SESIÓN LA LIBRERÍA PANDAS

CONTENIDOS:

- Reseña de la librería Pandas.
    - Para qué se utiliza.
- El tipo de dato serie:
    - Características del tipo de dato serie.
    - Creación de una serie
    - Obtención de datos de una serie
    - Operaciones sobre una serie.
- El tipo de dato dataframe:
    - Características del tipo de dato dataframe.
    - Creación de un dataframe.
- Selección de filas o columnas en un dataframe.
- Selección de elementos en un dataframe.
- Selección condicional en un dataframe.
- Métodos básicos de exploración (head, tail, info, describe).
- Métodos básicos de sumarización (min, max, count, mean, median, sum).
- Métodos unique, nunique, value_counts.

RESEÑA DE LA LIBRERÍA PANDAS

Pandas es una de las librerías más populares y poderosas de Python para la manipulación y análisis de datos. Desarrollada por Wes McKinney en 2008, Pandas se ha convertido en una herramienta esencial para científicos de datos, analistas e ingenieros debido a su facilidad de uso y su capacidad para manejar grandes volúmenes de datos de manera eficiente.

¿Qué es Pandas?

Pandas es una librería de código abierto construida sobre NumPy, diseñada para proporcionar estructuras de datos flexibles y eficientes que permiten trabajar con datos tabulares, estructurados y series temporales. Su nombre proviene de "Panel Data", un término econométrico que hace referencia a conjuntos de datos multidimensionales.

Características principales

- Manejo eficiente de datos tabulares en estructuras optimizadas como DataFrame y Series.
- Alta compatibilidad con librerías como NumPy, Matplotlib y Scikit-learn.
- Facilidad de lectura y escritura de múltiples formatos de datos como CSV, JSON, Excel y SQL.
- Potentes herramientas de manipulación, como filtrado, agrupación, transformación y combinación de datos.
- Optimización de rendimiento mediante operaciones vectorizadas basadas en NumPy y el uso eficiente de memoria.

¿Para qué se utiliza?

Pandas es una herramienta versátil utilizada en múltiples áreas de la ciencia de datos, análisis y procesamiento de datos. Sus principales usos incluyen:

- **Manipulación y limpieza de datos:** Permite manejar datos faltantes, eliminar duplicados, transformar columnas y normalizar datos de manera eficiente.
- **Análisis exploratorio de datos (EDA):** Facilita la inspección y descripción de los datos a través de estadísticas básicas, visualizaciones y resúmenes rápidos.
- **Agregación y agrupación:** Proporciona métodos avanzados para resumir y analizar datos mediante operaciones de agrupamiento (groupby), pivot tables y cálculos personalizados.
- **Manejo de series temporales:** Pandas tiene funciones especializadas para trabajar con fechas y datos temporales, facilitando operaciones como indexación por tiempo y resampling.
- **Integración con otras herramientas:** Se integra con librerías como NumPy para cálculos numéricos, Matplotlib y Seaborn para visualización de datos, y Scikit-learn para machine learning.
- **Carga y almacenamiento de datos:** Permite importar y exportar datos en múltiples formatos como CSV, Excel, JSON, HDF5, SQL, Parquet, entre otros.

Ilustración 1 Logo pandas

EL TIPO DE DATO SERIE

Una Serie en Pandas es una estructura de datos unidimensional, similar a un vector en NumPy o una lista en Python, pero con índices personalizables. Cada elemento en una Serie está asociado con una etiqueta (índice), lo que permite acceder y manipular los datos de manera eficiente.

Características del Tipo de Dato Serie

- Es una estructura de datos unidimensional, lo que significa que contiene una sola columna de valores.
- Cada elemento en una Serie tiene un índice asociado, que puede ser numérico o de tipo string.
- Soporta operaciones matemáticas y funciones estadísticas aplicadas a todos los elementos de manera eficiente.
- Se basa en NumPy, lo que le otorga un alto rendimiento en cálculos numéricos.
- Es altamente flexible, permitiendo la conversión entre listas, diccionarios y arreglos de NumPy.

Creación de una Serie

Ejemplo 1: Crear una Serie desde una Lista

```
import pandas as pd
datos = [10, 20, 30, 40]
serie = pd.Series(datos, index=['a', 'b', 'c', 'd'])
print(serie)
```

Ilustración 2 Ejemplo crear una serie desde una lista

Ejemplo 2: Crear una Serie con Índices Personalizados

```
serie = pd.Series([100, 200, 300], index=["a", "b", "c"])
print(serie)
```

Ilustración 3 Ejemplo crear una serie con índices personalizados

Ejemplo 3: Crear una Serie desde un Diccionario

```
datos = {"Manzanas": 5, "Peras": 3, "Naranjas": 8}
serie = pd.Series(datos)
print(serie)
```

Ilustración 4 Ejemplo crear una serie desde un diccionario

Obtención de Datos de una Serie

- Acceso por índice: Podemos acceder a los valores de una Serie de manera similar a los diccionarios en Python.

```
print(serie["Peras"])  # Devuelve 3
print(serie[0])        # Devuelve 5 (accediendo por índice numérico)
```

Ilustración 5 Ejemplo de acceso por índice

- Acceso a Múltiples Elementos (Slicing)

```
print(serie[["Manzanas", "Naranjas"]])  # Devuelve los valores de las claves seleccionadas
```

Ilustración 6 Ejemplo acceso a múltiples elementos

- Filtrado de Datos en una Serie
  Podemos aplicar condiciones para seleccionar elementos específicos.

```
print(serie[serie > 4])  # Devuelve solo los valores mayores a 4
```

Ilustración 7 Ejemplo filtrado de datos en una serie

Operaciones sobre una Serie

Pandas permite realizar operaciones matemáticas sobre los elementos de una Serie de forma rápida y eficiente.

Ejemplo: Operaciones Matemáticas en una Serie

```
numeros = pd.Series([1, 2, 3, 4, 5])

print(numeros * 2)  # Multiplicación por escalar
print(numeros + 10) # Suma de un valor a cada elemento
print(numeros.mean())  # Media de los valores
print(numeros.max())   # Valor máximo
print(numeros.min())   # Valor mínimo
```

Ilustración 8 Ejemplo operaciones matemáticas en una serie

EL TIPO DE DATO DATAFRAME

Un DataFrame en Pandas es una estructura de datos bidimensional, similar a una tabla de una base de datos o una hoja de cálculo de Excel. Se compone de filas y columnas, donde cada columna puede contener diferentes tipos de datos.

Características del Tipo de Dato DataFrame:

- Permite almacenar y manipular datos estructurados en forma de tabla.
- Cada columna representa una Serie de Pandas, por lo que pueden contener distintos tipos de datos.
- Es altamente eficiente y optimizado para grandes volúmenes de datos.
- Soporta indexación avanzada y selección de datos mediante etiquetas o índices numéricos.
- Facilita la importación y exportación de datos desde archivos CSV, Excel, JSON y bases de datos SQL.
- Compatible con NumPy y otras librerías de ciencia de datos.

Creación de un DataFrame

Podemos crear un DataFrame desde múltiples fuentes, como listas, diccionarios o arreglos de NumPy.

Ejemplo 1: Crear un DataFrame desde un Diccionario

```
datos = {
    'Nombre': ['Juan', 'Ana', 'Luis'],
    'Edad': [25, 30, 35],
    'Ciudad': ['Madrid', 'Barcelona', 'Valencia']
}
df = pd.DataFrame(datos)
print(df)
```

Ilustración 9 Ejemplo Crear un DataFrame desde un diccionario

SELECCIÓN DE FILAS O COLUMNAS EN UN DATAFRAME.

Pandas proporciona herramientas poderosas para seleccionar y manipular datos dentro de un DataFrame, lo que facilita la exploración y análisis de datos tabulares. A continuación, exploramos las formas más eficientes de seleccionar filas, columnas y elementos específicos, así como métodos clave de exploración y agregación de datos.

- **Selección de Columnas**

  Podemos acceder a una columna de un DataFrame de diferentes maneras:

```
import pandas as pd

# Crear un DataFrame de ejemplo
datos = {
    "Nombre": ["Ana", "Luis", "Pedro", "Marta"],
    "Edad": [23, 34, 29, 42],
    "Ciudad": ["Madrid", "Barcelona", "Sevilla", "Bilbao"]
}

df = pd.DataFrame(datos)

# Acceder a una columna como una Serie
print(df["Nombre"])

# También se puede acceder como un atributo (solo si el nombre de la columna no tiene espacios)
print(df.Nombre)
```

Ilustración 10 Ejemplo selección de columnas

- **Seleccionar varias columnas**

```
print(df[["Nombre", "Edad"]])  # Devuelve solo las columnas seleccionadas
```

Ilustración 11 Ejemplo seleccionar varias columnas

- **Selección de Filas**

  Pandas permite seleccionar filas utilizando etiquetas (loc) o índices numéricos (iloc).

```
# Seleccionar una fila por su índice con loc
print(df.loc[1])  # Devuelve la fila con índice 1

# Seleccionar una fila por su posición con iloc
print(df.iloc[2])  # Devuelve la tercera fila
```

Ilustración 12 Ejemplo de selección de filas

- **Seleccionar múltiples filas**

```
print(df.loc[0:2])  # Devuelve las filas con índice de 0 a 2
```

Ilustración 13 Ejemplo seleccionar múltiples filas

SELECCIÓN DE ELEMENTOS EN UN DATAFRAME

Podemos acceder a elementos específicos utilizando la notación de fila, columna.

```
# Seleccionar un elemento específico (fila 1, columna "Edad")
print(df.loc[1, "Edad"])  # Devuelve 34

# Usando iloc para acceder por posición
print(df.iloc[1, 1])  # También devuelve 34
```

Ilustración 14 Ejemplo selección de elementos de un Dataframe

SELECCIÓN CONDICIONAL EN UN DATAFRAME

Pandas permite aplicar condiciones lógicas para filtrar filas.

```
# Filtrar filas donde la edad sea mayor a 30
df_filtrado = df[df["Edad"] > 30]
print(df_filtrado)
```

Ilustración 15 Ejemplo selección condicional en un Dataframe

Filtrar filas con múltiples condiciones

```
df_filtrado = df[(df["Edad"] > 30) & (df["Ciudad"] == "Barcelona")]
print(df_filtrado)
```

Ilustración 16 Ejemplo filtrar filas con múltiples condiciones

MÉTODOS BÁSICOS DE EXPLORACIÓN (HEAD, TAIL, INFO, DESCRIBE)

Pandas proporciona métodos rápidos para explorar la estructura y el contenido de un DataFrame.

- **head() y tail():** Estos métodos permiten visualizar las primeras o últimas filas del DataFrame.

```
print(df.head(2))  # Primeras 2 filas
print(df.tail(1))  # Última fila
```

Ilustración 17 Ejemplo head() y tail()

- **Info():** Muestra un resumen del DataFrame, incluyendo tipos de datos y valores nulos.

```
print(df.info())
```

Ilustración 18 Ejemplo info()

- **Describe():** Genera estadísticas descriptivas para las columnas numéricas.

```
print(df.describe())
```

Ilustración 19 Ejemplo describe()

MÉTODOS BÁSICOS DE SUMARIZACIÓN (MIN, MAX, COUNT, MEAN, MEDIAN, SUM)

Pandas permite obtener información estadística de manera rápida.

```
print(df["Edad"].min())     # Valor mínimo
print(df["Edad"].max())     # Valor máximo
print(df["Edad"].count())   # Número de elementos
print(df["Edad"].mean())    # Media (promedio)
print(df["Edad"].median())  # Mediana
print(df["Edad"].sum())     # Suma total de los valores
```

Ilustración 20 Ejemplo para obtener información estadística

MÉTODOS UNIQUE, NUNIQUE, VALUE_COUNTS

Cuando trabajamos con datos categóricos en Pandas, estos métodos nos permiten obtener información sobre los valores únicos y su distribución.

- **unique():** Devuelve los valores únicos en una columna.

```
print(df['Ciudad'].unique())  # Salida: ['Madrid' 'Barcelona' 'Valencia']
```

Ilustración 21 Ejemplo método unique()

- **nunique():** Cuenta el número de valores únicos en una columna.

```
print(df['Ciudad'].nunique())  # Salida: 3
```

Ilustración 22 Ejemplo método nunique

- **value_counts():** Cuenta la frecuencia de cada valor único en una columna.

```
print(df['Ciudad'].value_counts())  # Salida: Madrid      1
                                    #         Barcelona   1
                                    #         Valencia    1
```

Ilustración 23 Ejemplo value_counts()