**FUNDAMENTOS DE PROGRAMACIÓN – RETO 9 – 10**

**Mariana Duque Rúa**

**C.C. 1214747015**

**mariana.duque3@udea.edu.co**

**PREGUNTAS**

1. ¿Cuáles son los elementos diferenciales de librería Pandas?

* DataFrames: Los DataFrames son estructuras de datos multidimensionales que permiten almacenar y manipular datos de forma eficiente. Es similar a una tabla de base de datos o una hoja de cálculo de Excel.
* Series: Las Series son estructuras de datos unidimensionales que permiten almacenar datos de una sola columna similares a un array o lista en Python, pero con etiquetas en los datos, es decir, cada elemento en una Serie tiene un índice asociado.
* Indexación: La indexación en Pandas permite acceder a los datos de forma eficiente.
* Filtrado: El filtrado en Pandas permite seleccionar los datos que cumplen una determinada condición.
* Estadísticas: Pandas proporciona una amplia gama de funciones estadísticas para analizar los datos en términos de limpiar, transformar, analizar y visualizar los datos de manera efectiva.

1. ¿Cómo se realiza el parseo de datos y para qué sirve?

El parseo de datos se refiere a convertir datos en un formato específico en un objeto de Pandas (como un DataFrame). Pandas ofrece funciones como pd.read\_csv(), pd.read\_excel(), etc., para parsear datos desde diferentes formatos de archivo y que puedan ser manipulados por Pandas. Por ejemplo, el siguiente código lee datos de un archivo CSV y los convierte en un DataFrame.

**import pandas as pd**

**# Leer datos de un archivo CSV**

**df = pd.read\_csv("data.csv")**

**# Imprimir el DataFrame**

**print(df)**

1. ¿Cómo se estructuran los filtros en la información?

Los filtros en la información son condiciones que se utilizan para seleccionar los datos que cumplen una determinada condición. En el contexto de Pandas, los filtros se pueden estructurar de varias maneras.

Una forma de estructurar los filtros es utilizando la sintaxis de Python. Por ejemplo, el siguiente código filtra los datos de un DataFrame para seleccionar las filas donde el valor de la columna edad es mayor que 18.

**import pandas as pd**

**# Leer datos de un archivo CSV**

**df = pd.read\_csv("data.csv")**

**# Filtrar datos**

**df = df[df["edad"] > 18]**

**# Imprimir el DataFrame filtrado**

**print(df)**

Otra forma de estructurar los filtros es utilizando las funciones de Pandas. Por ejemplo, el siguiente código filtra los datos de un DataFrame para seleccionar las filas donde el valor de la columna edad está entre 20 y 30.

**import pandas as pd**

**# Leer datos de un archivo CSV**

**df = pd.read\_csv("data.csv")**

**# Filtrar datos**

**df = df[df["edad"].between(20, 30)]**

**# Imprimir el DataFrame filtrado**

**print(df)**

Además, los filtros en Pandas se pueden aplicar usando operadores de comparación (<, >, ==, etc.) o funciones de filtro como df.query(). También se pueden combinar múltiples condiciones utilizando operadores lógicos como & (AND) y | (OR) para estructurar filtros más complejos.

1. ¿Cómo se realiza un histograma?

Un histograma es una representación gráfica de la distribución de datos. En el contexto de Pandas, se puede crear un histograma utilizando la función hist().

También es posible personalizar el histograma especificando el número de bins, el rango de valores y otras propiedades gráficas para hacer que el histograma se adapte a ciertas necesidades. Por ejemplo, el siguiente código crea un histograma de los datos de una columna de un DataFrame.

**import pandas as pd**

**import matplotlib.pyplot as plt**

**# Leer datos de un archivo CSV**

**df = pd.read\_csv("data.csv")**

**# Crear un histograma**

**plt.hist(df["edad"])**

**# Mostrar el histograma**

**plt.show()**

1. ¿Cómo se consumen archivos planos?

Los archivos planos son archivos de texto que contienen datos en forma de filas y columnas. Pandas puede leer datos desde archivos planos como CSV (Comma-Separated Values) y archivos de texto delimitados por tabulaciones, comas u otros caracteres.

Para leer un archivo CSV, es posible usar pd.read\_csv('archivo.csv'). Para archivos de texto delimitados, es posible usar pd.read\_table('archivo.txt', delimiter='\t') donde '\t' representa el delimitador (en este caso, una tabulación).

Por ejemplo, el siguiente código consume un archivo plano y lo convierte en un DataFrame.

**import pandas as pd**

**# Leer datos de un archivo plano**

**df = pd.read\_table("data.txt", sep=",")**

**# Imprimir el DataFrame**

**print(df)**