Pandas

Series

CertiDevs

Índice de contenidos

1. Introducción

Una **Serie** en Pandas es un objeto **unidimensional** etiquetado capaz de contener cualquier tipo de datos, como enteros, flotantes, cadenas de texto y objetos de Python.

Las etiquetas en una Serie se denominan **índices**, y se utilizan para acceder y manipular elementos de la Serie.

2. Creación de Series

Puede crear una Serie en Pandas utilizando el constructor `pd.Series()`.

A continuación se muestran diferentes formas de crear Series a partir de listas, diccionarios y arrays de NumPy:

```
# Crear una Serie a partir de una lista
my_list = [10, 20, 30, 40, 50]
serie_from_list = pd.Series(my_list)
print("Serie a partir de una lista:")
print(serie_from_list)

# Crear una Serie a partir de un diccionario
my_dict = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
serie_from_dict = pd.Series(my_dict)
print("\nSerie a partir de un diccionario:")
print(serie_from_dict)

# Crear una Serie a partir de un array de NumPy
my_array = np.array([10, 20, 30, 40, 50])
serie_from_array = pd.Series(my_array)
print("\nSerie a partir de un array de NumPy:")
print(serie_from_array)
```

3. Relación con arrays de Numpy

Una **Serie** en Pandas es un objeto unidimensional que puede contener datos de cualquier tipo, como enteros, flotantes, cadenas de caracteres, etc.

Internamente, una **Serie de Pandas** almacena sus datos en un **array de Numpy**, pero también incluye etiquetas para los **índices** y tiene algunas funcionalidades adicionales específicas de Pandas.

Aunque una Serie de Pandas se basa en un array de Numpy, no es exactamente igual a un array de Numpy. Sin embargo, es fácil convertir una Serie en un array de Numpy si es necesario.

Por ejemplo, dada una Serie de Pandas llamada serie, puedes obtener el array de Numpy subyacente utilizando la propiedad `.values`:

```
import pandas as pd
import numpy as np

# Crear una serie de Pandas
serie = pd.Series([1, 2, 3, 4, 5])

# Obtener el array de Numpy subyacente
numpy_array = serie.values

print(type(numpy_array))  # Output: <class 'numpy.ndarray'>
```

En este ejemplo, numpy_array es un array de Numpy que contiene los mismos datos que la Serie de Pandas serie.

4. Acceder a elementos y slicing en Series

Puede **acceder a los elementos** de una Serie utilizando sus índices, ya sea por posición o por etiqueta (si se proporcionaron etiquetas en el momento de la creación).

También puede rebanar una Serie para seleccionar un subconjunto de elementos:

```
# Acceder a elementos por posición
print("\nPrimer elemento de la serie:")
print(serie_from_list[0])

# Acceder a elementos por etiqueta
print("\nElemento con etiqueta 'a':")
print(serie_from_dict['a'])

# Rebanar una Serie
print("\nPrimeros tres elementos de la serie:")
print(serie_from_list[:3])
```

5. Operaciones básicas y funciones estadísticas en Series

Pandas admite una amplia variedad de operaciones matemáticas y funciones estadísticas en Series.

Algunas de las funciones más comunes incluyen `sum()`, `mean()`, `median()`, `min()`, `max()` y `std()`:

```
# Operaciones básicas en Series
addition = serie_from_list + 10
multiplication = serie_from_list * 2

# Funciones estadísticas en Series
```

```
total = serie_from_list.sum()
mean = serie_from_list.mean()
median = serie_from_list.median()
minimum = serie_from_list.min()
maximum = serie_from_list.max()
std_dev = serie_from_list.std()
```

6. Modificar elementos y agregar datos a una Serie

Puede modificar elementos de una Serie asignando un nuevo valor a un índice específico.

Además, puede agregar datos a una Serie utilizando el método `append()`:

```
# Modificar un elemento de una Serie
serie_from_list[0] = 100
print("\nSerie modificada:")
print(serie_from_list)

# Agregar datos a una Serie
new_data = pd.Series([60, 70, 80])
serie_extended = serie_from_list.append(new_data, ignore_index=True)
print("\nSerie extendida:")
print(serie_extended)
```

7. Manejar datos faltantes en Series

Pandas utiliza el valor **NaN (Not a Number)** para representar **datos faltantes** en una Serie.

Puede verificar si una Serie tiene datos faltantes utilizando el método `isnull()` y reemplazar los valores faltantes con un valor específico utilizando el método `fillna()`:

Crear una Serie con datos faltantes

```
serie_with_nan = pd.Series([10, 20, np.nan, 40, 50])
print("\nSerie con datos faltantes:")
print(serie_with_nan)
```

Verificar si una Serie tiene datos faltantes

```
null_values = serie_with_nan.isnull()
print("\nValores faltantes en la serie:")
print(null_values)
```

Reemplazar datos faltantes con un valor específico

```
serie_no_nan = serie_with_nan.fillna(0)
print("\nSerie con valores faltantes reemplazados:")
print(serie_no_nan)
```