**Introducción a Pandas**

Pandas es una biblioteca de Python ampliamente utilizada para manipulación y análisis de datos. Proporciona estructuras de datos de alto nivel, herramientas para trabajar con datos faltantes y funciones para la manipulación y limpieza de datos. Las dos estructuras de datos principales en Pandas son Series y DataFrame.

**Estructuras de Datos en Pandas**

1. **Series**: Una Serie es un arreglo unidimensional etiquetado que puede contener cualquier tipo de datos, como enteros, flotantes, cadenas, etc. Cada elemento en una Serie tiene una etiqueta o índice asociado.
2. **DataFrame**: Un DataFrame es una estructura de datos bidimensional similar a una tabla de base de datos o una hoja de cálculo de Excel. Está compuesto por filas y columnas, donde cada columna puede contener datos de un tipo diferente.

**Ejemplo 6: Creación de una Serie y un DataFrame en Pandas**

python

import pandas as pd

# Crear una Serie a partir de una lista

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

print("Serie:")

print(serie)

# Crear un DataFrame a partir de un diccionario

data = {'Nombre': ['Juan', 'María', 'Pedro', 'Ana'],

'Edad': [25, 30, 35, 40],

'Ciudad': ['Bogotá', 'Medellín', 'Cali', 'Barranquilla']}

df = pd.DataFrame(data)

print("\nDataFrame:")

print(df)

**Ejercicio 3: Manipulación y Limpieza de Datos con Pandas**

Para este ejercicio, supongamos que tenemos un DataFrame con datos de ventas de productos en una tienda en línea. Queremos limpiar y manipular estos datos para análisis posteriores. Aquí hay un ejemplo de cómo podríamos hacerlo:

python

# Supongamos que tenemos un DataFrame llamado 'ventas\_df'

# Limpiar los datos eliminando filas con valores faltantes

ventas\_df = ventas\_df.dropna()

# Convertir la columna 'Fecha' a tipo de dato de fecha y hora

ventas\_df['Fecha'] = pd.to\_datetime(ventas\_df['Fecha'])

# Agregar una nueva columna 'Ventas Totales' que calcula las ventas totales por producto

ventas\_df['Ventas Totales'] = ventas\_df['Cantidad'] \* ventas\_df['Precio']

# Filtrar solo las ventas realizadas en el último mes

ventas\_ultimo\_mes = ventas\_df[ventas\_df['Fecha'] >= pd.Timestamp('today') - pd.DateOffset(months=1)]

Este es solo un ejemplo simple de cómo podríamos manipular y limpiar datos con Pandas. Las posibilidades son casi ilimitadas dependiendo de los datos específicos que estés trabajando y los análisis que quieras realizar.

**Ejercicio 3: Manipulación y Limpieza de Datos con Pandas**

**1. Eliminar Datos Faltantes**

Los datos faltantes son comunes en conjuntos de datos del mundo real. Pandas proporciona métodos para eliminar filas o columnas que contienen datos faltantes.

python

# Eliminar filas con valores faltantes

ventas\_df = ventas\_df.dropna()

# Eliminar columnas con valores faltantes

ventas\_df = ventas\_df.dropna(axis=1)

**2. Convertir Tipos de Datos**

Es importante asegurarse de que los tipos de datos de las columnas sean apropiados para el análisis. Pandas proporciona métodos para convertir tipos de datos, como fechas y números.

python

# Convertir la columna 'Fecha' a tipo de dato de fecha y hora

ventas\_df['Fecha'] = pd.to\_datetime(ventas\_df['Fecha'])

# Convertir la columna 'Cantidad' a tipo entero

ventas\_df['Cantidad'] = ventas\_df['Cantidad'].astype(int)

**3. Agregar Columnas Calculadas**

A menudo necesitamos agregar nuevas columnas al DataFrame que sean el resultado de operaciones en otras columnas.

python

# Agregar una nueva columna 'Ventas Totales' que calcula las ventas totales por producto

ventas\_df['Ventas Totales'] = ventas\_df['Cantidad'] \* ventas\_df['Precio']

# Agregar una nueva columna 'Precio Unitario' que calcula el precio unitario de cada producto

ventas\_df['Precio Unitario'] = ventas\_df['Precio'] / ventas\_df['Cantidad']

**4. Filtrar Datos**

Podemos filtrar datos basados en condiciones específicas utilizando métodos de indexación booleana.

python

# Filtrar solo las ventas realizadas en el último mes

ventas\_ultimo\_mes = ventas\_df[ventas\_df['Fecha'] >= pd.Timestamp('today') - pd.DateOffset(months=1)]

# Filtrar las ventas de un producto específico

ventas\_producto\_1 = ventas\_df[ventas\_df['Producto'] == 'Producto 1']

**5. Agrupar y Resumir Datos**

Pandas permite agrupar datos por ciertas columnas y realizar operaciones de resumen, como sumar, contar o calcular el promedio.

python

# Calcular las ventas totales por producto

ventas\_por\_producto = ventas\_df.groupby('Producto')['Ventas Totales'].sum()

# Calcular el número total de ventas por categoría

ventas\_por\_categoria = ventas\_df.groupby('Categoria').size()

# Calcular el precio promedio por producto

precio\_promedio\_por\_producto = ventas\_df.groupby('Producto')['Precio'].mean()

**Conclusiones**

Estos son solo algunos ejemplos de las muchas formas en que puedes manipular y limpiar datos utilizando Pandas. La biblioteca ofrece una amplia gama de funcionalidades para realizar análisis de datos de manera eficiente y efectiva, lo que la convierte en una herramienta invaluable para cualquier científico de datos o analista. ¡Espero que esta introducción te ayude a comenzar con tu análisis de datos utilizando Pandas!