill_between



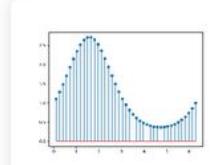
Spectrum representations



Stackplots and streamgraphs



Stairs Demo



Stem plot

<https://matplotlib.org>

Visualización en Python con matplotlib



Cheat sheet
Version 3.7.4

Quick start

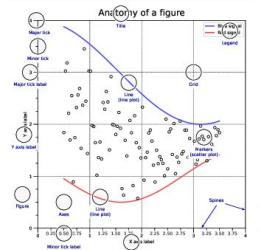
```
import numpy as np
import matplotlib as plt
import matplotlib.pyplot as plt

X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
plt.show()
```

Anatomy of a figure



Subplots layout

```
subplots([rows, cols, ...])
fig, axes = plt.subplots(3, 3)

G = gridspec(rows,cols, ...)
ax = G[0, :]

ax.inset_axes(extent)
d = make_axes_locatable(ax)
ax = d.new_horizontal('18%')
```

Getting help

matplotlib.org
github.com/matplotlib/matplotlib/issues
https://github.com/jakevdp/mpl_tutorial
stackoverlow.com/questions/tagged/matplotlib
https://gitter.im/matplotlib/matplotlib
twitter.com/matplotlib
Matplotlib users mailing list

Basic plots

```
plot([x], y, [fmt, ...]
X, Y, fmt, color, marker, linestyle)
```

```
scatter(X, Y, ...)
```

```
bar[h](x, height, ...)
```

```
x, height, width, bottom, align, color
```

```
imshow(Z, ...)
```

```
X, Y, Z, cmap, interpolation, extent, origin
```

```
contour(f)([x], [y], Z, ...)
```

```
X, Y, Z, levels, colors, extent, origin
```

```
pcolormesh([x], [y], Z, ...)
```

```
X, Y, Z, vmin, vmax, cmap
```

```
quiver([x], [y], U, V, ...)
```

```
X, Y, U, V, C, units, angles
```

```
pie(x, ...)
```

```
Z, explode, labels, colors, radius
```

```
text(x, y, text, ...)
```

```
x, y, text, va, ha, size, weight, transform
```

```
fill([between])(x) ...
```

```
X, Y1, Y2, color, where
```

Advanced plots

```
step(x, y, [fmt, ...]
```

```
X, Y, fmt, color, marker, where
```

```
boxplot(x, ...)
```

```
X, notch, sym, bootstrap, widths
```

```
errorbar(X,y,err,yerr, ...)
```

```
X, Y, yerr, yerr, fmt
```

```
hist(x, bins, ...)
```

```
X, bins, range, density, weights
```

```
violinplot(D, ...)
```

```
D, positions, widths, vert
```

```
barbs(X, Y, U, V, ...)
```

```
X, Y, U, V, C, length, pivot, sizes
```

```
eventplot(positions, ...)
```

```
positions, orientation, lineoffsets
```

```
hexbin(X, Y, C, ...)
```

```
X, Y, C, gridsize, bins
```

Scales

```
ax.set_[xy]scale(scale, ...)
```

```
linear any values
```

```
log values > 0
```

```
symlog any values
```

```
logit 0 < values < 1
```

Projections

```
subplot(..., projection=p)
```

```
p='polar'
```

```
p='3d'
```

```
p=ccrs.Orthographic()
```

```
import cartopy.crs as ccrs
```

```
Figure
```

Tick locators

```
from matplotlib import ticker
ax.[x|y]axis.set_[minor|major].locator(locator)
```

```
ticker.NullLocator()
```

```
ticker.MultipleLocator(base=0.5)
```

```
ticker.FixedLocator([0, 1, 5])
```

```
ticker.LinearLocator(numticks=3)
```

```
ticker.IndexLocator(base=0.5, offset=-0.25)
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

```
ticker.LogLocator(base=10, numticks=15)
```

```
ticker.AutoLocator()
```

```
ticker.MaxNLocator(n=4)
```

Animation

```
import matplotlib.animation as mpl_a
```

```
T = np.linspace(0, 2*np.pi, 100)
```

```
S = np.sin(T)
```

```
line, = plt.plot(T, S)
```

```
def animate(i):
```

```
    line.set_data(np.sin(T+i/50))
```

```
anim = plt.FuncAnimation(
```

```
    plt.gcf(), animate, interval=5)
```

```
plt.show()
```

Styles

```
plt.style.use(style)
```

```
default
```

```
classic
```

```
grayscale
```

```
ggplot
```

```
seaborn-v0_8
```

```
fast
```

```
bmh
```

```
Solarize_Light2
```

```
seaborn-v0_8-notebook
```

Quick reminder

```
ax.grid()
```

```
ax.set_xlim(min, max)
```

```
ax.set_xlabel(label)
```

```
ax.set_xticks(ticks, [labels])
```

```
ax.set_xticklabels(labels)
```

```
ax.set_title(title)
```

```
ax.tick_params(width=10, ...)
```

```
ax.set_axis([on|off])
```

```
fig.suptitle(title)
```

```
fig.tight_layout()
```

```
plt.gcf(), plt.gca()
```

```
mpl.rcParams['axes.linewidth']=1, ...
```

```
[fig].patch.set_alpha(alpha)
```

```
text=r'$\frac{e^{\pi i}}{\pi} \sin(\frac{e^{\pi i}}{\pi})$'
```

Keyboard shortcuts

```
ctrl+s Save
```

```
ctrl+w Close plot
```

```
f Fullscreen 0/1
```

```
b View back
```

```
p Pan view
```

```
z Zoom to rect
```

```
x Pan/zoom
```

```
y Pan/zoom
```

```
g Minor grid 0/1
```

```
c Major grid 0/1
```

```
l X axis log/linear
```

```
r Y axis log/linear
```

Ten simple rules

```
1. Know your audience
```

```
2. Identify your message
```

```
3. Adapt the figure
```

```
4. Captions are not optional
```

```
5. Do not trust the defaults
```

```
6. Be consistent or be clever
```

```
7. Do not assume the reader
```

```
8. Avoid "chartjunk"
```

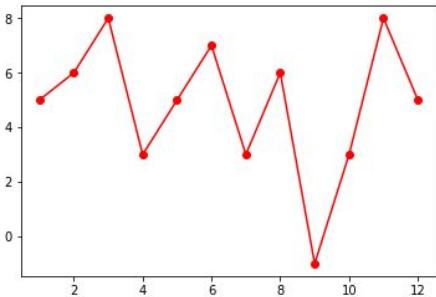
```
9. Message trumps beauty
```

```
10. Get the right numbers
```

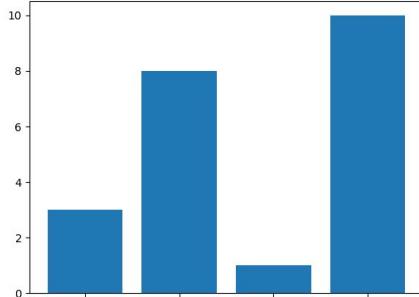
<https://matplotlib.org/cheatsheets/>

Visualización en Python con matplotlib

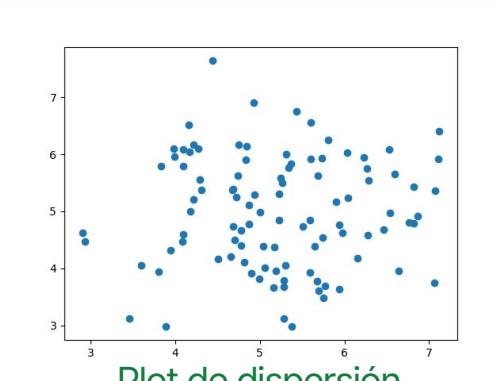
- Los 6 tipos de plots más frecuentemente usados:



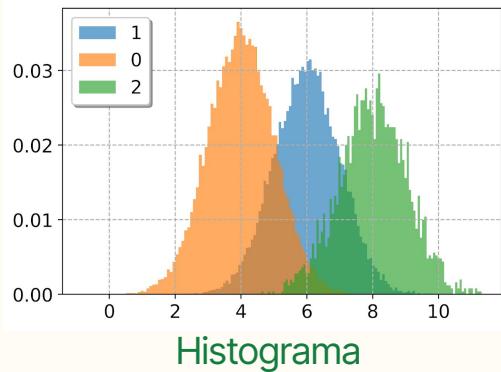
Plot de línea



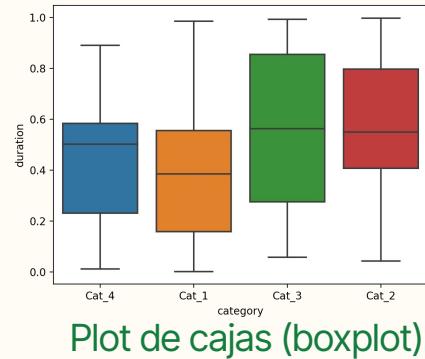
Plot de barras



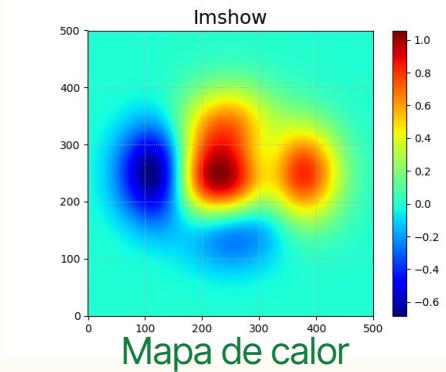
Plot de dispersión
(scatter plot)



Histograma



Plot de cajas (boxplot)



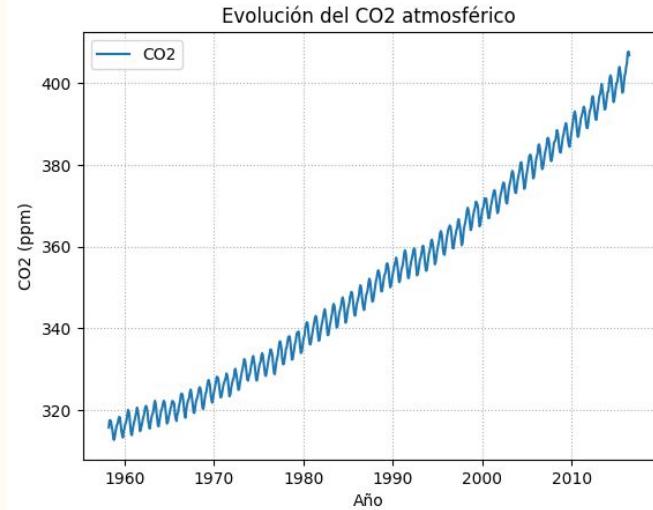
Mapa de calor

Visualización en Python con matplotlib

- `plt.plot()`: gráficas de líneas (i.e. datos ordenados) con marcadores opcionales

matplotlib.ipynb

```
plt.plot(tiempo, CO2, label="CO2")
plt.xlabel("Año")
plt.ylabel("CO2 (ppm)")
plt.title("Evolución del CO2 atmosférico")
plt.grid(ls=":")
plt.legend()
# plt.ylim(0, 450)
plt.show()
```



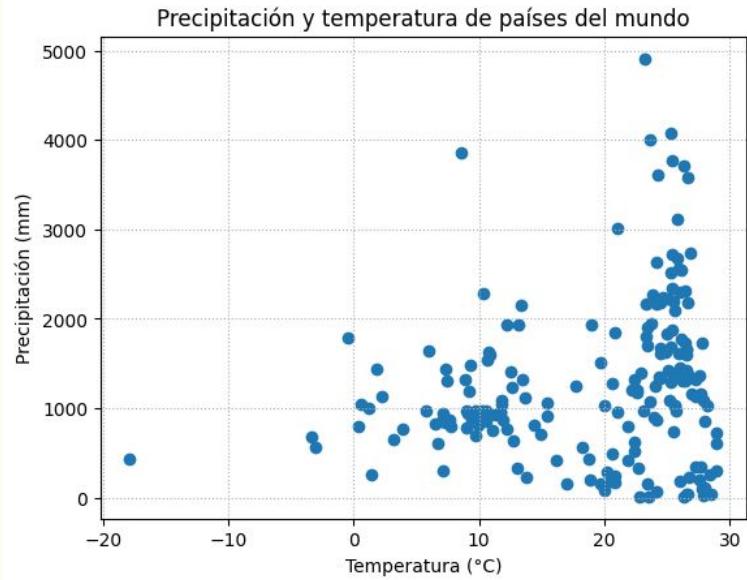
<https://gml.noaa.gov/ccgg/data/>

Visualización en Python con matplotlib

- `plt.scatter()`: scatter plots (parejas de datos no-ordenados)

matplotlib.ipynb

```
df = pd.read_csv("temp_precip.csv")
plt.scatter(df["Temp"], df["Precip"])
plt.xlabel("Temperatura (°C)")
plt.ylabel("Precipitación (mm)")
plt.title("Precipitación vs
temperatura para países del mundo")
plt.grid(ls":")
plt.tight_layout()
plt.show()
```



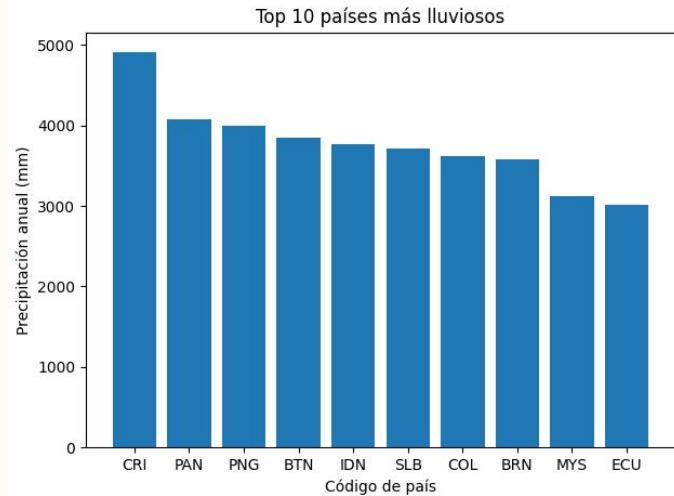
Datos de <https://ourworldindata.org/>

Visualización en Python con matplotlib

- `plt.bar()`: plots de barras

matplotlib.ipynb

```
df = pd.read_csv("temp_precip.csv")
top10_mas_lluviosos =
df.sort_values(by="Precip",
ascending=False).head(10)
plt.bar(top10_mas_lluviosos["Code"],
top10_mas_lluviosos["Precip"])
plt.xlabel("Código de país")
plt.ylabel("Precipitación anual (mm)")
plt.title("Top 10 países más lluviosos")
plt.tight_layout()
plt.show()
```



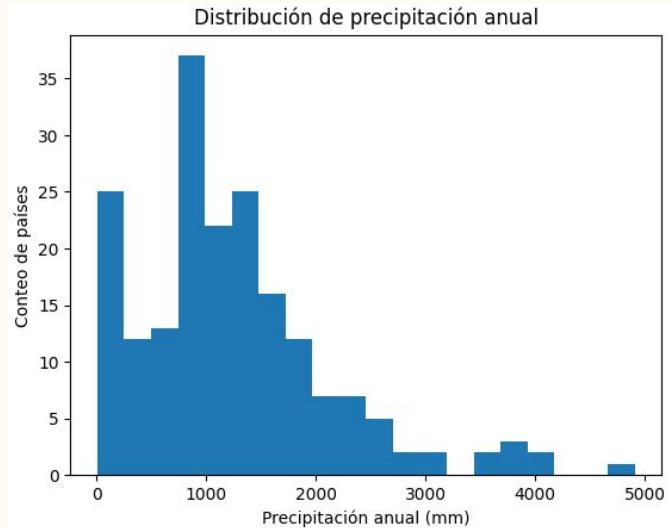
Datos de <https://ourworldindata.org/>

Visualización en Python con matplotlib

- `plt.hist()`: histogramas

matplotlib.ipynb

```
plt.hist(df["Precip"], bins=20)
# plt.hist(df["Precip"], bins=20,
histtype="step")
plt.xlabel("Precipitación anual (mm)")
plt.ylabel("Conteo de países")
plt.title("Distribución de precipitación
anual")
plt.show()
```



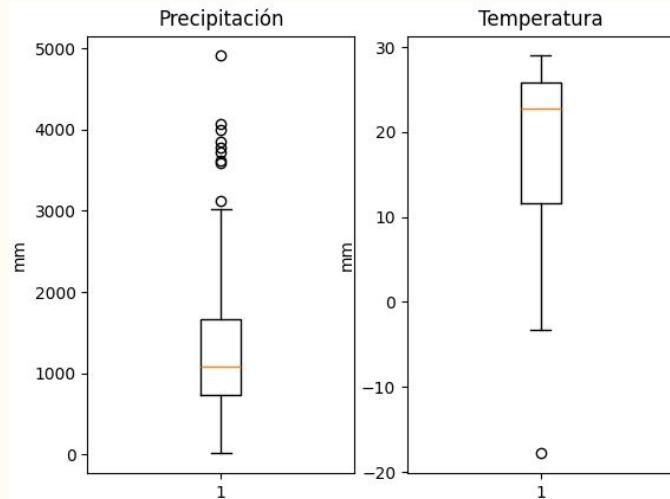
Datos de <https://ourworldindata.org/>

Visualización en Python con matplotlib

- `plt.boxplot()`: plots de cajas (boxplots)

matplotlib.ipynb

```
plt.figure()
plt.subplot(1, 2, 1)
plt.boxplot(df["Precip"])
plt.title("Precipitación")
plt.ylabel("mm")
plt.subplot(1, 2, 2)
plt.boxplot(df["Temp"])
plt.title("Temperatura")
plt.ylabel("mm")
plt.show()
```



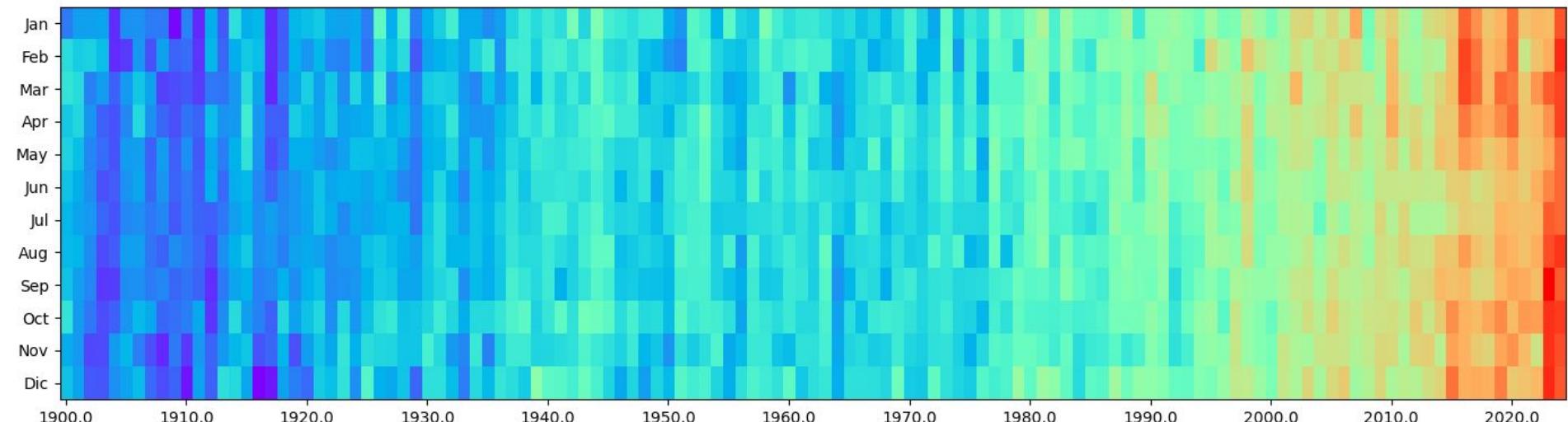
Datos de <https://ourworldindata.org/>

Visualización en Python con matplotlib

- `plt.imshow()`: mapas de color (datos 2D)

➤ `plot_global_temp_anomaly.py`

`matplotlib.ipynb`



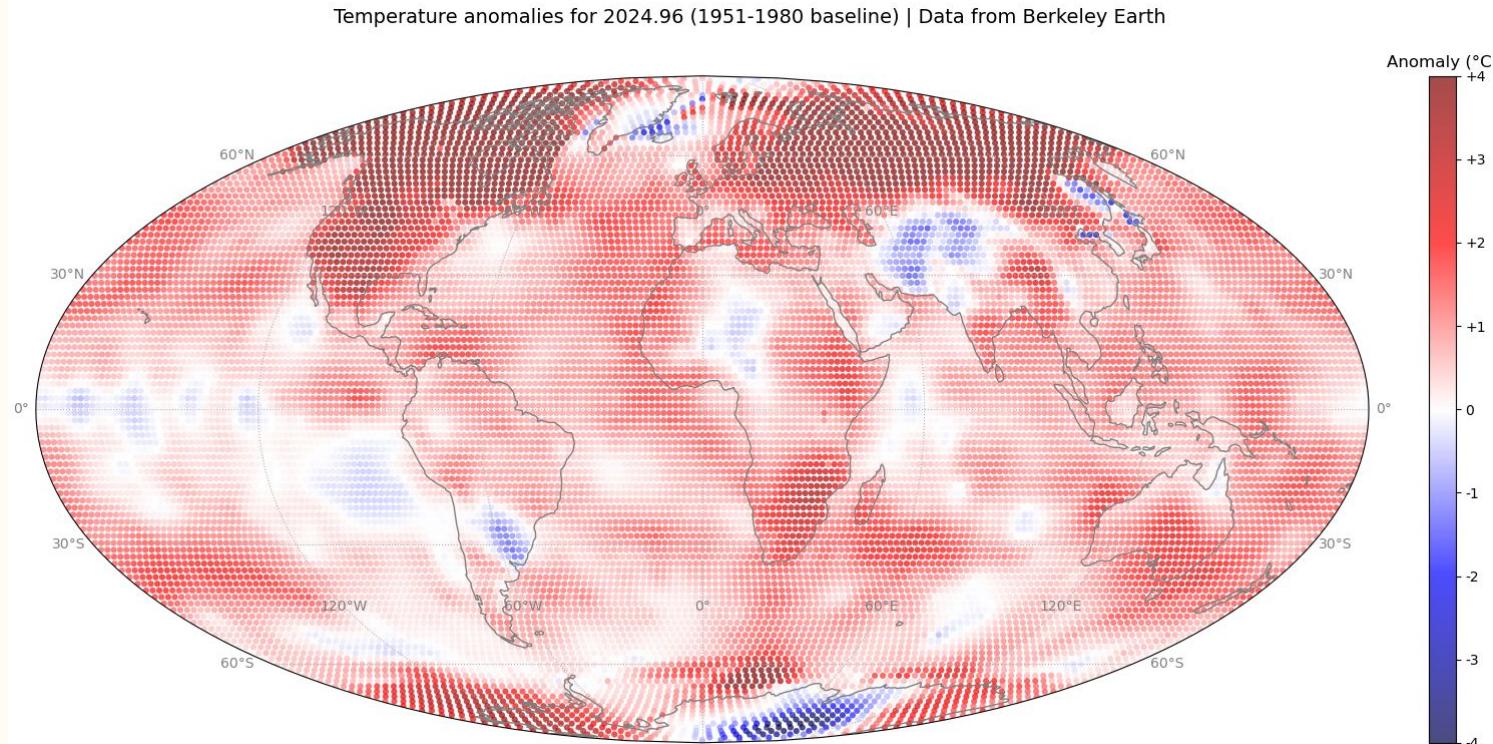
Datos de <https://berkeleyearth.org>

Visualización de datos geoespaciales en Python

Con la librería **cartopy** se pueden graficar datos geoespaciales.

<https://scitools.org.uk/cartopy/>

plot_temperature_map_cartopy.ipynb



Limpieza de datos

Es el proceso de detectar y corregir datos faltantes o inválidos en un conjunto de datos.

Cuando faltan valores en un archivo, usualmente se leerán como **NaN** (Not A Number) o **NA** (Not Available)

limpieza.ipynb

Estrategias para lidiar con valores faltantes:

- **Descartar** datos inválidos
 - Filas (muestras) con NaNs en alguna columna
 - Columnas enteras (variables) si tienen demasiados NaNs
 - Elimina información
- **Imputación** de datos (rellenar)
 - Se usa la media o mediana de una columna para llenar faltantes
 - Inventamos información que no existía; implica suposiciones sobre los datos

Ninguna estrategia es perfecta: se debe juzgar qué hacer en cada caso.

Análisis de datos: estadística descriptiva

Proporciona estadísticas numéricas simples que describen los datos:

- **Centralidad:** ¿Cuál es un valor típico?
- **Dispersión:** ¿Qué tan variables son los datos (comparado con lo típico)?
- **Rango:** ¿Qué tan ancho es el intervalo de los datos?

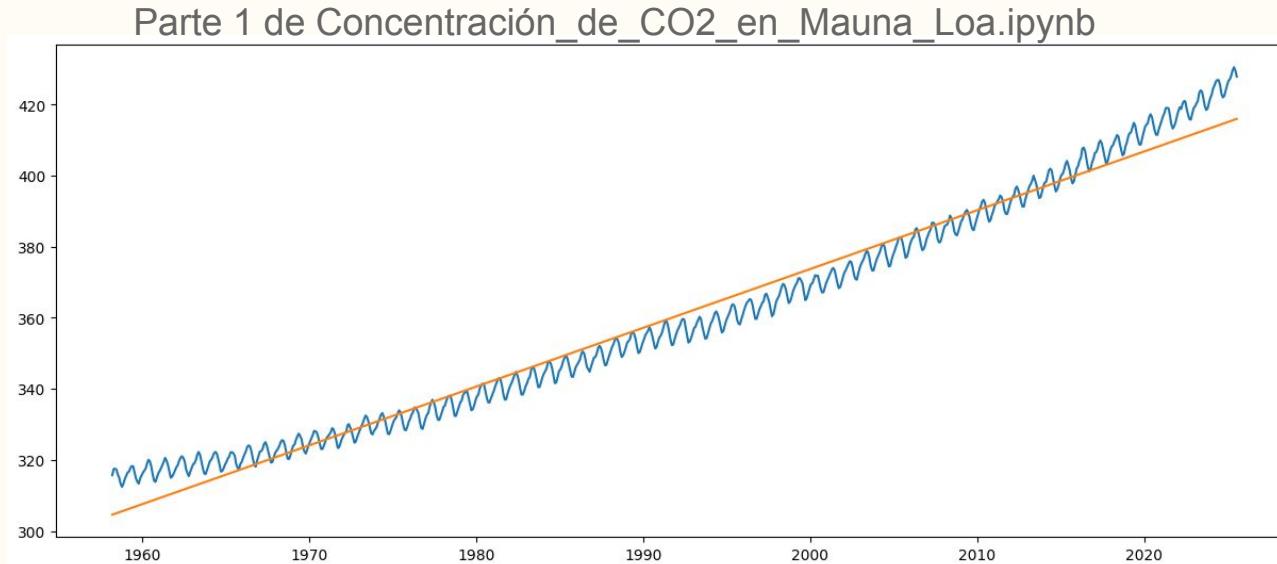
```
import numpy as np
import pandas as pd
x = pd.Series([10, 12, 13, 15, 100])
mean = x.mean()
median = x.median()
std = x.std()
min_val = x.min()
max_val = x.max()
q1 = x.quantile(0.25)
q3 = x.quantile(0.75)
```

Media	30.0
Mediana	13.0
Desv. Est.	39.0
Mínimo	10
Cuartil 1	12
Cuartil 3	15
Máximo	100

Análisis de datos: tendencia

Quizás la manera más sencilla de modelar la tendencia, puesto que cambia lentamente, es usando un **ajuste de curva** lineal o cuadrático.

La función **curve_fit()**, de `scipy.optimize`, permite hacer **ajustar un modelo** matemático (no necesariamente una línea) a datos.



Análisis de datos: media móvil

Los modelos matemáticos simples no siempre se ajustan tan bien a los datos reales.

Otra técnica para extraer tendencia, muy simple y bastante efectiva, es el **suavizado**.

Esto elimina la variabilidad de alta frecuencia y permite separar la tendencia de las componentes periódicas.

Una manera sencilla de suavizar es calculando una **media móvil**:

$$M_t = \frac{X_t + X_{t-1} + X_{t-2} + \cdots + X_{t-N+1}}{N}$$

Media móvil al tiempo t

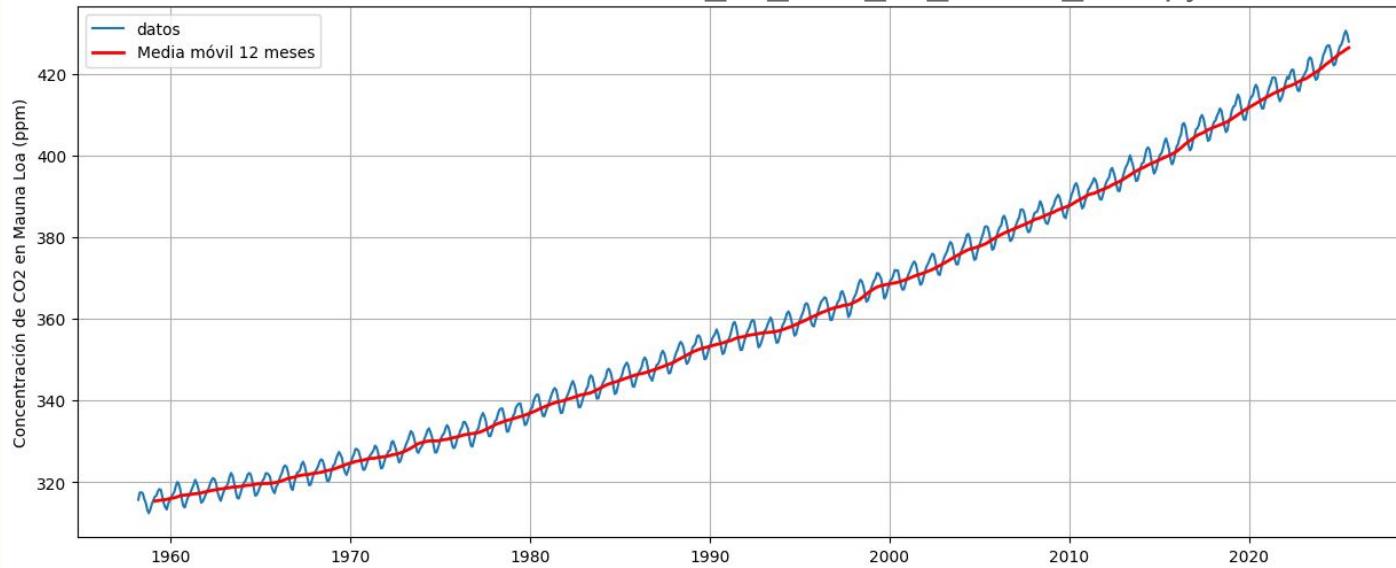
Promedio del valor a tiempo t
y los N-1 valores anteriores

Análisis de datos: media móvil

Pandas tiene una función integrada para calcular medias móviles (llamada en inglés moving average o también *rolling average*):

```
ts.rolling(window=N).mean()
```

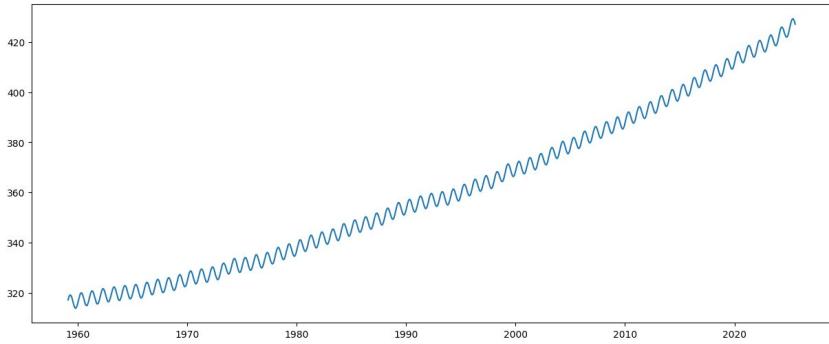
Parte 2 de Concentración_de_CO2_en_Mauna_Loa.ipynb



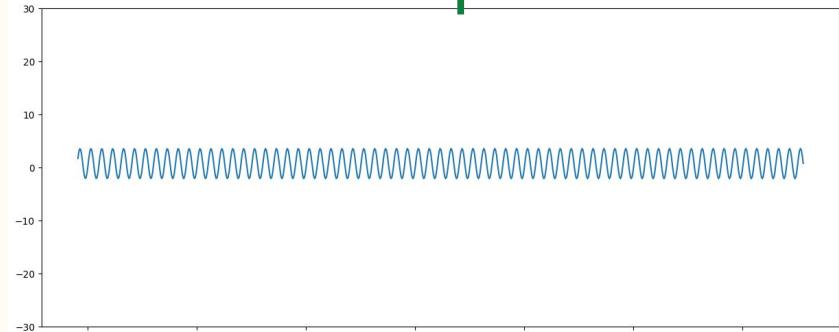
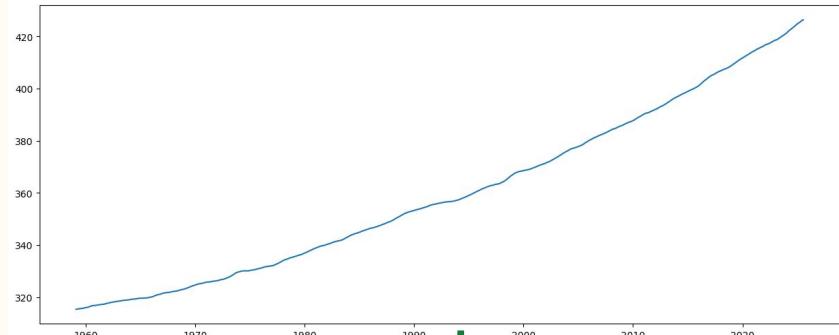
Análisis de datos: media móvil

Podemos ahora ajustar un modelo cosenoidal a la estacionalidad.

Finalmente, construimos un modelo compuesto por la tendencia (media móvil).



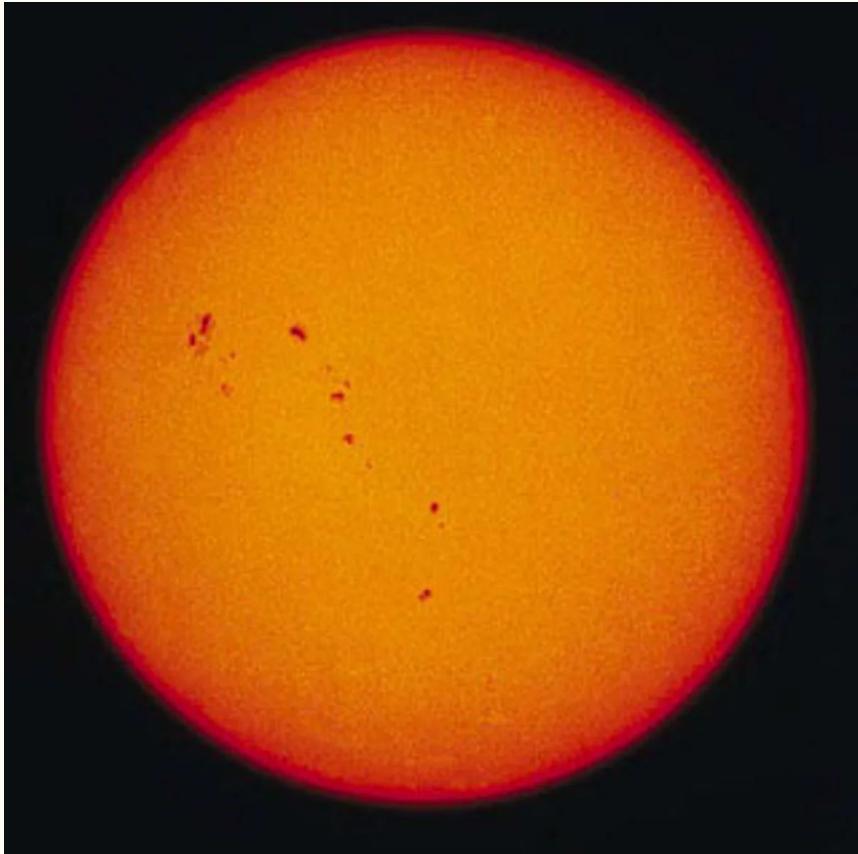
=



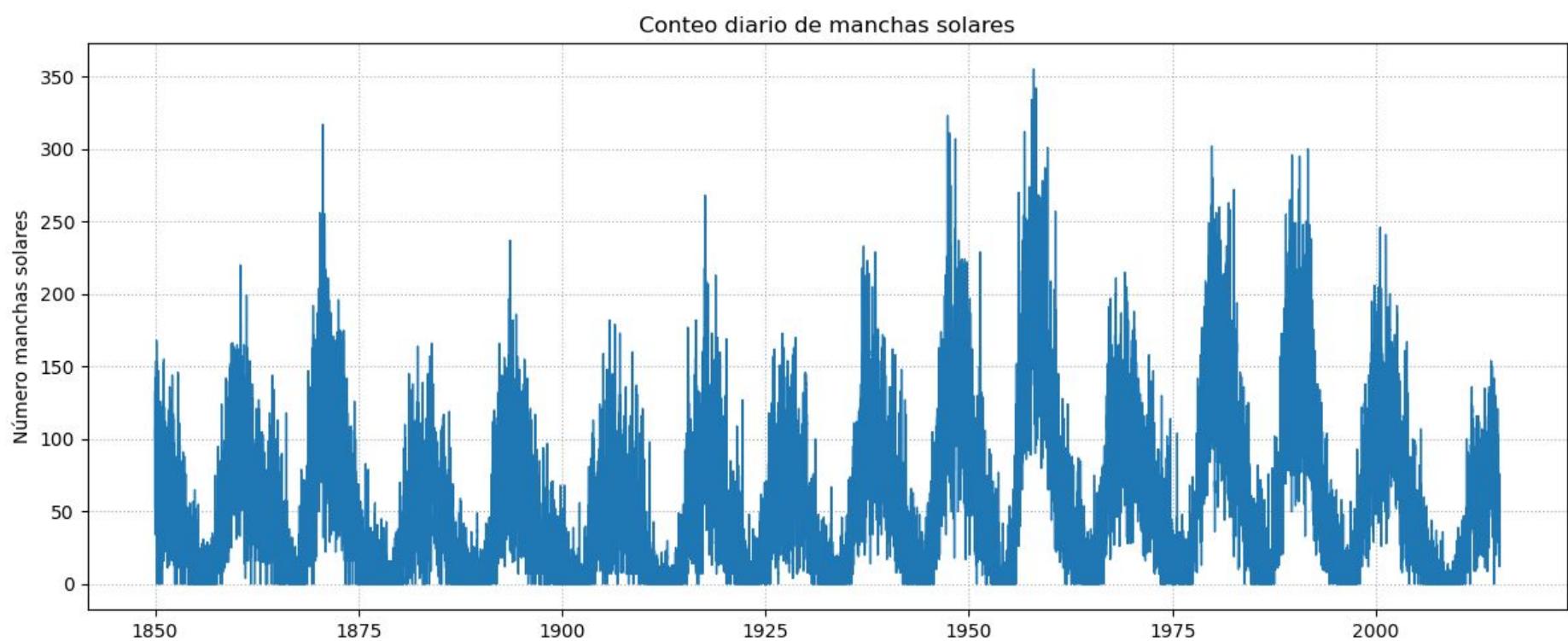
Análisis de datos: periodicidades

Manchas solares: regiones ligeramente más frías y por tanto menos brillantes de la superficie solar

Son un buen indicador de la actividad solar, relacionada con llamaradas y eyecciones de masa que pueden causar problemas en la Tierra y en el espacio cercano.

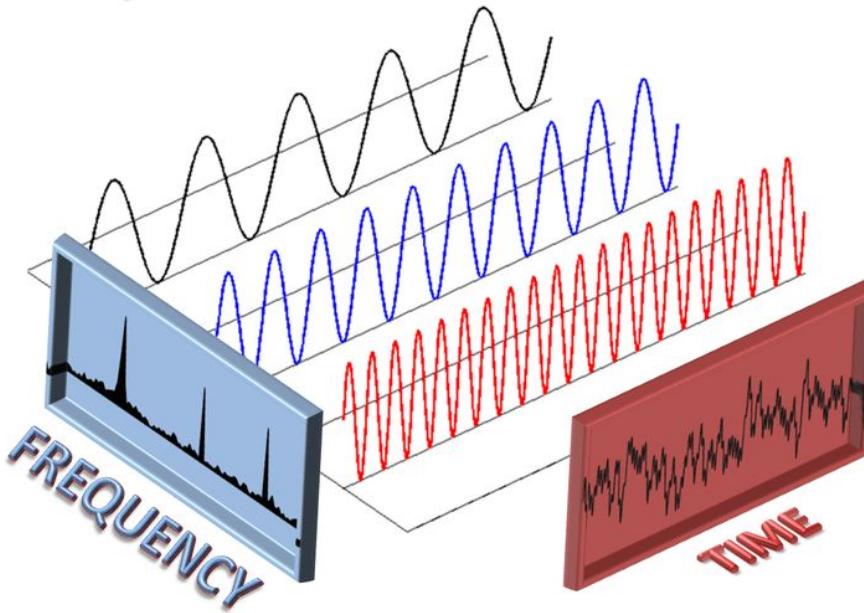


Análisis de datos: periodicidades



Análisis de datos: periodicidades

La transformada de Fourier nos permite encontrar periodicidades.



$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n]$$



[SciPy User Guide](#) > [Fourier Transforms \(scipy.fft\)](#)

[Fourier Transforms \(scipy.fft\)](#)

Contents

[Fourier Transforms \(scipy.fft\)](#)

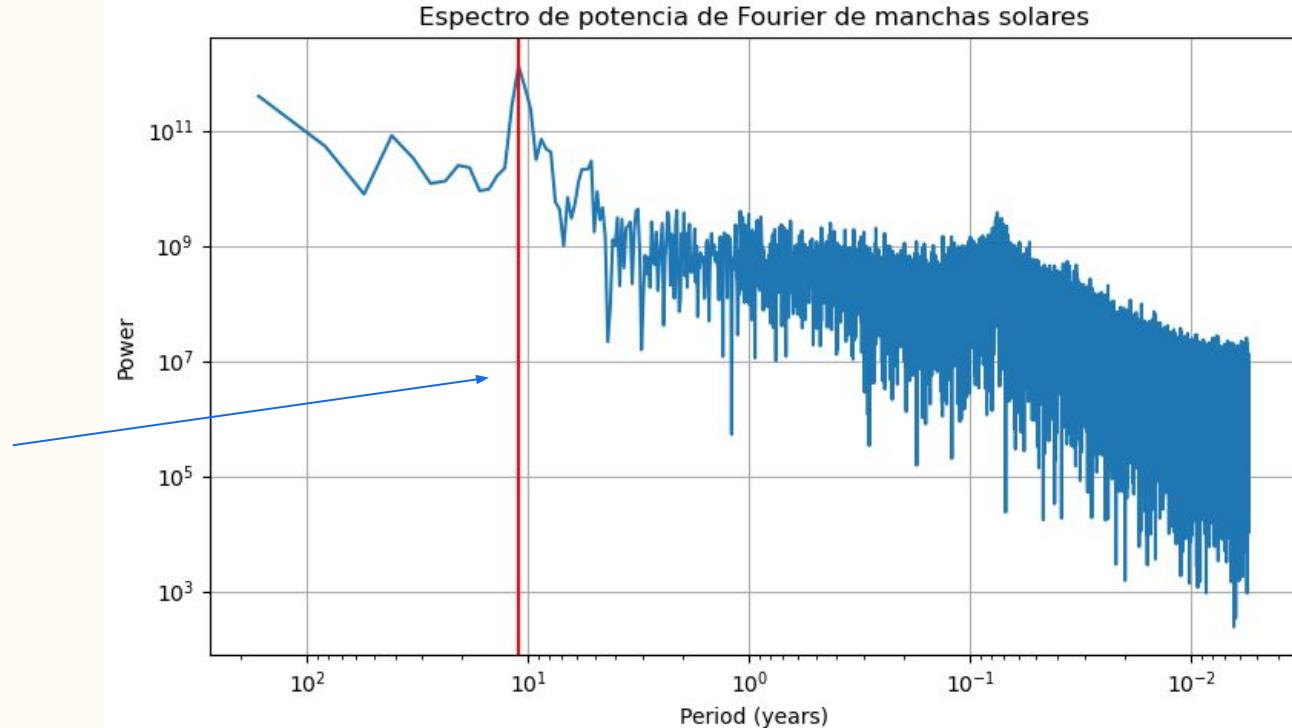
- [Fast Fourier transforms](#)
 - [1-D discrete Fourier transforms](#)
 - [2- and N-D discrete Fourier transforms](#)
- [Discrete Cosine Transforms](#)

Análisis de datos: periodicidades

La transformada de Fourier nos permite encontrar periodicidades.

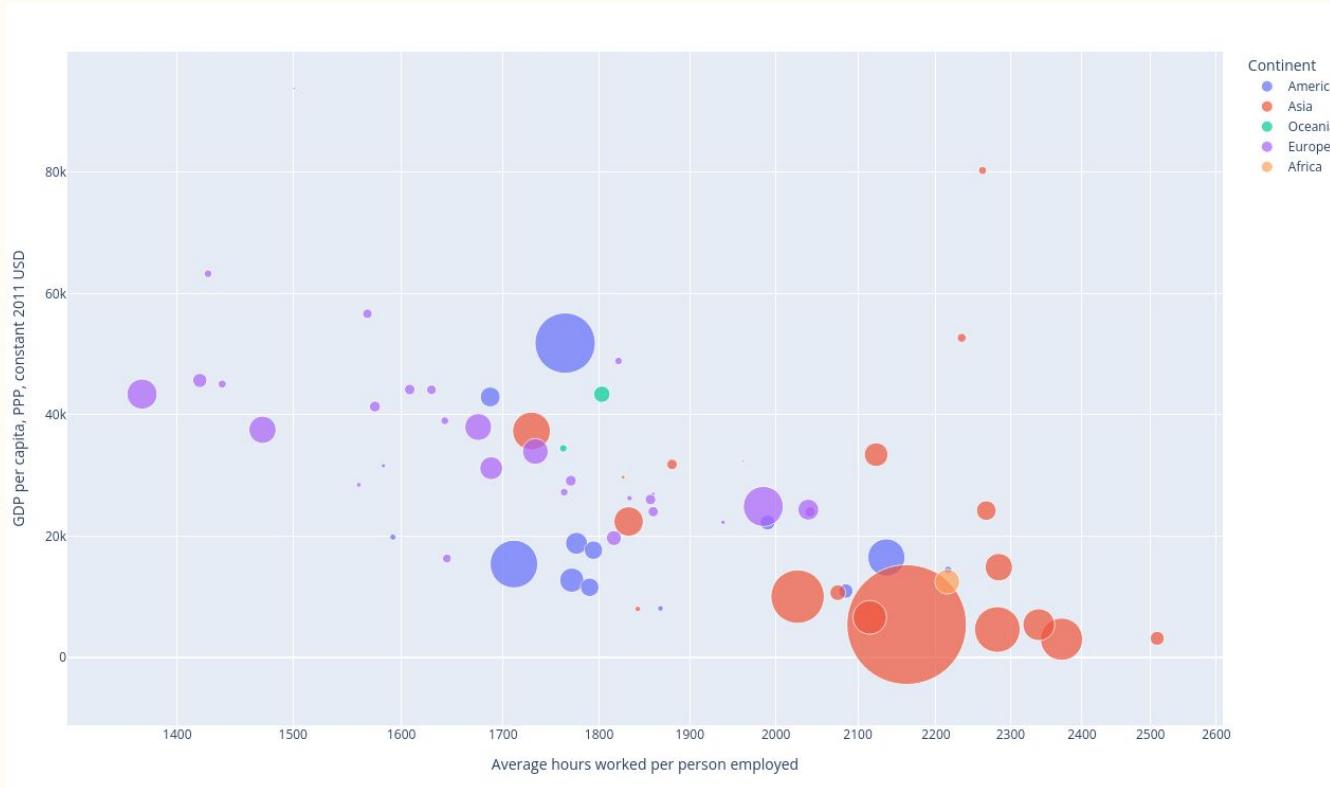
[manchas_solares.ipynb](#)

Periodo de manchas solares: 11 años (el "ciclo solar")



Visualización interactiva: Plotly

Plotly es una librería de Python enfocada en plots bonitos e interactivos.



plotly_ejemplo.py

Visualización interactiva: Plotly

Una ventaja de plotly es que existe una versión de **Javascript** ([Plotly.js](https://plotly.js.org/)) que puede usarse para gráficas interactivas en web.

➤ https://meithan.net/hours_worked/

