

# Visualización y Análisis de datos con Python

Escuela de Código de PILARES

**Dr. Juan Claudio Toledo Roy**

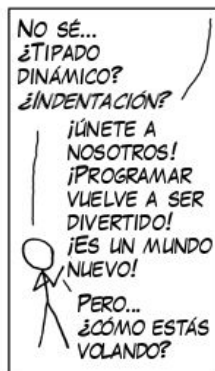
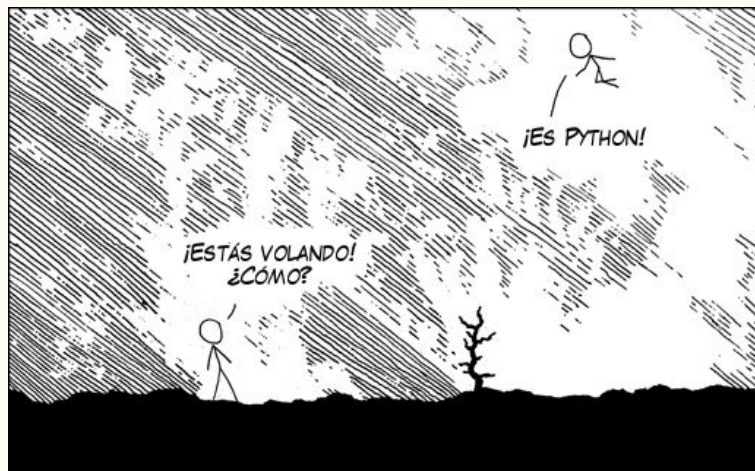
Instituto de Ciencias Nucleares UNAM

[juan.toledo@nucleares.unam.mx](mailto:juan.toledo@nucleares.unam.mx)

# ¿Qué veremos en este mini curso?

- Python como lenguaje de análisis y visualización de datos
- Introducción al análisis de series de tiempo en Python
- Lectura de datos
- Visualización de datos
- Manejo de datos espaciales

# 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



Aug 2025	Aug 2024	Change	Programming Language		Ratings	Change
1	1			Python	26.14%	+8.10%
2	2			C++	9.18%	-0.86%
3	3			C	9.03%	-0.15%
4	4			Java	8.59%	-0.58%
5	5			C#	5.52%	-0.87%
6	6			JavaScript	3.15%	-0.76%
7	8	^		Visual Basic	2.33%	+0.15%
8	9	^		Go	2.11%	+0.08%
9	25	^^		Perl	2.08%	+1.17%
10	12	^		Delphi/Object Pascal	1.82%	+0.19%

<https://www.tiobe.com/tiobe-index/>

# Generalidades de Python

Características principales:

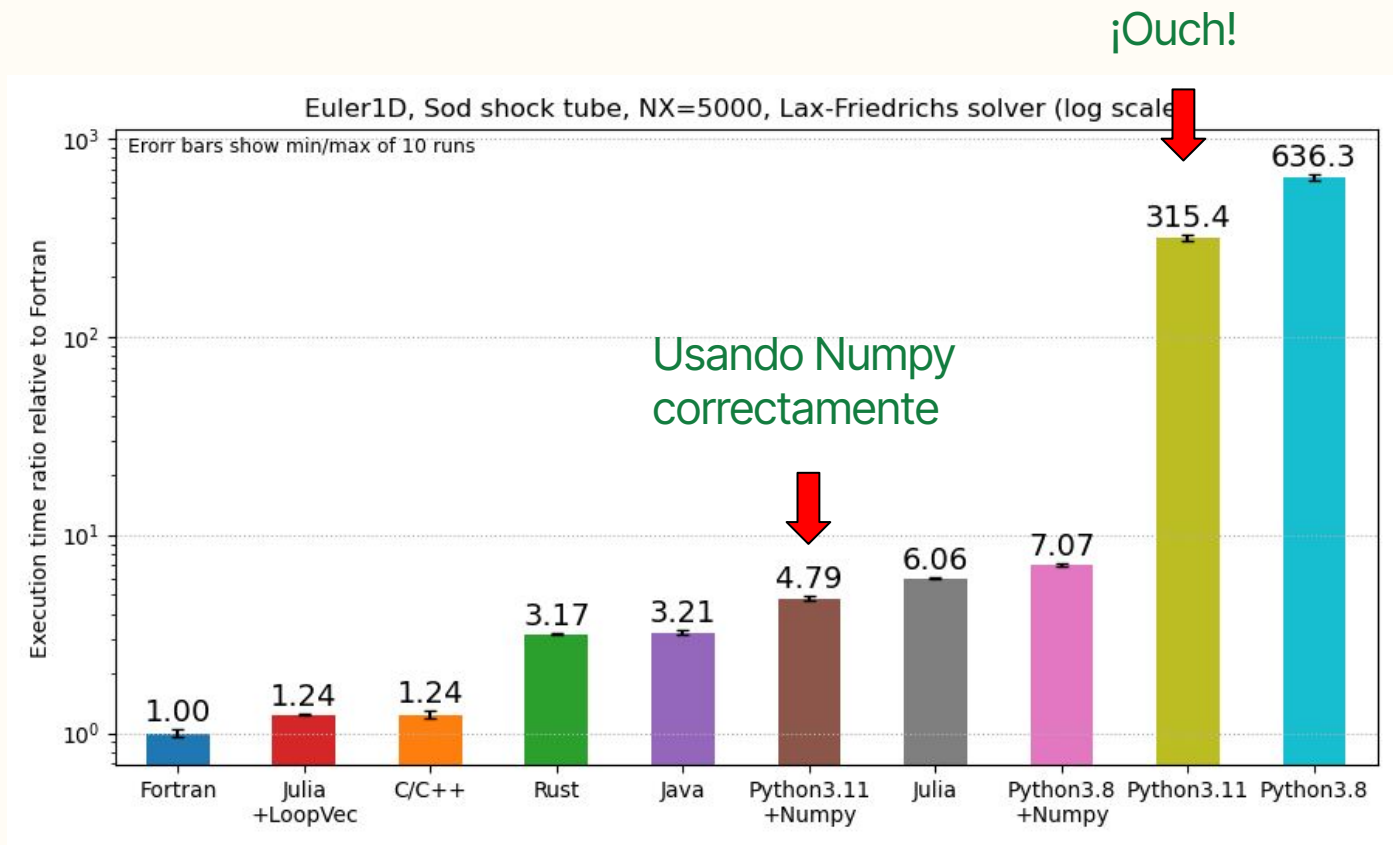
- Lenguaje de (muy) **alto nivel**
- **Dinámicamente tipado, bytecode interpretado**
- Sintaxis **ligera, intuitiva y poderosa**
- **Multi-paradigma**: procedural, orientado a objetos, funcional, etc.
- **Extensa librería estándar** (estructuras de datos, matemáticas, tipos de datos, etc)
- **Multi-plataforma** y tremendamente 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 sintácticamente poderoso y ligero, no para ser rápido
- Hay formas de usarlo mucho más rápidas, sobre todo vía librerías como Numpy
- Existen algunos compiladores avanzados

# Generalidades de Python





# 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(arr[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
  - **matplotlib**: graficación y visualización de datos
  - **pandas**: especializada en análisis de datos
  - **scikit-learn**: machine learning
  - **NetworkX**: análisis de redes
  - **PyTorch, TensorFlow**: deep learning, IA

 **matplotlib**

 **NumPy**

 **SciPy**

 **pandas**

 **scikit-learn**

 **NetworkX**  
Network Analysis in Python

 **TensorFlow**

# 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**



The screenshot shows the PyPI page for matplotlib 3.10.5. A red arrow points to the `pip install matplotlib` button. The page includes a search bar, navigation links (Help, Docs, Sponsors, Log in, Register), and a 'Latest version' badge. The project description section shows various badges for downloads, testing, and documentation. The matplotlib logo is prominently displayed at the bottom.

Type '/' to search projects

Help Docs Sponsors Log in Register

**matplotlib 3.10.5** ✓ Latest version

`pip install matplotlib`

Released: Jul 31, 2025

Python plotting package

**Navigation**

- Project description
- Release history
- Download files

**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**

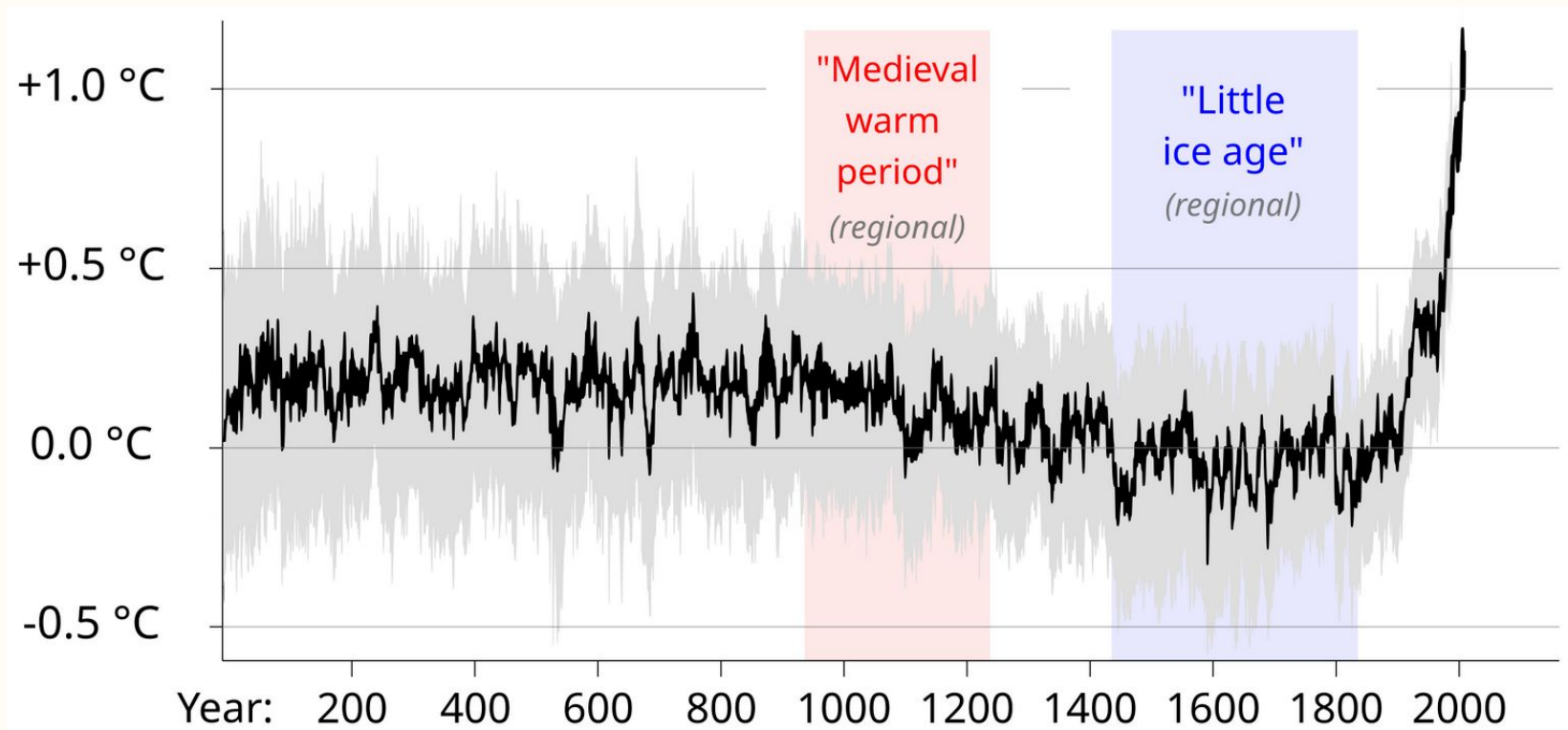
# Introducción al análisis de series de tiempo

**Serie de tiempo:** secuencia de datos ordenados en el tiempo

- **Ubicuas:** casi todos los sistemas naturales y artificiales tienen señales asociadas que cambian con el tiempo
- El **orden temporal** da cabida a que exista **correlación temporal** en los datos: el valor presente puede depender de los valores pasados
- Identificar **patrones** temporales del sistema: tendencias, periodicidades, etc
- Entender **correlaciones** y relaciones **causales** entre dos o más fenómenos
- Hacer **predicciones** del comportamiento futuro
- Monitorear y detectar **anomalías**

# Introducción al análisis de series de tiempo

Anomalía de temperatura planetaria promedio en los últimos 2000 años



# Introducción al análisis de series de tiempo

**Componentes** de una serie de tiempo (modelo)

$$X_t = T_t + S_t + C_t + R_t$$

## Tendencia

Cambios de los datos a largo plazo

No tiene que ser lineal, y puede cambiar de dirección

## Estacionalidad

Patrones repetitivos con un periodo definido.

Usualmente podemos asociarla a un fenómeno (p. ej. las estaciones del año)

## Ciclicidad

Cambios cíclicos sin un periodo bien definido

Es difícil identificar un periodo claro para esta variabilidad.

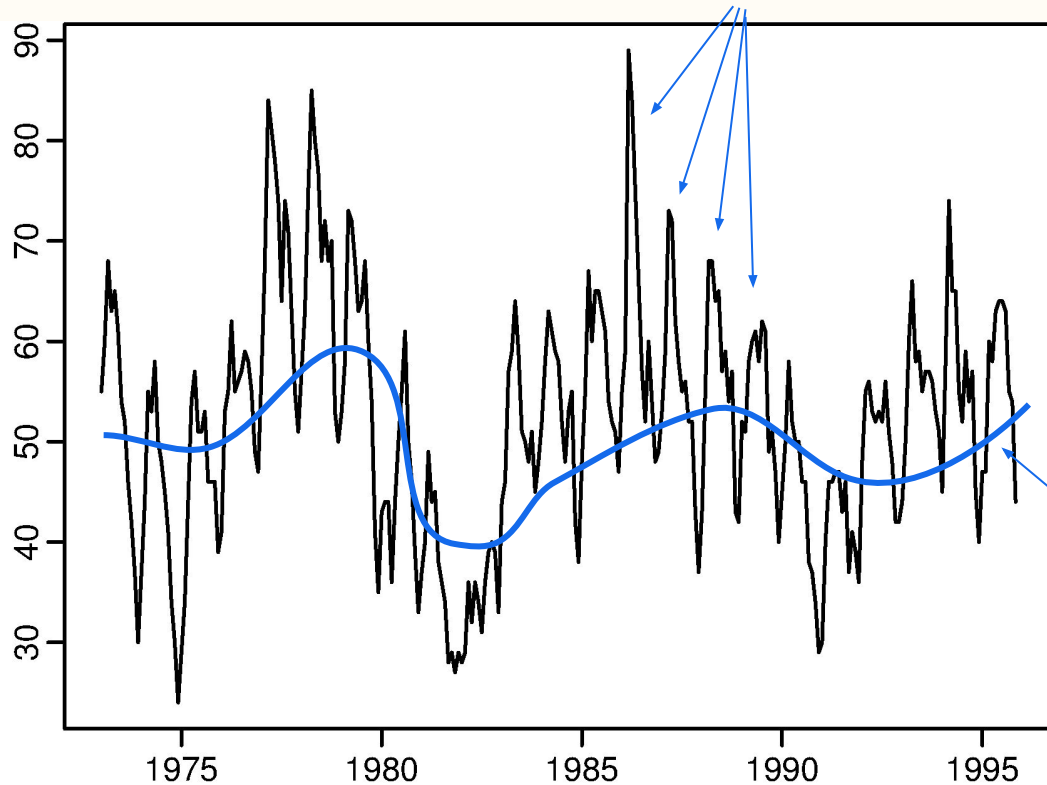
## Residuo

Fluctuaciones irregulares y usualmente impredecibles (ruido, aleatoriedad, etc)

# Introducción al análisis de series de tiempo

Tendencia: NO

Estacionalidad

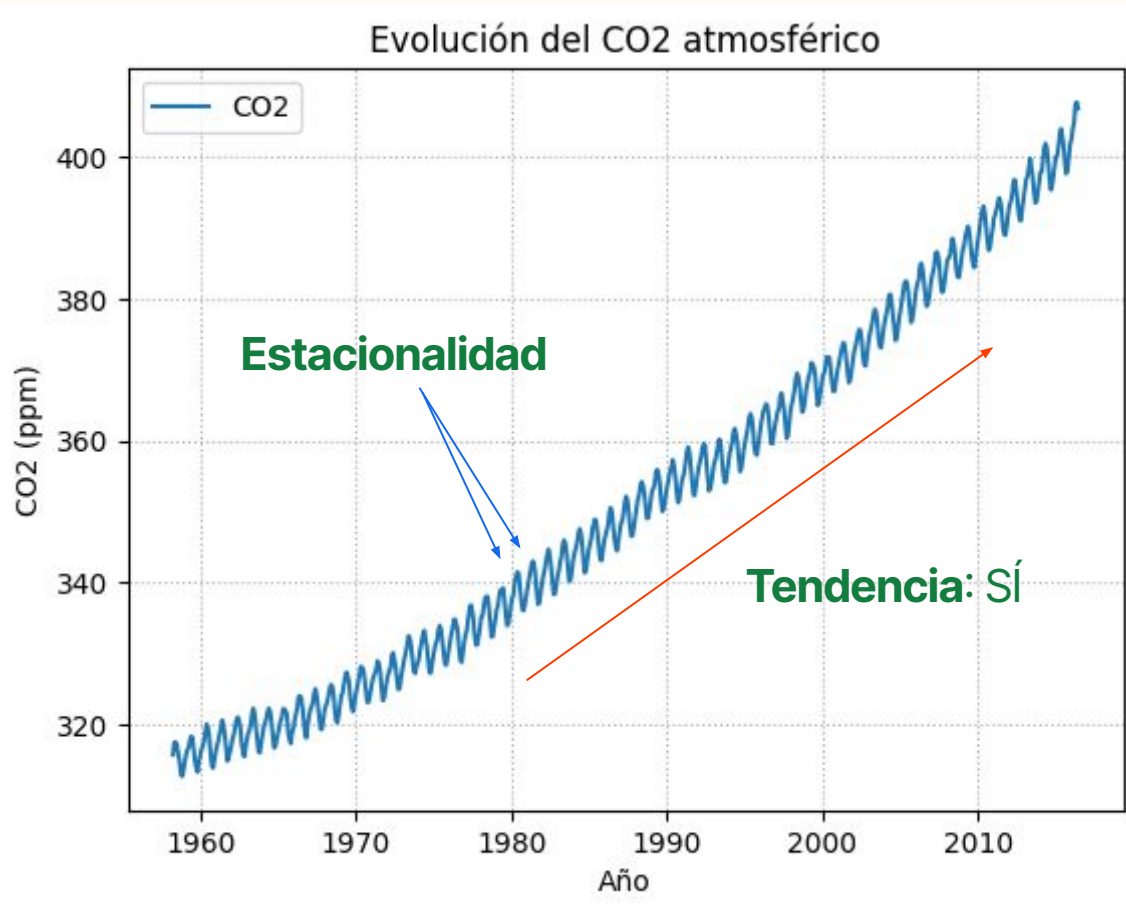


Ventas  
inmobiliarias en  
EE.UU. (millones)

¿Ciclicidad?

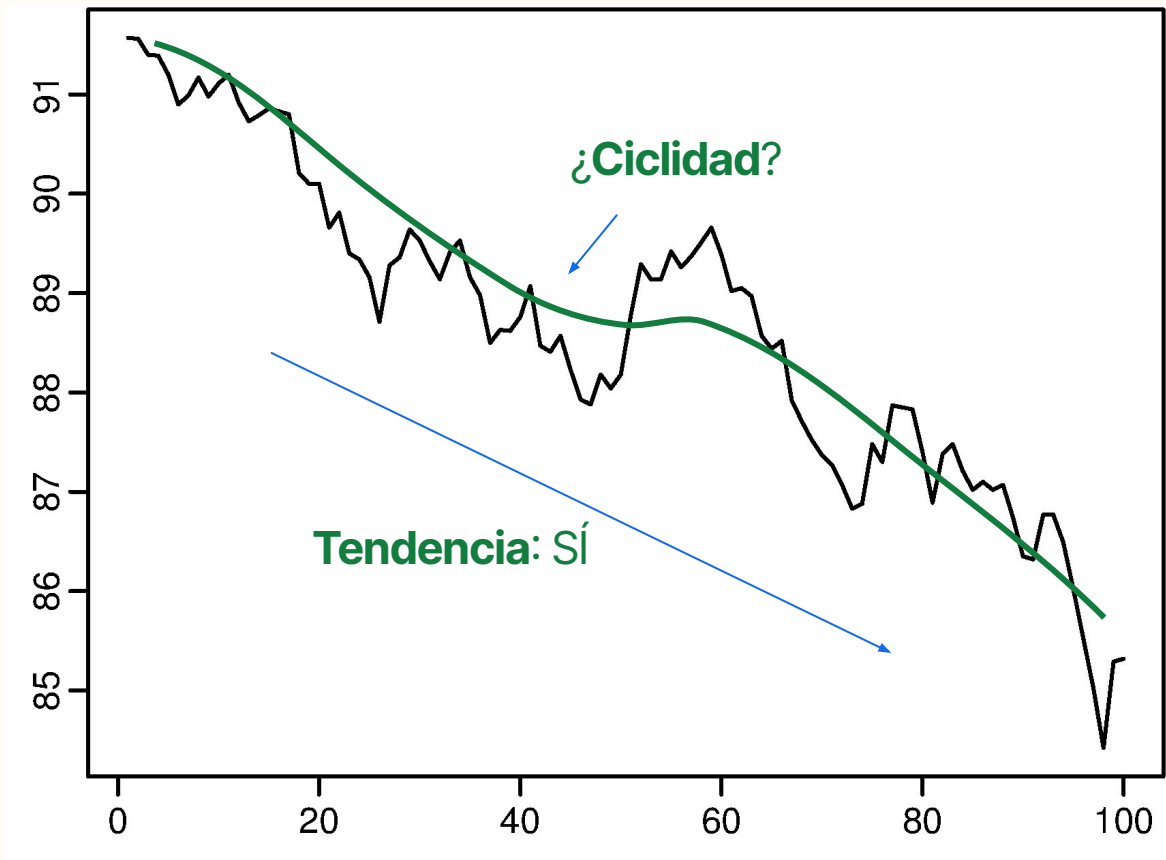


# Introducción al análisis de series de tiempo



¿NO Ciclidad?

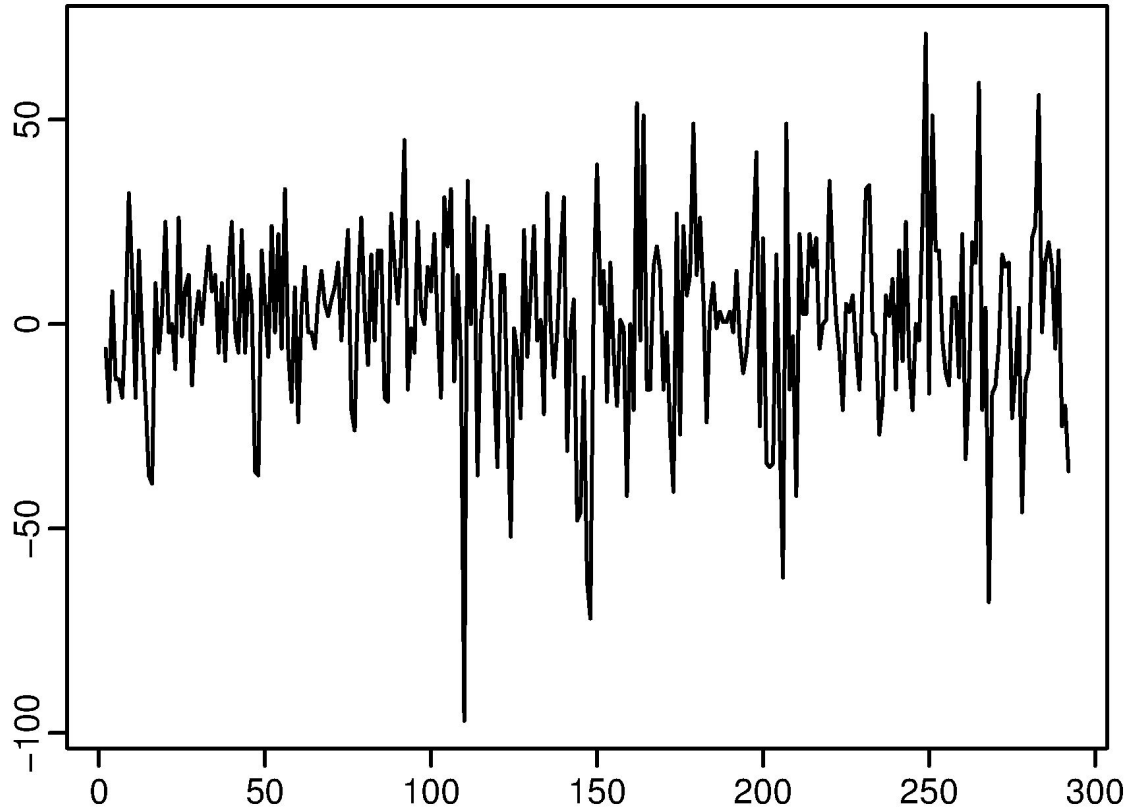
# Introducción al análisis de series de tiempo



Contratos de deuda del  
Depto. Tesoro EE.UU

**Estacionalidad:**  
NO

# Introducción al análisis de series de tiempo



Índice Dow Jones

**Tendencia:** NO

**Estacionalidad:** NO

**Ciclicidad:** NO

**¿Sólo ruido?**

# Introducción al análisis de series de tiempo

Antes de analizar cualquier serie de tiempo ...

## ¡GRAFICAR LA SERIE DE TIEMPO!

- Descubrir rápidamente las propiedades generales de la serie
- Determinar el tipo de preguntas que podemos contestar
- Seleccionar adecuadamente las herramientas para analizarla

# 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()  
    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 85
```

# 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("C02.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 en automático a números (floats o ints)
- El archivo debe contener sólo números, y mismo número de valores por fila
- Se pueden brincar líneas de encabezado y comentarios
- Se puede cambiar el `delimiter` de las columnas (default es espacio)

# Lectura de datos en Python

- La sintaxis de manejos de arreglos de Numpy es práctica

```
import numpy as np
datos = np.loadtxt("co2_mm_mlo.txt", comments="#")
print(datos)
print(datos.shape)
mes = datos[:,2]
CO2 = datos[:,3]
incert = datos[:,7]
```



# Lectura de datos en Python

- Usando **pandas** (múltiples formatos)

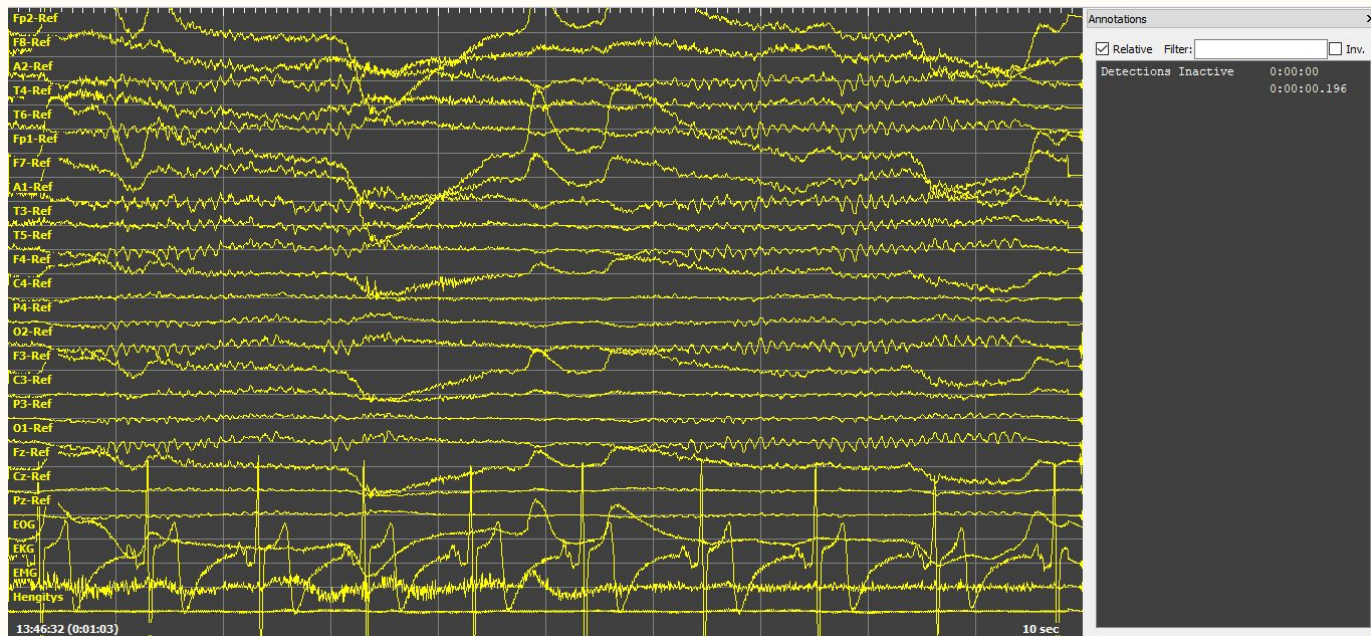
```
import pandas as pd
df = pd.read_csv("CO2.csv")
print(df)
print(type(df))
print(df.columns)
print(df["CO2 (ppm)"])
```

- 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: usualmente hay librerías de terceros en Python

Formato **EDF** (series de tiempo médicas): **PyEDFlib**



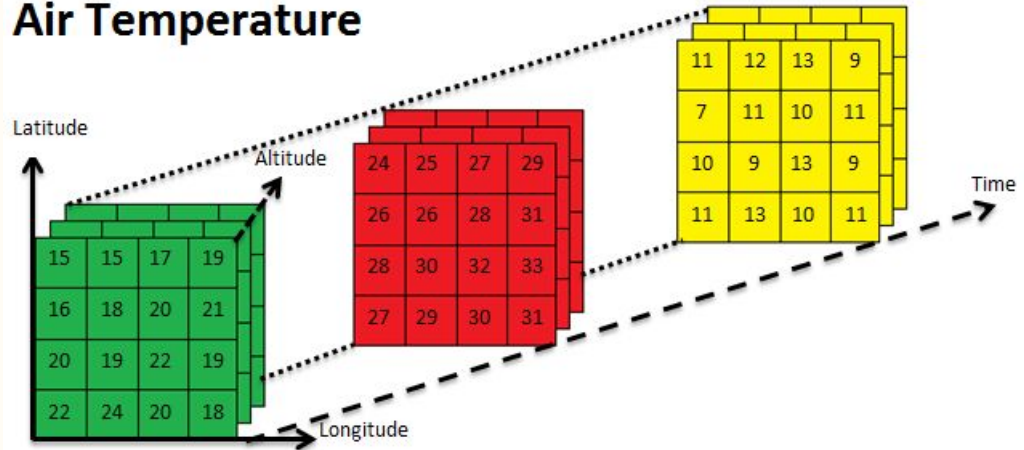
# Lectura de datos en Python

- Formatos especializados: usualmente hay librerías de terceros en Python

Formato **NetCDF** (series de tiempo y datos geospaciales): **netCDF4**

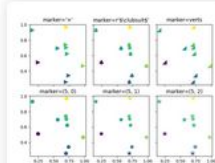


## Air Temperature

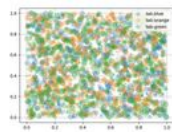


# Visualización en Python con matplotlib

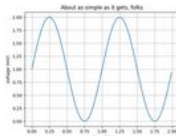
- **matplotlib** es una poderosa librería de graficación y visualización
- Muchos tipos de gráficos: líneas, scatters, barras, heatmaps, vectores, 3D, etc.
- Se usa mucho en conjunto con Numpy (de hecho, es una dependencia)



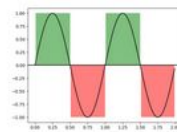
Marker examples



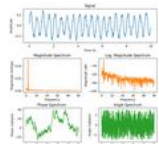
Scatter plot with a legend



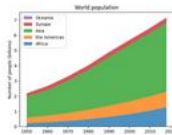
Line plot



Shade regions defined by a logical mask using `fill_between`



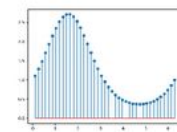
Spectrum representations



Stackplots and streamgraphs



Stairs Demo



Stem plot

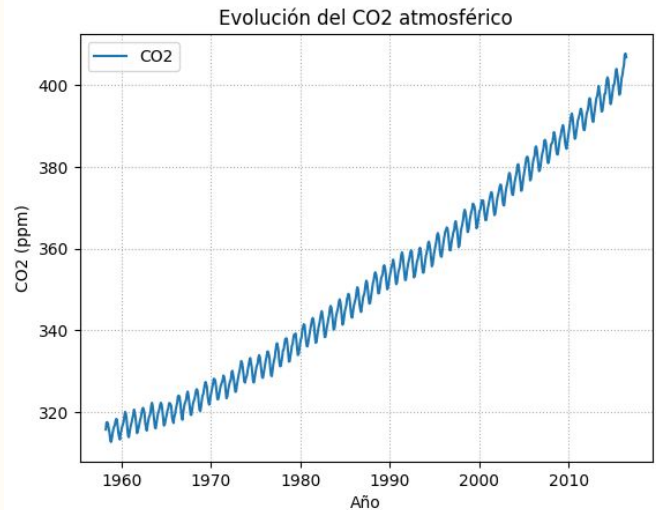
<https://matplotlib.org>



# Visualización en Python con matplotlib

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

```
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()
```

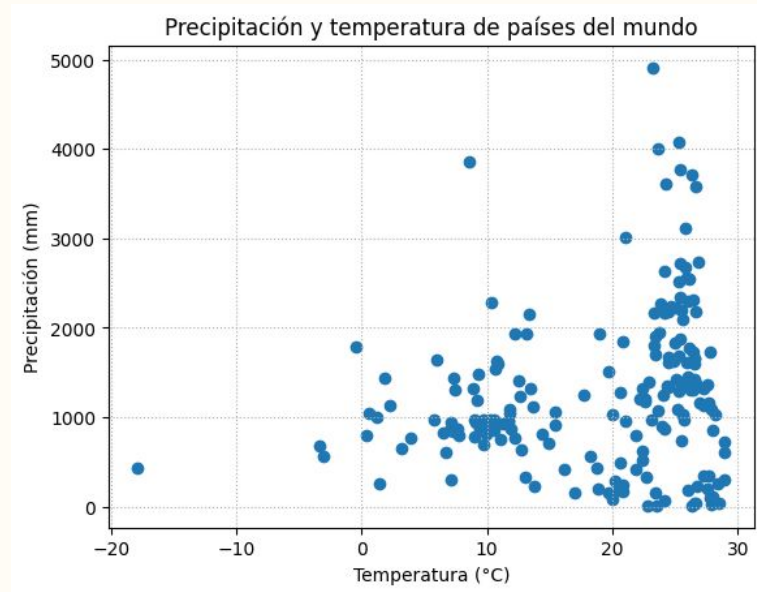


<https://gml.noaa.gov/ccgg/data/>

# Visualización en Python con matplotlib

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

```
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()
```



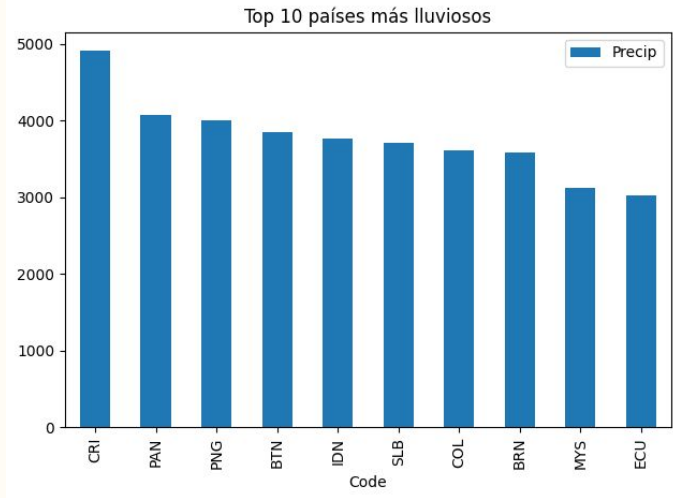
Datos de <https://ourworldindata.org/>



# Visualización en Python con matplotlib

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

```
df = pd.read_csv("temp_precip.csv")
df1 = df.sort_values(by="Precip",
                    ascending=False).head(10)
df1.plot(x="Code", y="Precip",
        kind="bar")
plt.title("Top 10 países más lluviosos")
plt.tight_layout()
plt.show()
```



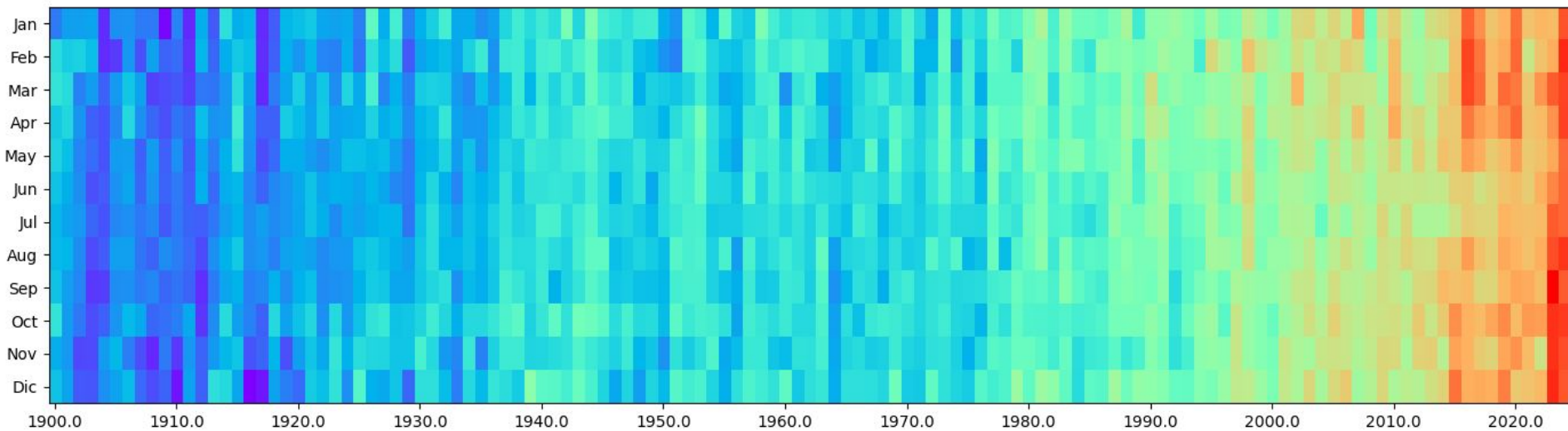
Datos de <https://ourworldindata.org/>



# Visualización en Python con matplotlib

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

➤ `plot_global_temp_anomaly.py`

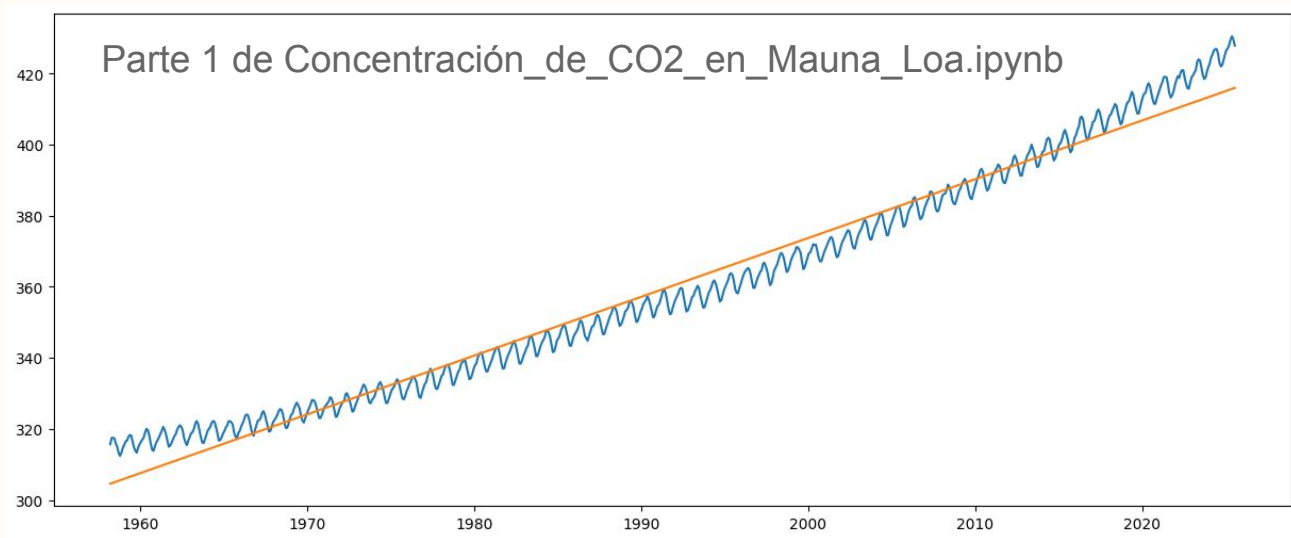


Datos de <https://berkeleyearth.org>

# Análisis de series de tiempo: 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 series de tiempo: 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**:

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

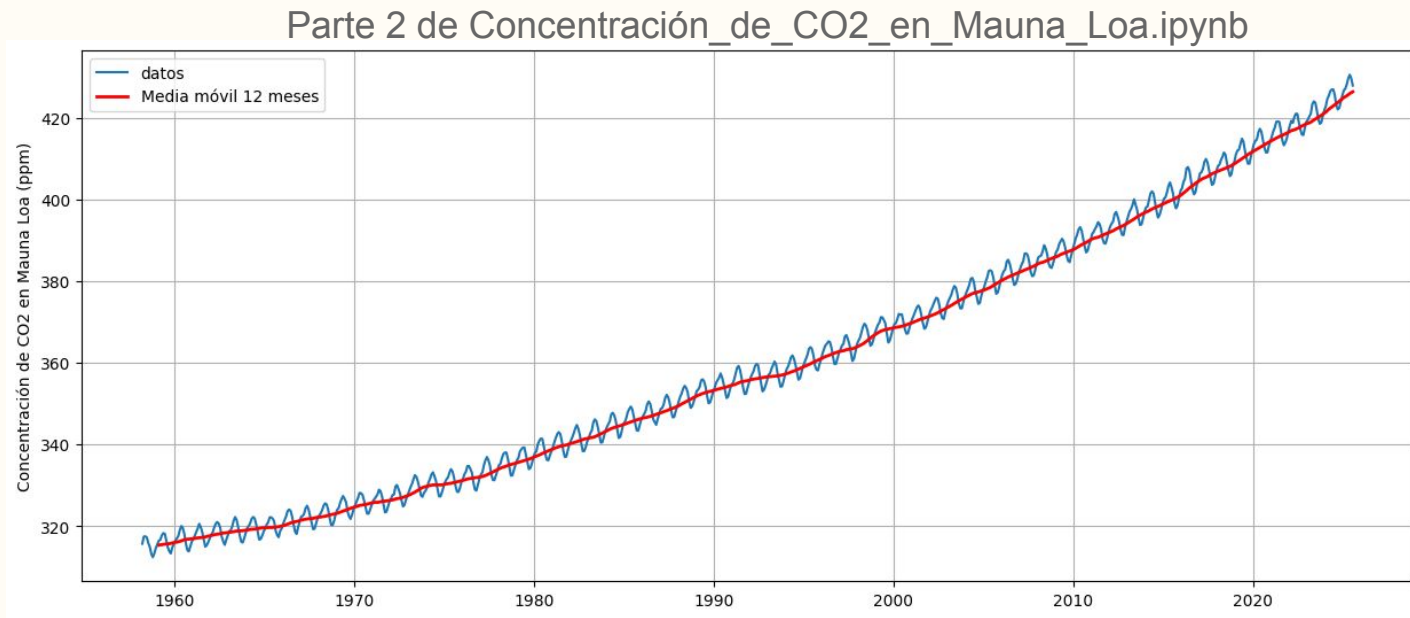
Media móvil al tiempo t

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

# Análisis de series de tiempo: media móvil

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

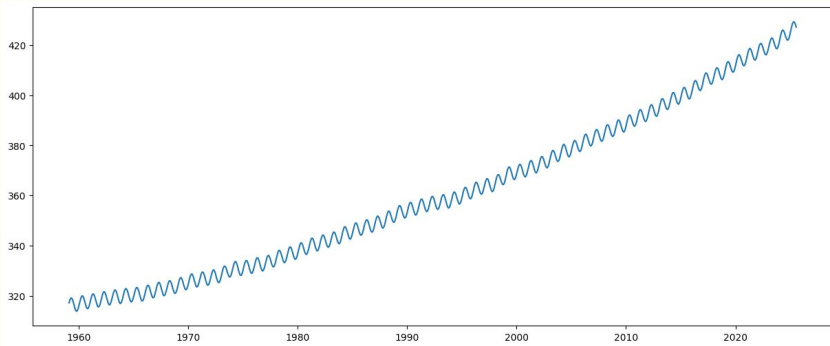
```
ts.rolling(window=N).mean()
```



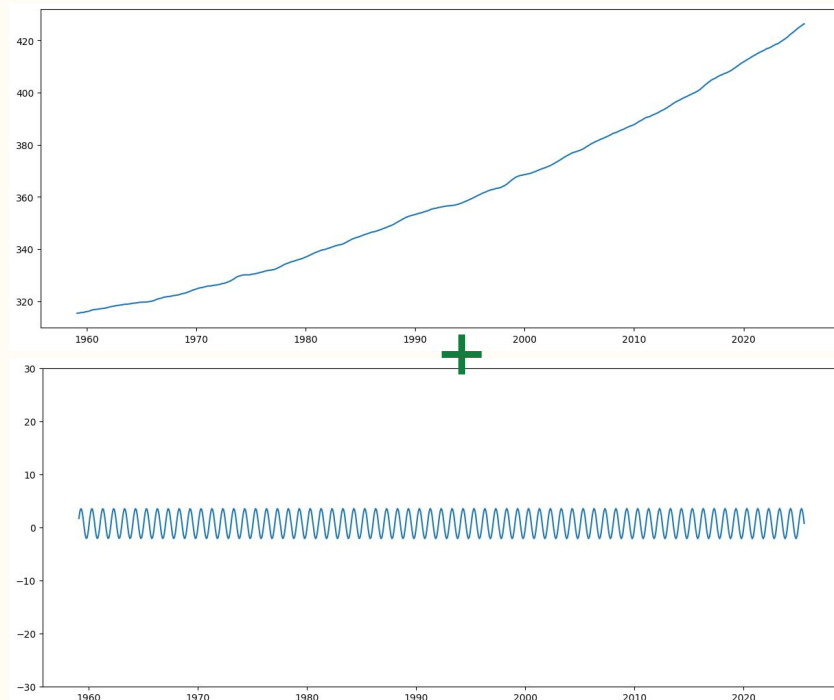
# Análisis de series de tiempo: media móvil

Podemos ahora ajustar un modelo cosenoidal a la estacionalidad.

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



=



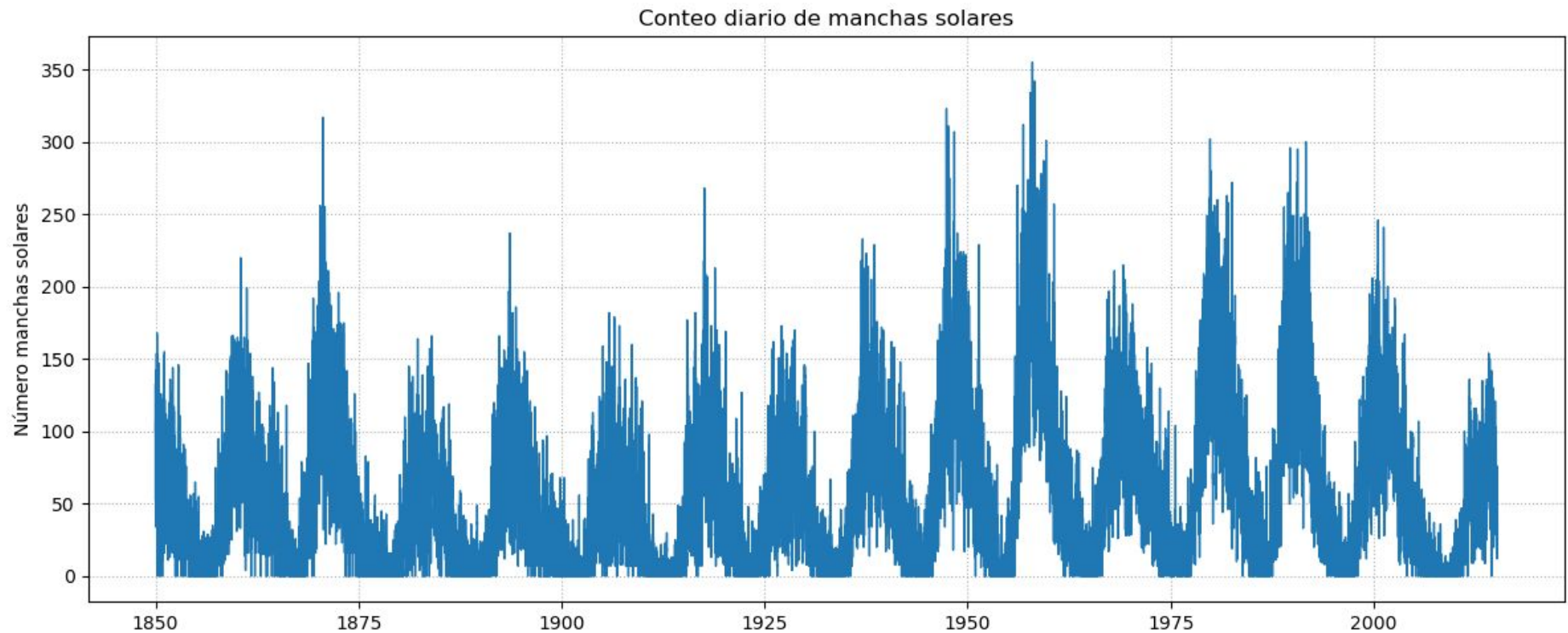
# Análisis de series de tiempo: periodicidades

**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.



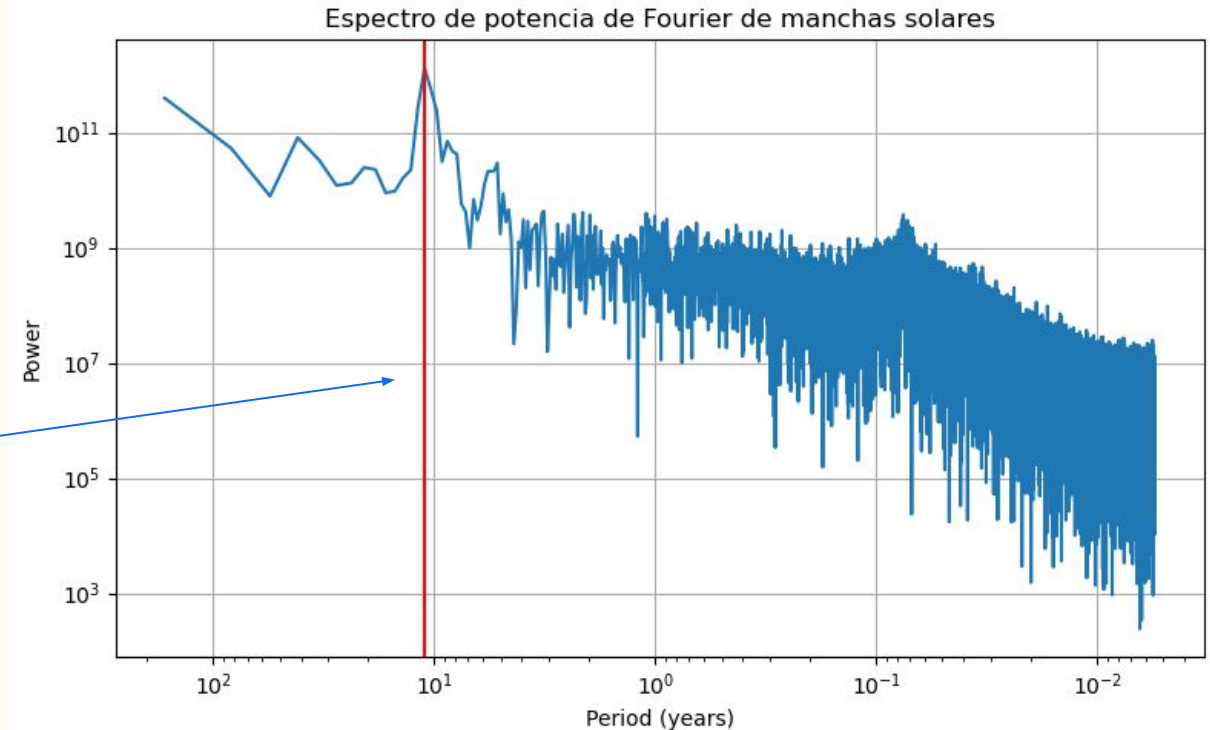
# Análisis de series de tiempo: periodicidades



# Análisis de series de tiempo: periodicidades

La **transformada de Fourier** nos permite encontrar periodicidades.

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





# Visualización de datos espaciales en Python

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

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

plot\_temperature\_map\_cartopy.py

