Pandas

Introducción

Pandas es una biblioteca Python utilizada principalmente para el análisis y manipulación de datos. Permite limpiar, transformar y analizar grandes volúmenes de datos de forma rápida y eficiente.

La estructura de datos principal en Pandas es el **DataFrame**. Se trata de una tabla bidimensional de datos, con filas y columnas, similar a una hoja de cálculo de Excel. Los DataFrames son útiles para manipular grandes cantidades de datos y ofrecen funciones como la selección y filtrado de datos, agregación, además de la capacidad de realizar operaciones matemáticas y estadísticas.

Uso básico de Pandas

Para comenzar a usar Pandas, debemos importarlo con: `import pandas as pd`. A partir de ahí, se puede usar `pd` para acceder a todas las funciones y estructuras de datos que Pandas ofrece.

Podemos crear un DataFrame a partir de un diccionario de Python de la siguiente manera:

```
import pandas as pd

data = {
    'Nombre': ['Ana', 'Juan', 'Carlos', 'María'],
    'Edad': [25, 32, 18, 47],
    'Ciudad': ['Madrid', 'Buenos Aires', 'México DF', 'Sant
iago']
}

df = pd.DataFrame(data)

df # Permite visualizar el DataFrame
```

En este caso, `df` es un DataFrame que contiene la información de nuestro diccionario. Cada clave del diccionario se convierte en una columna del

DataFrame, y los valores asociados a cada clave se convierten en las filas de esa columna.

El método `df.info()` nos proporciona información sobre el DataFrame: cantidad de filas, columnas, el tipo de dato, y si alguna de las columnas tiene un valor nulo.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nombre  4 non-null      object
 1   Edad    4 non-null      int64
 2   Ciudad  4 non-null      object
dtypes: int64(1), object(2)
memory usage: 228.0+ bytes
```

Filtrado de datos

Pandas proporciona varias formas para filtrar datos en un DataFrame. Por ejemplo, si quisiéramos seleccionar solo las filas donde 'Edad' es mayor que 30, podríamos hacerlo de la siguiente manera:

```
df[df['Edad'] > 30]
```

Este código devuelve un nuevo DataFrame que solo incluye las filas donde la condición se cumple. Podemos usar condiciones más complejas utilizando los operadores lógicos de Python `&` (y) y `|` (o).

También se pueden filtrar los datos para quedarnos solamente con las columnas que nos interesan. Por ejemplo, si solo nos interesan los nombres y edades, se pueden filtrar de esta manera:

```
df[['Nombre', 'Edad']]
```

Este código devuelve un nuevo DataFrame que incluye solamente las columnas Nombre y Edad.

Selección de datos

En Pandas, podemos seleccionar datos específicos de un DataFrame utilizando los métodos `loc` y `iloc`. El método `loc` se utiliza para seleccionar filas y columnas según sus etiquetas, mientras que `iloc` se utiliza para seleccionar filas y columnas por su posición numérica en el DataFrame.

Por ejemplo, si quisiéramos seleccionar la primera fila de nuestro DataFrame, podríamos usar `iloc` de la siguiente manera:

```
df.iloc[0]
```

Esto nos devuelve los valores de la primera fila (en el ejemplo anterior, los datos de Ana). Del mismo modo, podríamos seleccionar la primera columna (los nombres) utilizando:

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

Por otro lado, si quisiéramos seleccionar una columna por su nombre, podemos usar `loc`. Por ejemplo, para seleccionar columna “Nombre”:

```
df.loc[:, 'Nombre']
```

Para seleccionar las primeras o las últimas filas de un DataFrame, podemos usar los métodos `df.head(n)` y `df.tail(n)` respectivamente, donde *n* es la cantidad de filas que queremos que nos devuelva. Si no lo especificamos, devolverá 5 filas.

Operaciones básicas en Pandas

Pandas proporciona varias operaciones que podemos aplicar a nuestros DataFrames. Algunas de las más comunes incluyen sumar, restar, multiplicar y dividir columnas, calcular el promedio, etc. También podemos aplicar funciones personalizadas a nuestras columnas.

Por ejemplo, podríamos querer agregar una nueva columna a nuestro DataFrame que sea la suma de dos columnas existentes. Podríamos hacer esto de la siguiente manera:

```
df['Total'] = df['columna1'] + df['columna2']
```

Limpieza de datos

Otra operación común en Pandas es la limpieza de datos. Esto implica eliminar filas o columnas con valores faltantes, o reemplazar valores faltantes con algún valor predeterminado. Por ejemplo, podemos eliminar todas las filas con algún valor perdido utilizando el método `dropna()`.

Por otro lado, si deseamos reemplazar los valores faltantes en lugar de eliminarlos, podemos usar el método `fillna()`. Este método nos permite reemplazar los valores perdidos con un valor específico.

Por ejemplo, podríamos reemplazar todos los valores que falten en nuestro DataFrame con `0`:

```
df.fillna(0)
```

Otra operación que se realiza con frecuencia es la de transformar los datos. Por ejemplo, podríamos querer cambiar todos los valores de una columna a mayúsculas. Podríamos hacer esto utilizando el método `apply()`.

```
df['Nombre'] = df['Nombre'].apply(lambda x: x.upper())
```

Este código cambiará todos los valores en la columna 'Nombre' a mayúsculas.

Funciones de agregación y agrupación

Pandas también proporciona funciones de agregación y agrupación que son muy útiles para resumir y analizar datos. La función `groupby()` es una de las más usadas.

Por ejemplo, podríamos querer agrupar nuestro DataFrame por 'Ciudad' y calcular la media de 'Edad' para cada ciudad. Podríamos hacer esto de la siguiente manera:

```
df.groupby('Ciudad')['Edad'].mean()
```

Este código agrupará el DataFrame por 'Ciudad', seleccionará la columna 'Edad' y calculará la media de 'Edad' para cada grupo.

Funciones de ordenado de datos

El método `sort_values()` nos permite ordenar un DataFrame por una o más columnas. Por ejemplo, para ordenar nuestro DataFrame por 'Edad' de forma ascendente, podríamos usar `df.sort_values('Edad')`. Si queremos ordenarlo de forma descendente, añadimos el parámetro `ascending=False`.

Documentación completa

Lista completa de los métodos disponibles en la clase DataFrame de Pandas:

pandas.DataFrame — pandas 2.2.2 documentation

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html