pandas

Bienvenidos al mundo de Pandas

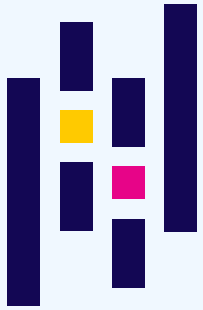
Tu portal al análisis de datos con Python.

Ideal para comenzar tu viaje en el mundo de Data Science

```
import pandas as pd
df = pd.DataFrame({'datos': [1, 2, 3]})
print(df)
```

Powered by python

¿De qué vamos a hablar hoy?



1. ¿Qué es Pandas?
2. Instalación
3. Primeros pasos: Series y DataFrames
4. Operaciones básicas
5. Manipulación de datos
6. Ejemplos prácticos
7. Recomendaciones y recursos

```
import pandas as pd  # ¡Comencemos nuestra aventura con Pandas!
```

¿Qué es Pandas?

Pandas es una librería de Python de código abierto que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar.

Estructuras de datos principales

Series (1D) y DataFrame (2D) para manipular datos tabulares.

Fundamental en Data Science

Es la base de muchos proyectos de análisis de datos, Machine Learning e Inteligencia Artificial.

Funcionalidades principales

Importar/exportar datos, limpiar, transformar, analizar y visualizar información de múltiples fuentes.

```python
import pandas as pd

# Crear un DataFrame simple
datos = {
    'Producto': ['A', 'B', 'C'],
    'Ventas': [150, 200, 120]
}

df = pd.DataFrame(datos)
print(df)
```

Resultado:

	Producto	Ventas
0	A	150
1	B	200
2	C	120

¿Por qué usar Pandas?

Pandas te permite manejar datos de manera eficiente, similar a Excel pero con el poder de Python:

Fácil importación de datos

Importa datos desde CSV, Excel, SQL, JSON y muchos otros formatos con una sola línea de código.

Limpieza de datos simplificada

Detecta y maneja valores faltantes, duplicados o incorrectos rápidamente.

Filtrado y transformación

Filtra, agrupa y transforma grandes volúmenes de datos con operaciones intuitivas.

Análisis estadístico

Genera estadísticas descriptivas y visualizaciones de datos con facilidad.

```python
import pandas as pd

# 1. Importar datos desde CSV
ventas = pd.read_csv('ventas.csv')

# 2. Limpiar datos
ventas_limpias = ventas.dropna()

# 3. Filtrar datos
ventas_altas = ventas[ventas['monto'] > 1000]

# 4. Análisis estadístico
resumen = ventas.describe()

# 5. Agrupar y agregar
por_region = ventas.groupby('region')['monto'].sum()
```

Ventaja sobre Excel:

- Procesa millones de filas (Excel se limita a ~1 millón)
- Automatiza tareas repetitivas con código
- Flujos de trabajo reproducibles y documentados
- Integración perfecta con otras librerías de Python

Cómo instalar Pandas

Instalar Pandas es muy sencillo. Si ya tienes Python, solo necesitas ejecutar un comando en la terminal:

Terminal

```
$ pip install pandas
Collecting pandas
Downloading pandas-2.0.3-cp310-cp310-win_amd64.whl (10.5 MB)
Installing collected packages: pandas
Successfully installed pandas-2.0.3
```

Para verificar que la instalación fue exitosa, puedes importar Pandas en Python:

Python

```
>>> import pandas as pd
>>> print(pd.__version__)
2.0.3
```

Dependencias de Pandas

Al instalar Pandas, se instalarán automáticamente estas dependencias:

- NumPy: para operaciones numéricas
- Python-dateutil: manejo de fechas
- Pytz: zonas horarias

Otras formas de instalación

Con Anaconda (recomendado para Data Science):

```
conda install pandas
```

Desde GitHub (versión de desarrollo):

```
pip install git+https://github.com/pandas-dev/pandas.git
```

Importando Pandas

Siempre que trabajes con Pandas, el primer paso es importar la librería al inicio de tu código:

Importación estándar

Utiliza esta convención para importar Pandas en todos tus proyectos:

```
import pandas as pd
```

¿Por qué `as pd`?

- Es un **alias corto** que facilita escribir código y es el estándar en la comunidad
- Evita tener que escribir `pandas` completo cada vez que uses una función

```
import pandas as pd

# Ahora puedes usar pd en lugar de pandas
datos = {
    'Nombre': ['Ana', 'Luis'],
    'Edad': [25, 30]
}

df = pd.DataFrame(datos)
print(df)
```

⚠ **Importante:** Nunca olvides importar Pandas al inicio de tu código. El alias `pd` es el estándar en la comunidad y mejora la legibilidad.

Series: Tu primer objeto en Pandas

Una Series es una estructura de datos unidimensional (1D) que puede contener cualquier tipo de datos.

¿Qué es una Series?

Es similar a una columna en Excel o una lista de Python, pero con funcionalidades adicionales para análisis de datos.

Componentes clave

Datos: Los valores almacenados

Índice: Etiquetas para acceder a los valores

```python
import pandas as pd

# Crear una Series con valores numéricos
serie = pd.Series([10, 20, 30, 40])

print(serie)
```

Resultado:

```
0    10
1    20
2    30
3    40
dtype: int64
```

DataFrames: El corazón de Pandas

Un **DataFrame** es una estructura de datos tabular bidimensional (2D), similar a:

- Una hoja de cálculo de Excel
- Una tabla de SQL
- Una colección de Series que comparten un índice

Características principales:

- Tiene filas y columnas etiquetadas
- Puede contener diferentes tipos de datos
- Permite realizar operaciones aritméticas por columna
- Es mutable (se puede modificar)

```python
import pandas as pd

# Crear un DataFrame con datos de personas
datos = {
    'Nombre': ['Ana', 'Luis', 'Sofía'],
    'Edad': [22, 35, 29]
}

df = pd.DataFrame(datos)
print(df)
```

Resultado:

	Nombre	Edad
0	Ana	22
1	Luis	35
2	Sofía	29

Leyendo datos desde un archivo

Pandas facilita la lectura de datos desde diferentes formatos de archivo, especialmente CSV.

Leyendo archivos CSV

El formato más común para datos tabulares. Pandas lo lee con una sola línea de código.

Otros formatos soportados

Excel: pd.read_excel()
JSON: pd.read_json()
SQL: pd.read_sql()
HTML: pd.read_html()

```python
import pandas as pd

# Leer un archivo CSV
df = pd.read_csv('datos_ventas.csv')

# Ver las primeras 5 filas para explorar los datos
print(df.head())
```

Resultado de df.head():

#	Fecha	Producto	Cantidad	Precio
0	2023-01-15	Laptop	2	1200.50
1	2023-01-16	Mouse	5	25.99
2	2023-01-16	Teclado	3	45.75

Tip: Usa `df.info()` para ver tipos de datos y `df.describe()` para estadísticas básicas.

Operaciones básicas con DataFrames

Los DataFrames de Pandas ofrecen numerosas operaciones que facilitan el acceso y análisis de tus datos:

Acceso a columnas

Usa la notación `df['nombre_columna']` para extraer una columna completa como Series.

Acceso a filas

Usa `df.loc[índice]` para extraer una fila completa por su índice.

Resumen estadístico

`df.describe()` genera automáticamente estadísticas básicas para todas las columnas numéricas.

DataFrame de ejemplo:

```
import pandas as pd

datos = {
    'Nombre': ['Ana', 'Luis', 'Sofía'],
    'Edad': [22, 35, 29],
    'Ciudad': ['Madrid', 'Barcelona', 'Sevilla']
}

df = pd.DataFrame(datos)
print(df)
```

	Nombre	Edad	Ciudad
0	Ana	22	Madrid
1	Luis	35	Barcelona
2	Sofía	29	Sevilla

Columnas | Filas | Estadísticas

```
# Acceder a la columna 'Nombre'
df['Nombre']

# Resultado:
0    Ana
1    Luis
2    Sofía
Name: Nombre, dtype: object

# Acceder a la columna 'Edad'
df['Edad']
```

DataFrame

Filtrado de datos

Pandas permite **filtrar filas** que cumplen una condición específica usando una sintaxis simple y poderosa.

¿Cómo funciona?

1. Crea una condición lógica en una columna
2. Aplica la condición dentro de corchetes `df[condición]`
3. Obtienes solo las filas que cumplen la condición

Otros filtros comunes:

- `df[df['Columna'] == valor]`
- `df[df['Columna'].isin([val1, val2])]`
- `df[(condición1) & (condición2)]`

filter

```python
# Definimos un DataFrame de ejemplo
datos = {
    'Nombre': ['Ana', 'Luis', 'Sofía', 'Carlos'],
    'Edad': [22, 35, 29, 42]
}
df = pd.DataFrame(datos)

# Filtramos personas mayores de 25 años
mayores_25 = df[df['Edad'] > 25]
print(mayores_25)
```

DataFrame original (df):

	Nombre	Edad
0	Ana	22
1	Luis	35
2	Sofía	29
3	Carlos	42

↓ df[df['Edad'] > 25] ↓

Resultado (mayores_25):

	Nombre	Edad
1	Luis	35
2	Sofía	29
3	Carlos	42

Pandas ofrece diversas funciones para manipular fácilmente los datos de un DataFrame:

Agregar columnas

Crea nuevas columnas basadas en datos existentes o nuevos valores.

Modificar valores

Actualiza valores existentes usando condiciones o funciones.

Eliminar datos

Quita columnas o filas innecesarias usando `drop`.

DataFrame inicial:

```
import pandas as pd

datos = {
    'Nombre': ['Ana', 'Luis', 'Sofía'],
    'Edad': [17, 35, 29]
}

df = pd.DataFrame(datos)
print(df)
```

	Nombre	Edad
0	Ana	17
1	Luis	35
2	Sofía	29

1. Agregar una nueva columna:

```
# Crear columna basada en una condición
df['MayorDeEdad'] = df['Edad'] >= 18
print(df)
```

	Nombre	Edad	MayorDeEdad
0	Ana	17	False
1	Luis	35	True
2	Sofía	29	True

2. Eliminar una columna:

```
# Eliminar columna (axis=1 indica columnas)
df_nuevo = df.drop('Nombre', axis=1)
print(df_nuevo)
```

	Edad	MayorDeEdad
0	17	False
1	35	True

Limpieza de datos

Los datos del mundo real raramente están completos. Pandas ofrece herramientas para identificar y manejar valores faltantes (NaN, None, NULL).

Identificar datos faltantes

`df.isnull()` devuelve un DataFrame de booleanos (True donde faltan datos).

Eliminar filas o columnas

`df.dropna()` elimina filas con valores faltantes. Usa `axis=1` para columnas.

Rellenar valores faltantes

`df.fillna(valor)` reemplaza NaN con el valor especificado.

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

# Crear DataFrame con valores faltantes
datos = {
    'A': [1, 2, np.nan, 4],
    'B': [5, np.nan, np.nan, 8],
    'C': [9, 10, 11, 12]
}

df = pd.DataFrame(datos)
print(df)
```

df.isnull()

	A	B	C
0	False	False	False
1	False	True	False
2	True	True	False
3	False	False	False

DataFrame original

	A	B	C
0	1	5	9
1	2	*NaN*	10
2	*NaN*	*NaN*	11
3	4	8	12

df.fillna(0)

	A	B	C
0	1	5	9
1	2	**0**	10
2	**0**	**0**	11
3	4	8	12

df.dropna()

	A	B	C
0	1	5	9
3	4	8	12

Filas 1 y 2 eliminadas por tener NaN

Ejemplo práctico: Analizando un CSV real

Pasos del ejemplo

1. Leer un archivo CSV
2. Explorar los datos
3. Realizar filtros y estadísticas
4. Limpiar información

Sobre el dataset

ventas_tienda.csv

Contiene datos de ventas de una tienda con información sobre productos, fechas, precios y cantidades.

```python
import pandas as pd

# 1. Leer el archivo CSV
df = pd.read_csv('ventas_tienda.csv')

# 2. Explorar los datos
# Primeras 5 filas
print(df.head())

# Información sobre las columnas
print(df.info())

# Estadísticas básicas
print(df.describe())

# 3. Realizar filtros y estadísticas
# Filtrar productos con precio mayor a $50
productos_caros = df[df['precio'] > 50]
```

Resultado de ventas por categoría:

Categoría	Total Vendido
Electrónica	285
Ropa	162
Hogar	94

Resumen y recursos para seguir aprendiendo

Pandas es una herramienta fundamental para Data Science.
¡La práctica es clave para dominarla!

Documentación Oficial

La mejor referencia para aprender todas las funcionalidades de Pandas en detalle.

pandas.pydata.org/docs

Tutoriales en YouTube

Canales recomendados con tutoriales paso a paso de Pandas para todos los niveles.

- Corey Schafer
- freeCodeCamp
- Data School

Ejercicios Prácticos

Practica con datasets reales y proyectos guiados para reforzar tu aprendizaje.

- Kaggle Learn
- GitHub - pandas-exercises
- DataCamp

Consejo final:

La mejor forma de aprender Pandas es trabajando con datos reales. Busca datasets que te interesen en Kaggle o data.gov y comienza a explorarlos.

¡A programar!