Academy

CURSO

# Python aplicado a finanzas

MÓDULO 3

# Archivo CSV

- Tipo de documento en formato abierto sencillo.
- Representa datos en forma de table
- Las columnas se separan por comas

| Año | Marca | Modelo | Descripción | Precio |
|---|---|---|---|---|
| 1997 | Ford | E350 | ac, ABS, moon | 3000.00 |
| 1999 | Chevy | Venture | Extended Edition | 4900.00 |
| 1999 | Chevyr | Venture | Extended Edition, Very Large | 5000.00 |
| 1996 | Jeep | Grand Cherokee | MUST SELL! air, moon roof, loaded | 4799.00 |

```
Año,Marca,Modelo,Descripción,Precio
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevyr,Venture,Extended Edition,4900.00
1999,Chevy,Venture,"Extended Edition, Very Large",5000.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00
```

MÓDULO 3

# Pandas

- Pandas es una librería de software escrita como extensión de NumPy.
- Sirve para manipulación y análisis de datos para el lenguaje de programación Python.

# Características de **Pandas**

- Tipo de datos DataFrame para manipulación de datos con indexación integrada.
- Herramientas para leer y escribir datos entre estructuras de dato en-memoria y formatos de archivo variados.
- Alineación de datos y manejo integrado de datos faltantes.
- Reestructuración y segmentación de conjuntos de datos.
- Segmentación vertical basada en etiquetas, indexación elegante, y segmentación horizontal de grandes conjuntos de datos.
- Inserción y eliminación de columnas en estructuras de datos.
- Agrupación predefinida en la biblioteca lo que permite realizar cadenas de operaciones dividir-aplicar-combinar sobre conjuntos de datos.
- Mezcla y unión de datos.

- Indexación jerárquica de ejes para trabajar con datos de altas dimensiones en estructuras de datos de menor dimensión.
- Funcionalidad de series de tiempo:

1. Generación de rangos de fechas y conversión de frecuencias.
2. Desplazamiento de ventanas estadísticas y de regresiones lineales.
3. Desplazamiento de fechas y retraso

| Tabla de diferencias entre Pandas VS NumPy | | |
|---|---|---|
| | **PANDAS** | **NUMPY** |
| 1 | Cuando tenemos que trabajar con **datos tabulares** , preferimos el módulo p *andas* . | Cuando tenemos que trabajar con **datos numéricos** , preferimos el módulo n *umpy* . |
| 2 | Lo poderoso en Pandas son los **Data Frame y Series.** | Lo Poderoso de Numpy de *numpy* son las **matrices.** |
| 3 | *Pandas* consumen **más memoria** . | *Numpy* es **eficiente en memoria.** |
| 4 | *Pandas* tiene un mejor rendimiento cuando el número de filas es **500K o más.** | *Numpy* tiene un mejor rendimiento cuando el número de filas es **50K o menos.** |
| 5 | La indexación de la serie *pandas* es **muy lenta** en comparación con *las* matrices *numpy* . | La indexación de matrices *numpy* es **muy rápida** . |
| 6 | *Pandas* ofrece un objeto de tabla 2d llamado **DataFrame.** | *Numpy* es capaz de proporcionar **matrices multidimensionales.** |

# Data **frame**

- La estructura de datos también contiene ejes etiquetados (filas y columnas).
- Las operaciones aritméticas se alinean en las etiquetas de fila y columna.
- Se puede considerar como un contenedor similar a un dict para los objetos de la serie.
- La estructura de datos primaria de los pandas.
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

# Diferentes formas de construir un DataFrame

- Antes que nada hay que importar la librería pandas, se suele usar el alias pd

```
import pandas as pd
import numpy as np
```

# DataFrame a partir de Listas

```
[ ] import pandas as pd

    lista = list(range (5))
```

```
[ ] lista
```

```
[0, 1, 2, 3, 4]
```

```
[ ] df = pd.DataFrame(lista)
    df
```

| | 0 |
|---|---|
| **0** | 0 |
| **1** | 1 |
| **2** | 2 |
| **3** | 3 |
| **4** | 4 |

```
[ ] df["unos"]= 1
    df
```

| | 0 | unos |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 1 |
| **2** | 2 | 1 |
| **3** | 3 | 1 |
| **4** | 4 | 1 |

```
[ ] df
```

| | 0 | unos |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 1 |
| **2** | 2 | 1 |
| **3** | 3 | 1 |
| **4** | 4 | 1 |

```
[ ] lista2 = list(range(10,15,1))
    lista2
```

```
[10, 11, 12, 13, 14]
```

```
[ ] df["del10al14"] = [10, 11, 12, 13, 14]
    df
```

| | 0 | unos | del10al14 |
|---|---|---|---|
| **0** | 0 | 1 | 10 |
| **1** | 1 | 1 | 11 |
| **2** | 2 | 1 | 12 |
| **3** | 3 | 1 | 13 |
| **4** | 4 | 1 | 14 |

# DataFrame a partir de Lista de Listas

```
[ ] import pandas as pd
    #sea el siguiente panel
    data = [["EDN.BA",26.20],["CVH.BA",458.50],["BMA.BA",233.95],["GGAL.BA",126.55]]

    df = pd.DataFrame(data)
    #df.columns = ["ticker","precio"]
    df.columns = ["ticker","precio"]
    df
```

| | ticker | precio |
|---|---|---|
| 0 | EDN.BA | 26.20 |
| 1 | CVH.BA | 458.50 |
| 2 | BMA.BA | 233.95 |
| 3 | GGAL.BA | 126.55 |

```
df = df.rename(columns={"ticker":"simbolo"})
```

```
[ ] df
```

| | simbolo | precio |
|---|---|---|
| 0 | EDN.BA | 26.20 |
| 1 | CVH.BA | 458.50 |
| 2 | BMA.BA | 233.95 |
| 3 | GGAL.BA | 126.55 |

# DataFrame a partir de un Diccionario

```
[ ] d = {'col1': [1, 2], 'col2': [3, 4]}
    df = pd.DataFrame(d)
    df
```

| | col1 | col2 |
|---|---|---|
| **0** | 1 | 3 |
| **1** | 2 | 4 |

```
[ ] df.dtypes
```

```
col1    int64
col2    int64
dtype: object
```

# DataFrame a partir de una lista de Diccionarios

```
[ ] import pandas as pd

    data = [
            {"ticker": "ALUA", 'Precio': 19.15, "Tipo": "Accion"},
            {"ticker": "BBAR", 'Precio': 73.7, "Tipo": "Accion"},
            {"ticker": "BMA", 'Precio': 144.4},
            {"ticker": "COME", 'Precio': 234, "Tipo": "COME"},
           ]
    tabla = pd.DataFrame(data)
    tabla
```

| | ticker | Precio | Tipo |
|---|---|---|---|
| 0 | ALUA | 19.15 | Accion |
| 1 | BBAR | 73.70 | Accion |
| 2 | BMA | 144.40 | NaN |
| 3 | COME | 234.00 | COME |

```
[ ] tabla.fillna("No se Sabe")
```

| | ticker | Precio | Tipo |
|---|---|---|---|
| 0 | ALUA | 19.15 | Accion |
| 1 | BBAR | 73.70 | Accion |
| 2 | BMA | 144.40 | No se Sabe |
| 3 | COME | 234.00 | COME |

# Librería Yfinance

- Para descargar esta librería vamos a usar el gestor de paquetes pip
- pip es un sistema de gestión de paquetes:
  - Es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software. Se utiliza para instalar y administra paquetes de software escritos en Python.
- Muchos paquetes pueden ser encontrados en el Python Package Index (PyPI).
- https://es.wikipedia.org/wiki/Pip_(administrador_de_paquetes)

```
[ ] #https://pypi.org/project/yfinance/
    !pip install yfinance
```

```
[ ] import yfinance as yf


    #df = yf.download("ggal")
    ticker_obj = yf.Ticker("ggal")

    ticker_obj.get_info()
```

# Leyendo y Guardando **DataFrame**

**Csv** **Excel** **Html**

```
[ ] pip install yfinance
```

```
[ ] import yfinance as yf

    df = yf.download( "ggal.ba")
    df
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2000-07-26** | 1.730000 | 1.750000 | 1.700000 | 1.730000 | 1.650303 | 422242 |
| **2000-07-27** | 1.750000 | 1.760000 | 1.730000 | 1.750000 | 1.669382 | 344044 |
| **2000-07-28** | 1.720000 | 1.740000 | 1.710000 | 1.720000 | 1.640764 | 561980 |
| **2000-07-31** | 1.750000 | 1.760000 | 1.690000 | 1.750000 | 1.669382 | 381111 |
| **2000-08-01** | 1.760000 | 1.860000 | 1.690000 | 1.760000 | 1.678921 | 1968000 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2021-11-15** | 255.000000 | 257.000000 | 235.600006 | 239.850006 | 239.850006 | 3039025 |
| **2021-11-16** | 240.000000 | 240.000000 | 221.750000 | 222.300003 | 222.300003 | 2982714 |
| **2021-11-17** | 223.500000 | 225.899994 | 208.250000 | 213.050003 | 213.050003 | 3114923 |
| **2021-11-18** | 211.050003 | 220.500000 | 208.000000 | 220.050003 | 220.050003 | 2274237 |
| **2021-11-19** | 220.899994 | 222.449997 | 208.000000 | 209.600006 | 209.600006 | 2232363 |

5323 rows × 6 columns

```
[ ] df.to_csv("ggalba.csv")
```

```
[ ] df.to_excel("ggalba.xlsx")
```

```
[ ] import pandas as pd

    pd.read_csv("ggalba.csv")
```

```
[ ] ### Lectura de Excels y CSVs de Internet

    data = pd.read_csv('https://datahub.io/core/s-and-p-500/r/data.csv')
    data
```

| | Date | SP500 | Dividend | Earnings | Consumer Price Index | Long Interest Rate | Real Price | Real Dividend | Real Earnings | PE10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1871-01-01 | 4.44 | 0.26 | 0.40 | 12.46 | 5.32 | 89.00 | 5.21 | 8.02 | NaN |
| **1** | 1871-02-01 | 4.50 | 0.26 | 0.40 | 12.84 | 5.32 | 87.53 | 5.06 | 7.78 | NaN |
| **2** | 1871-03-01 | 4.61 | 0.26 | 0.40 | 13.03 | 5.33 | 88.36 | 4.98 | 7.67 | NaN |
| **3** | 1871-04-01 | 4.74 | 0.26 | 0.40 | 12.56 | 5.33 | 94.29 | 5.17 | 7.96 | NaN |
| **4** | 1871-05-01 | 4.86 | 0.26 | 0.40 | 12.27 | 5.33 | 98.93 | 5.29 | 8.14 | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1763** | 2017-12-01 | 2664.34 | 48.93 | 109.88 | 246.52 | 2.40 | 2700.13 | 49.59 | 111.36 | 32.09 |
| **1764** | 2018-01-01 | 2789.80 | 49.29 | NaN | 247.87 | 2.58 | 2811.96 | 49.68 | NaN | 33.31 |
| **1765** | 2018-02-01 | 2705.16 | 49.64 | NaN | 248.99 | 2.86 | 2714.34 | 49.81 | NaN | 32.12 |
| **1766** | 2018-03-01 | 2702.77 | 50.00 | NaN | 249.55 | 2.84 | 2705.82 | 50.06 | NaN | 31.99 |
| **1767** | 2018-04-01 | 2642.19 | NaN | NaN | 249.84 | 2.80 | 2642.19 | NaN | NaN | 31.19 |

1768 rows × 10 columns

```
[ ] import pandas as pd

    data = pd.read_excel('https://covid.ourworldindata.org/data/owid-covid-data.xlsx')
    data.sort_values("total_cases_per_million", ascending=False)
```

| | iso_code | continent | location | date | total_cases | new_cases | new_cases_smoothed | total_deaths | new_deaths | new_deaths_smoothed | total_cases |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **82179** | MNE | Europe | Montenegro | 2021-11-21 | 154758.0 | 294.0 | 381.571 | 2245.0 | 6.0 | 5.429 | |
| **82178** | MNE | Europe | Montenegro | 2021-11-20 | 154464.0 | 336.0 | 390.571 | 2239.0 | 2.0 | 5.571 | |
| **82177** | MNE | Europe | Montenegro | 2021-11-19 | 154128.0 | 385.0 | 408.000 | 2237.0 | 6.0 | 6.143 | |
| **82176** | MNE | Europe | Montenegro | 2021-11-18 | 153743.0 | 409.0 | 421.429 | 2231.0 | 4.0 | 6.571 | |
| **82175** | MNE | Europe | Montenegro | 2021-11-17 | 153334.0 | 403.0 | 431.571 | 2227.0 | 10.0 | 6.571 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **132388** | WLF | Oceania | Wallis and Futuna | 2021-11-11 | NaN | NaN | NaN | NaN | NaN | NaN | |
| **132389** | WLF | Oceania | Wallis and Futuna | 2021-11-12 | NaN | NaN | NaN | NaN | NaN | NaN | |
| **132390** | WLF | Oceania | Wallis and Futuna | 2021-11-13 | NaN | NaN | NaN | NaN | NaN | NaN | |
| **132391** | WLF | Oceania | Wallis and Futuna | 2021-11-14 | NaN | NaN | NaN | NaN | NaN | NaN | |
| **132392** | WLF | Oceania | Wallis and Futuna | 2021-11-15 | NaN | NaN | NaN | NaN | NaN | NaN | |

134880 rows × 67 columns

```
[ ] df = pd.read_html("https://en.wikipedia.org/wiki/List_of_S%26P_500_companies")
    list(df[0]["Symbol"])
```

# Borrando
# **Columnas/Filas**

```
[ ]  ## Dropeando columnas

     data = pd.read_excel('ggalba.xlsx')
     copia = data.drop(['High', 'Low'], axis=1)
     copia.head()
```

```
[ ]  df = pd.read_excel("ggalba.xlsx")
     df.drop(4)
```

# Índices y **nombres de columnas**

```
[ ] df = yf.download("ggal.ba")
    df
```

```
[*********************100%***********************]  1 of 1 completed
```

| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **2000-07-26** | 1.730000 | 1.750000 | 1.700000 | 1.730000 | 1.650303 | 422242 |
| **2000-07-27** | 1.750000 | 1.760000 | 1.730000 | 1.750000 | 1.669382 | 344044 |
| **2000-07-28** | 1.720000 | 1.740000 | 1.710000 | 1.720000 | 1.640764 | 561980 |
| **2000-07-31** | 1.750000 | 1.760000 | 1.690000 | 1.750000 | 1.669382 | 381111 |
| **2000-08-01** | 1.760000 | 1.860000 | 1.690000 | 1.760000 | 1.678921 | 1968000 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2021-11-15** | 255.000000 | 257.000000 | 235.600006 | 239.850006 | 239.850006 | 3039025 |
| **2021-11-16** | 240.000000 | 240.000000 | 221.750000 | 222.300003 | 222.300003 | 2982714 |
| **2021-11-17** | 223.500000 | 225.899994 | 208.250000 | 213.050003 | 213.050003 | 3114923 |
| **2021-11-18** | 211.050003 | 220.500000 | 208.000000 | 220.050003 | 220.050003 | 2274237 |
| **2021-11-19** | 220.899994 | 222.449997 | 208.000000 | 209.600006 | 209.600006 | 2232363 |

5323 rows × 6 columns

```
[ ] df.columns = ["Open", "High", "Low",  "Close",  "Adj Close",  "Volume"]
```

# Índices y **nombres de columnas**

```
[ ] df
```

| | Adj Close | Open | High | Low | Close | Volume |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2000-07-26** | 1.730000 | 1.750000 | 1.700000 | 1.730000 | 1.650303 | 422242 |
| **2000-07-27** | 1.750000 | 1.760000 | 1.730000 | 1.750000 | 1.669382 | 344044 |
| **2000-07-28** | 1.720000 | 1.740000 | 1.710000 | 1.720000 | 1.640764 | 561980 |
| **2000-07-31** | 1.750000 | 1.760000 | 1.690000 | 1.750000 | 1.669382 | 381111 |
| **2000-08-01** | 1.760000 | 1.860000 | 1.690000 | 1.760000 | 1.678921 | 1968000 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2021-11-15** | 255.000000 | 257.000000 | 235.600006 | 239.850006 | 239.850006 | 3039025 |
| **2021-11-16** | 240.000000 | 240.000000 | 221.750000 | 222.300003 | 222.300003 | 2982714 |
| **2021-11-17** | 223.500000 | 225.899994 | 208.250000 | 213.050003 | 213.050003 | 3114923 |
| **2021-11-18** | 211.050003 | 220.500000 | 208.000000 | 220.050003 | 220.050003 | 2274237 |
| **2021-11-19** | 220.899994 | 222.449997 | 208.000000 | 209.600006 | 209.600006 | 2232363 |

5323 rows × 6 columns

# Índices

```
[ ] import pandas as pd
```

```
[ ] pd.DataFrame(range(5))
```

| | 0 |
|---|---|
| **0** | 0 |
| **1** | 1 |
| **2** | 2 |
| **3** | 3 |
| **4** | 4 |

```
[ ] pd.DataFrame(range(3), index=["a","b","c"])
```

| | 0 |
|---|---|
| **a** | 0 |
| **b** | 1 |
| **c** | 2 |

# Filtrado de **DataFrame**

- .loc
- .iloc

```
[ ] df.loc["2021-11-16":"2021-11-19", ["Open","Close"]]
```

| | Open | Close |
|---|---|---|
| **Date** | | |
| **2021-11-16** | 183.309998 | 183.369995 |
| **2021-11-17** | 183.369995 | 183.339996 |
| **2021-11-18** | 182.110001 | 178.770004 |
| **2021-11-19** | 179.440002 | 179.229996 |

```
[ ] df.iloc[::-1, [0,1]]
```

| | Open | High |
|---|---|---|
| **Date** | | |
| **2021-11-22** | 179.250000 | 182.229996 |
| **2021-11-19** | 179.440002 | 180.809998 |
| **2021-11-18** | 182.110001 | 182.539993 |
| **2021-11-17** | 183.369995 | 183.869995 |
| **2021-11-16** | 183.309998 | 184.199997 |
| **...** | ... | ... |
| **1970-01-08** | 7.000000 | 7.109375 |
| **1970-01-07** | 6.960938 | 7.015625 |
| **1970-01-06** | 6.890625 | 6.960938 |
| **1970-01-05** | 6.859375 | 6.898438 |
| **1970-01-02** | 6.851563 | 6.890625 |

13091 rows × 2 columns

# yFinance - Segunda Parte

```
[ ] help(yf.download)
```

```
Help on function download in module yfinance.multi:

download(tickers, start=None, end=None, actions=False, threads=True, group_by='column', auto_adjust=False, back_adjust=False, progress=True, period=
    Download yahoo tickers
    :Parameters:
        tickers : str, list
            List of tickers to download
        period : str
            Valid periods: 1d,5d,1mo,3mo,6mo,1y,2y,5y,10y,ytd,max
            Either Use period parameter or use start and end
        interval : str
            Valid intervals: 1m,2m,5m,15m,30m,60m,90m,1h,1d,5d,1wk,1mo,3mo
            Intraday data cannot extend last 60 days
        start: str
            Download start date string (YYYY-MM-DD) or _datetime.
            Default is 1900-01-01
        end: str
            Download end date string (YYYY-MM-DD) or _datetime.
            Default is now
        group_by : str
            Group by 'ticker' or 'column' (default)
        prepost : bool
            Include Pre and Post market data in results?
            Default is False
        auto_adjust: bool
            Adjust all OHLC automatically? Default is False
        actions: bool
            Download dividend + stock splits data. Default is False
        threads: bool / int
            How many threads to use for mass downloading. Default is True
        proxy: str
            Optional. Proxy server URL scheme. Default is None
        rounding: bool
            Optional. Round values to 2 decimal places?
        show_errors: bool
            Optional. Doesn't print errors if True
        timeout: None or float
            If not None stops waiting for a response after given number of
            seconds. (Can also be a fraction of a second e.g. 0.01)
```

# yFinance - Segunda Parte

```
[ ] yf.download("ggal", start="2021-11-01", interval="1h", rounding=True)
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **2021-11-01 09:30:00-04:00** | 10.67 | 11.14 | 10.55 | 10.81 | 10.81 | 135607 |
| **2021-11-01 10:30:00-04:00** | 10.82 | 10.87 | 10.70 | 10.79 | 10.79 | 120187 |
| **2021-11-01 11:30:00-04:00** | 10.78 | 10.84 | 10.67 | 10.67 | 10.67 | 92555 |
| **2021-11-01 12:30:00-04:00** | 10.67 | 10.75 | 10.60 | 10.75 | 10.75 | 70028 |
| **2021-11-01 13:30:00-04:00** | 10.73 | 10.83 | 10.72 | 10.74 | 10.74 | 65456 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2021-11-22 11:30:00-05:00** | 9.46 | 9.46 | 9.20 | 9.27 | 9.27 | 88579 |
| **2021-11-22 12:30:00-05:00** | 9.27 | 9.31 | 9.22 | 9.26 | 9.26 | 69076 |
| **2021-11-22 13:30:00-05:00** | 9.26 | 9.29 | 9.20 | 9.21 | 9.21 | 75794 |
| **2021-11-22 14:30:00-05:00** | 9.20 | 9.30 | 9.17 | 9.30 | 9.30 | 116813 |
| **2021-11-22 15:30:00-05:00** | 9.28 | 9.30 | 9.12 | 9.12 | 9.12 | 210981 |

112 rows × 6 columns

```
[ ] yf.download(["ggal", "ggal.ba"], auto_adjust=True)["Close"]
```

```
[*********************100%***********************]  2 of 2 completed
```

| Date | GGAL | GGAL.BA |
|---|---|---|
| **2000-07-25** | 16.033371 | NaN |
| **2000-07-26** | 16.033371 | 1.650303 |
| **2000-07-27** | 16.033371 | 1.669382 |
| **2000-07-28** | 15.918847 | 1.640764 |
| **2000-07-31** | 16.205154 | 1.669382 |
| **...** | ... | ... |
| **2021-11-16** | 10.500000 | 222.300003 |
| **2021-11-17** | 10.060000 | 213.050003 |
| **2021-11-18** | 10.190000 | 220.050003 |
| **2021-11-19** | 9.580000 | 209.600006 |
| **2021-11-22** | 9.150000 | NaN |

5519 rows × 2 columns

# Método **concat**

```
[ ]  ggal_adr = yf.download('GGAL', auto_adjust= True)['Close']
     ggal_bcba = yf.download('GGAL.BA', auto_adjust= True)['Close']
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

```
[ ]  # axis = 1 => Columnas
     # axis = 0 => Filas


     # join => inner  es una interseccion
     # join => outer es una union
```

```
[ ]  data = pd.concat([ggal_adr,ggal_bcba] , join='inner', axis=1)

     data
```

| | Close | Close |
|---|---|---|
| **Date** | | |
| **2000-07-26** | 16.033371 | 1.650303 |
| **2000-07-27** | 16.033371 | 1.669382 |
| **2000-07-28** | 15.918847 | 1.640764 |
| **2000-07-31** | 16.205158 | 1.669382 |
| **2000-08-01** | 16.033371 | 1.678921 |
| ... | ... | ... |
| **2021-11-15** | 11.390000 | 239.850006 |
| **2021-11-16** | 10.500000 | 222.300003 |
| **2021-11-17** | 10.060000 | 213.050003 |
| **2021-11-18** | 10.190000 | 220.050003 |
| **2021-11-19** | 9.580000 | 209.600006 |

5172 rows × 2 columns

# Ordenamiento sort_index y sort_values

```
[ ] import yfinance as yf

    df = yf.download("ggal")
```

```
[ ] df.sort_index(ascending=False)
```

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2021-11-22** | 9.720000 | 9.7200 | 9.1200 | 9.1500 | 9.150000 | 827400 |
| **2021-11-19** | 10.120000 | 10.2800 | 9.5500 | 9.5800 | 9.580000 | 1000700 |
| **2021-11-18** | 9.900000 | 10.3300 | 9.6900 | 10.1900 | 10.190000 | 784400 |
| **2021-11-17** | 10.640000 | 10.6400 | 9.8400 | 10.0600 | 10.060000 | 1354500 |
| **2021-11-16** | 11.300000 | 11.3800 | 10.4700 | 10.5000 | 10.500000 | 1272100 |
| ... | ... | ... | ... | ... | ... | ... |
| **2000-07-31** | 17.500000 | 17.6875 | 16.8750 | 17.6875 | 16.205153 | 178400 |
| **2000-07-28** | 17.562500 | 17.5625 | 17.1250 | 17.3750 | 15.918846 | 146100 |
| **2000-07-27** | 17.500000 | 17.6250 | 17.3750 | 17.5000 | 16.033367 | 61200 |
| **2000-07-26** | 17.250000 | 17.5625 | 17.1875 | 17.5000 | 16.033367 | 28900 |
| **2000-07-25** | 17.484375 | 17.7500 | 16.7500 | 17.5000 | 16.033367 | 126200 |

5368 rows × 6 columns

```
[ ] df.sort_values("Volume", ascending=False)
```

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2019-08-12** | 19.309999 | 19.360001 | 15.170000 | 16.7500 | 16.427402 | 30223700 |
| **2019-05-28** | 26.600000 | 27.010000 | 25.160000 | 25.7600 | 25.263872 | 15804000 |
| **2020-05-29** | 8.940000 | 8.940000 | 7.890000 | 8.0100 | 7.855731 | 14391500 |
| **2019-09-03** | 11.760000 | 11.850000 | 9.530000 | 9.5700 | 9.385685 | 6994500 |
| **2019-08-13** | 17.360001 | 18.020000 | 16.799999 | 17.2300 | 16.898157 | 6724700 |
| ... | ... | ... | ... | ... | ... | ... |
| **2000-12-07** | 14.187500 | 14.500000 | 14.187500 | 14.3125 | 13.113005 | 700 |
| **2002-09-17** | 0.700000 | 0.700000 | 0.700000 | 0.7000 | 0.653967 | 400 |
| **2002-08-19** | 0.700000 | 0.700000 | 0.660000 | 0.6600 | 0.616598 | 200 |
| **2002-08-14** | 0.660000 | 0.660000 | 0.660000 | 0.6600 | 0.616598 | 0 |
| **2002-09-30** | 0.640000 | 0.640000 | 0.640000 | 0.6400 | 0.597913 | 0 |

5368 rows × 6 columns

# shift()

```
[ ] data =  yf.download('AAPL')

    data['gap_nominal_pp'] = data['Open'] - data['Close'].shift()
    data['gap_porcentual_pp'] = (data['Open'] / data['Close'].shift() -1)*100

    data
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume | gap_nominal_pp | gap_porcentual_pp |
|---|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | | |
| **1980-12-12** | 0.128348 | 0.128906 | 0.128348 | 0.128348 | 0.100922 | 469033600 | NaN | NaN |
| **1980-12-15** | 0.122210 | 0.122210 | 0.121652 | 0.121652 | 0.095657 | 175884800 | -0.006138 | -4.782303 |
| **1980-12-16** | 0.113281 | 0.113281 | 0.112723 | 0.112723 | 0.088636 | 105728000 | -0.008371 | -6.881106 |
| **1980-12-17** | 0.115513 | 0.116071 | 0.115513 | 0.115513 | 0.090830 | 86441600 | 0.002790 | 2.475091 |
| **1980-12-18** | 0.118862 | 0.119420 | 0.118862 | 0.118862 | 0.093463 | 73449600 | 0.003349 | 2.899246 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2021-04-14** | 134.940002 | 135.000000 | 131.660004 | 132.029999 | 132.029999 | 87222800 | 0.510010 | 0.379387 |
| **2021-04-15** | 133.820007 | 135.000000 | 133.639999 | 134.500000 | 134.500000 | 89347100 | 1.790009 | 1.355759 |
| **2021-04-16** | 134.300003 | 134.669998 | 133.279999 | 134.160004 | 134.160004 | 84818500 | -0.199997 | -0.148697 |
| **2021-04-19** | 133.509995 | 135.470001 | 133.339996 | 134.839996 | 134.839996 | 93996100 | -0.650009 | -0.484503 |
| **2021-04-20** | 135.020004 | 135.529999 | 131.811493 | 133.110001 | 133.110001 | 93002207 | 0.180008 | 0.133497 |

10173 rows × 8 columns

# rolling()

```
[ ] data =  yf.download('ypfd.ba')

    data['sma_12'] = data['Adj Close'].rolling(12).mean()

    data
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume | sma_12 |
|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | |
| **2000-01-03** | 36.099998 | 36.549999 | 36.049999 | 36.099998 | 24.355095 | 24993 | NaN |
| **2000-01-04** | 36.099998 | 36.099998 | 35.900002 | 36.099998 | 24.355095 | 9896 | NaN |
| **2000-01-05** | 36.200001 | 36.200001 | 36.099998 | 36.200001 | 24.422565 | 2823 | NaN |
| **2000-01-06** | 35.849998 | 36.099998 | 35.849998 | 35.849998 | 24.186432 | 2285 | NaN |
| **2000-01-07** | 36.450001 | 36.450001 | 36.000000 | 36.450001 | 24.591225 | 4738 | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... |
| **2021-04-14** | 597.950012 | 614.250000 | 586.049988 | 588.200012 | 588.200012 | 128071 | 606.266668 |
| **2021-04-15** | 585.950012 | 597.700012 | 584.000000 | 586.000000 | 586.000000 | 54069 | 605.425003 |
| **2021-04-16** | 583.000000 | 593.950012 | 570.000000 | 591.299988 | 591.299988 | 152021 | 605.420837 |
| **2021-04-19** | 594.000000 | 604.250000 | 588.000000 | 590.299988 | 590.299988 | 120653 | 604.829168 |
| **2021-04-20** | 588.500000 | 588.500000 | 569.000000 | 573.500000 | 573.500000 | 165613 | 602.779170 |

5324 rows × 7 columns

```
[ ] data["sma_3"] = data.Close.rolling(3).mean()
```

```
[ ] data
```

| | Open | High | Low | Close | Volume | sma_3 |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2000-01-03** | 24.355099 | 24.658695 | 24.321366 | 24.355099 | 24993 | NaN |
| **2000-01-04** | 24.355099 | 24.355099 | 24.220169 | 24.355099 | 9896 | NaN |
| **2000-01-05** | 24.422562 | 24.422562 | 24.355094 | 24.422562 | 2823 | 24.377586 |
| **2000-01-06** | 24.186430 | 24.355094 | 24.186430 | 24.186430 | 2285 | 24.321363 |
| **2000-01-07** | 24.591228 | 24.591228 | 24.287633 | 24.591228 | 4738 | 24.400073 |
| ... | ... | ... | ... | ... | ... | ... |
| **2021-11-15** | 959.000000 | 960.000000 | 906.000000 | 909.750000 | 316880 | 930.800008 |
| **2021-11-16** | 910.000000 | 918.000000 | 867.000000 | 871.250000 | 289134 | 909.666667 |
| **2021-11-17** | 872.500000 | 888.000000 | 837.200012 | 849.349976 | 377671 | 876.783325 |
| **2021-11-18** | 855.000000 | 895.000000 | 835.049988 | 887.500000 | 266878 | 869.366659 |
| **2021-11-19** | 889.000000 | 892.000000 | 838.049988 | 842.250000 | 271858 | 859.699992 |

5470 rows × 6 columns

# Funciones ponderadas ewm()

```
[ ] data =  yf.download('ypfd.ba')
    data['ema_12'] = data['Adj Close'].ewm(span = 12).mean()
    data
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume | ema_12 |
|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | |
| **2000-01-03** | 36.099998 | 36.549999 | 36.049999 | 36.099998 | 24.262033 | 24993 | 24.262033 |
| **2000-01-04** | 36.099998 | 36.099998 | 35.900002 | 36.099998 | 24.262033 | 9896 | 24.262033 |
| **2000-01-05** | 36.200001 | 36.200001 | 36.099998 | 36.200001 | 24.329248 | 2823 | 24.288267 |
| **2000-01-06** | 35.849998 | 36.099998 | 35.849998 | 35.849998 | 24.094019 | 2285 | 24.226951 |
| **2000-01-07** | 36.450001 | 36.450001 | 36.000000 | 36.450001 | 24.497265 | 4738 | 24.300394 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2020-12-11** | 772.000000 | 778.450012 | 762.299988 | 774.849976 | 774.849976 | 79780 | 759.363344 |
| **2020-12-14** | 776.150024 | 789.000000 | 767.150024 | 779.599976 | 779.599976 | 119212 | 762.476672 |
| **2020-12-15** | 779.000000 | 785.000000 | 765.150024 | 771.650024 | 771.650024 | 89133 | 763.887957 |
| **2020-12-16** | 771.000000 | 797.950012 | 770.049988 | 778.799988 | 778.799988 | 147455 | 766.182116 |
| **2020-12-17** | 778.799988 | 798.000000 | 769.250000 | 771.599976 | 771.599976 | 116369 | 767.015632 |

5218 rows × 7 columns

# diff()

```
[ ] data =  yf.download('ypfd.ba', interval='1d')

    data['variacion_diaria_nominal'] = data['Adj Close'].diff()
    data['variacion_fw_10_nominal'] = data['Adj Close'].diff(-10)

    data.dropna().round(2)
```

```
[*********************100%***********************]  1 of 1 completed
```

| | Open | High | Low | Close | Adj Close | Volume | variacion_diaria_nominal | variacion_fw_10_nominal |
|---|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | | |
| **2000-01-04** | 36.10 | 36.10 | 35.90 | 36.10 | 24.26 | 9896 | 0.00 | 0.07 |
| **2000-01-05** | 36.20 | 36.20 | 36.10 | 36.20 | 24.33 | 2823 | 0.07 | -0.54 |
| **2000-01-06** | 35.85 | 36.10 | 35.85 | 35.85 | 24.09 | 2285 | -0.24 | -0.77 |
| **2000-01-07** | 36.45 | 36.45 | 36.00 | 36.45 | 24.50 | 4738 | 0.40 | -0.37 |
| **2000-01-10** | 36.45 | 36.45 | 35.90 | 36.45 | 24.50 | 812 | 0.00 | 0.27 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2020-11-25** | 800.00 | 815.00 | 770.00 | 771.50 | 771.50 | 273431 | -32.05 | -3.35 |
| **2020-11-26** | 760.55 | 790.00 | 760.00 | 779.85 | 779.85 | 45129 | 8.35 | 0.25 |
| **2020-11-27** | 769.05 | 823.00 | 769.05 | 812.65 | 812.65 | 166248 | 32.80 | 41.00 |
| **2020-11-30** | 810.05 | 812.00 | 750.00 | 765.40 | 765.40 | 236453 | -47.25 | -13.40 |
| **2020-12-01** | 780.00 | 797.90 | 766.00 | 788.40 | 788.40 | 191342 | 23.00 | 16.80 |

5207 rows × 8 columns

# pct_change()

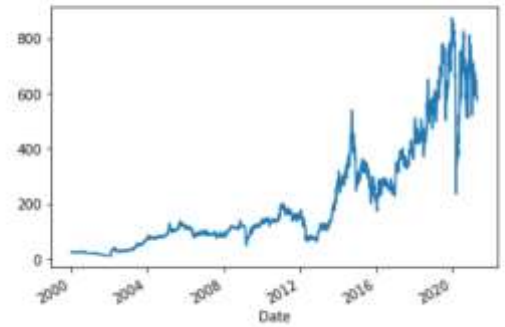
- Calcula el cambio porcentual de la fila inmediatamente anterior de forma predeterminada.
- Es útil para comparar el porcentaje de cambio en una serie temporal de elementos.

```
[ ] df = yf.download("ggal", auto_adjust=True)

    [*********************100%***********************]  1 of 1 completed

[ ] df["rendimiento"] = df.Close.pct_change() * 100

[ ] df
```

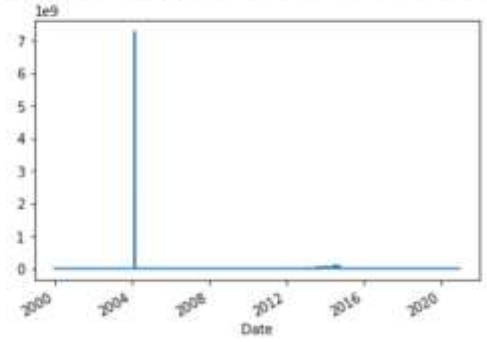
| | Open | High | Low | Close | Volume | rendimiento |
|---|---|---|---|---|---|---|
| Date | | | | | | |
| **2000-07-25** | 16.019054 | 16.262417 | 15.346225 | 16.033369 | 126200 | NaN |
| **2000-07-26** | 15.804321 | 16.090631 | 15.747059 | 16.033369 | 28900 | 0.000000 |
| **2000-07-27** | 16.033369 | 16.147893 | 15.918845 | 16.033369 | 61200 | 0.000000 |
| **2000-07-28** | 16.090633 | 16.090633 | 15.689799 | 15.918847 | 146100 | -0.714273 |
| **2000-07-31** | 16.033366 | 16.205153 | 15.460746 | 16.205153 | 178400 | 1.798531 |
| ... | ... | ... | ... | ... | ... | ... |
| **2021-11-16** | 11.300000 | 11.380000 | 10.470000 | 10.500000 | 1272100 | -7.813875 |
| **2021-11-17** | 10.640000 | 10.640000 | 9.840000 | 10.060000 | 1354500 | -4.190472 |
| **2021-11-18** | 9.900000 | 10.330000 | 9.690000 | 10.190000 | 784400 | 1.292238 |
| **2021-11-19** | 10.120000 | 10.280000 | 9.550000 | 9.580000 | 1000700 | -5.986258 |
| **2021-11-22** | 9.720000 | 9.720000 | 9.120000 | 9.150000 | 825508 | -4.488521 |

5368 rows × 6 columns

```
[ ] data["Adj Close"].plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7fec6282a850>
```




```
[ ] data["Volume"].plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7fecd2209b00>
```



# Funciones como Parámetros

```
[ ] def descontar_comision(monto):
      return monto * 0.99

    def descontar_iva(monto):
      return monto * 0.79

    def descontar_otros(montos):
      return monto * 0.98

    funcion_de_costos = [ descontar_comision, descontar_otros]

    type(funcion_de_costos[0])
```

```
function
```

```
monto = 1000
for funcion in funcion_de_costos:
  monto = funcion(monto)
monto
```

```
970.1999999999999
```

```
[ ] ## funciones anonimas
    sumar = lambda x, y: x + y

    restar = lambda x, y : x - y
```

```
[ ] sumar (1,2)
```

```
3
```

```
[ ] restar(1,2)
```

```
-1
```

# Agrupamientos - groupby()

```
[ ] import yfinance as yf
    import numpy as np

    df = yf.download("MMM", auto_adjust=True)

    [*********************100%***********************]  1 of 1 completed

[ ]

[ ]
    df["vela"] = np.where( df.Open > df.Close, "roja", np.where(df.Open == df.Close, "doji", "verde" )  )

[ ] df.groupby("vela").count()["Close"]

    vela
    doji     1397
    roja     1996
    verde    1930
    Name: Close, dtype: int64

[ ] # cantidad de ruedas por año

    data = df.Open.groupby(df.index.year).count().to_frame()

    data.columns = ["nro ruedas"]
```

# Funciones Acumulativas

```
[ ] df = yf.download("MSFT", auto_adjust=True)
```

```
[*********************100%***********************]  1 of 1 completed
```

```
[ ] df["maximo_historico"] = df.High.cummax()
    df
```

| | Open | High | Low | Close | Volume | maximo_historico |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **1986-03-13** | 0.055898 | 0.064119 | 0.055898 | 0.061378 | 1031788800 | 0.064119 |
| **1986-03-14** | 0.061378 | 0.064667 | 0.061378 | 0.063570 | 308160000 | 0.064667 |
| **1986-03-17** | 0.063570 | 0.065215 | 0.063570 | 0.064667 | 133171200 | 0.065215 |
| **1986-03-18** | 0.064667 | 0.065215 | 0.062474 | 0.063022 | 67766400 | 0.065215 |
| **1986-03-19** | 0.063022 | 0.063570 | 0.061378 | 0.061926 | 47894400 | 0.065215 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2021-11-16** | 335.066992 | 340.047900 | 334.897319 | 338.890015 | 20886800 | 340.047900 |
| **2021-11-17** | 338.940002 | 342.190002 | 338.000000 | 339.119995 | 19053400 | 342.190002 |
| **2021-11-18** | 338.179993 | 342.450012 | 337.119995 | 341.269989 | 22463500 | 342.450012 |
| **2021-11-19** | 342.640015 | 345.100006 | 342.200012 | 343.109985 | 21942200 | 345.100006 |
| **2021-11-22** | 344.619995 | 349.670013 | 339.549988 | 339.829987 | 31004600 | 349.670013 |

8999 rows × 6 columns

Academy