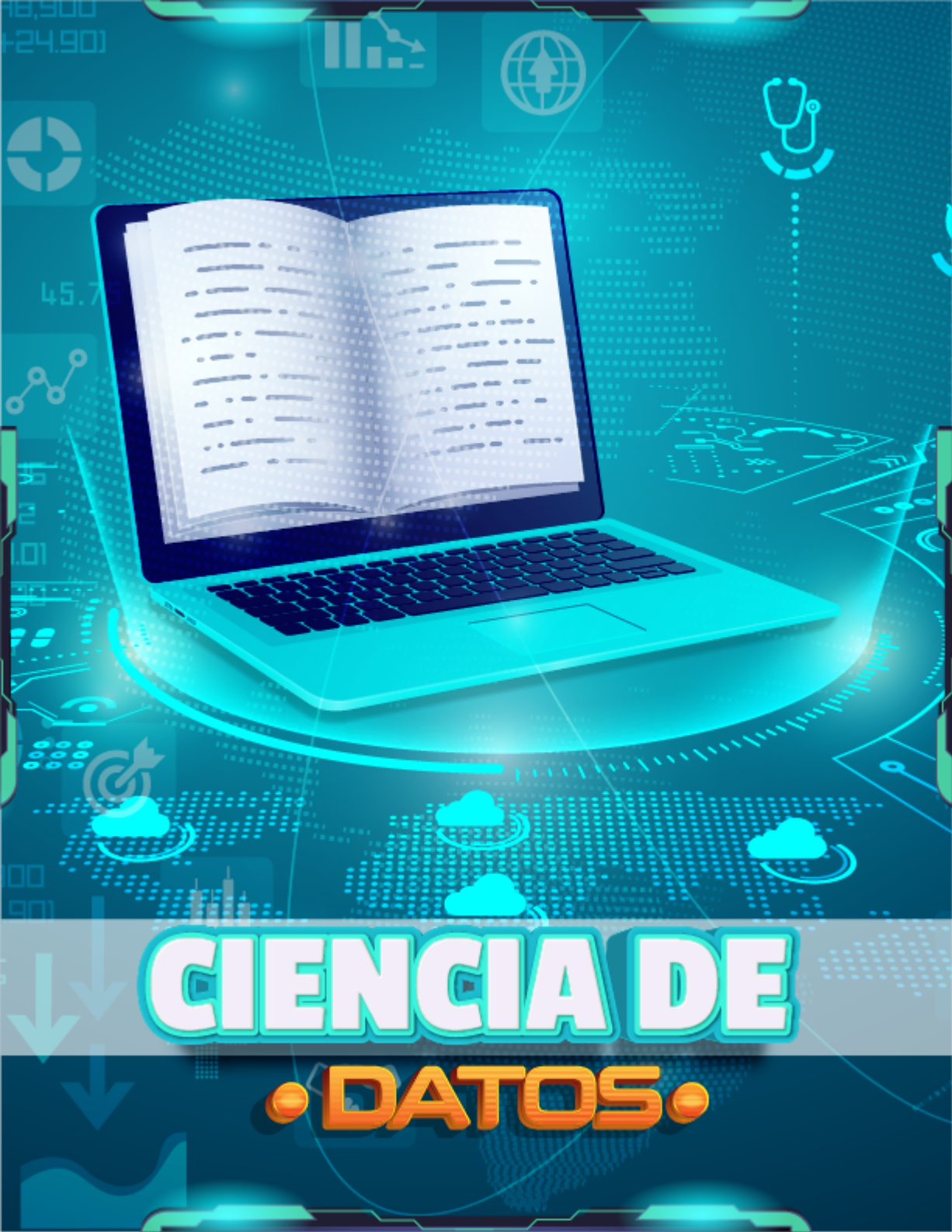

CIENCIA DE
DATOS

Guía de los Comandos más Utilizados para el Procesamiento de Datos en Python

Python es uno de los lenguajes de programación más versátiles y poderosos utilizados en diversas disciplinas, desde la ciencia de datos hasta la inteligencia artificial. En este documento, discutiremos algunos de los comandos más utilizados en Python para el procesamiento de datos.

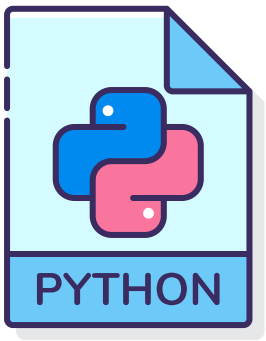
Importación de Bibliotecas

Antes de empezar a procesar datos en Python, es necesario importar las bibliotecas pertinentes. A continuación se mencionan dos de las bibliotecas más comúnmente utilizadas en el procesamiento de datos:

pandas: Una biblioteca de software escrita para la manipulación y el análisis de datos.

numpy: Una biblioteca para el lenguaje de programación Python que da soporte a arrays y matrices de gran tamaño, junto con una gran colección de funciones matemáticas de alto nivel.

```
import pandas as pd
import numpy as np
```

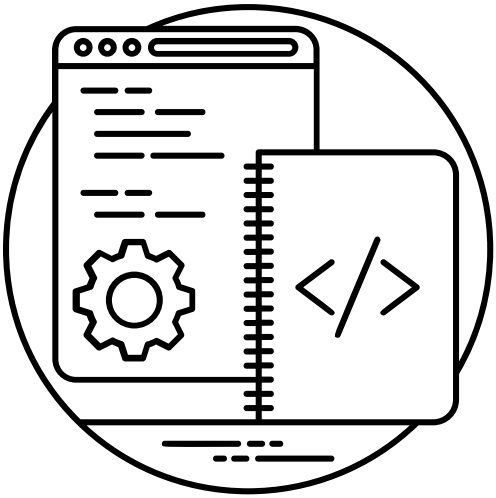


Lectura de Datos

La biblioteca **pandas** proporciona varias funciones para leer datos en diferentes formatos. Aquí está la forma más común de leer datos:

```
df = pd.read_csv('filename.csv')
```

También se pueden leer otros tipos de archivos, como Excel (read_excel), JSON (read_json), etc.

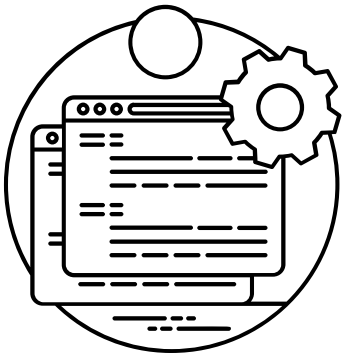


Exploración de Datos

Una vez que los datos se han cargado en Python, es posible explorarlos utilizando varias funciones:

head(): Muestra las primeras N filas de datos.
tail(): Muestra las últimas N filas de datos.
describe(): Proporciona un resumen estadístico de los datos.
info(): Proporciona un resumen de los tipos de datos y los valores no nulos.

```
df.head()
df.tail()
df.describe()
df.info()
```



Selección y Filtrado de Datos

Pandas permite seleccionar y filtrar datos de muchas maneras diferentes:

Seleccionar una columna: df['column_name']
Seleccionar varias columnas: df[['column_name1', 'column_name2']]
Filtrar filas: df[df['column_name'] > value]

```
df['column_name']
df[['column_name1', 'column_name2']]
df[df['column_name'] > value]
```

Manipulación de Datos

Python también proporciona una variedad de formas de manipular datos, incluyendo:

rename(): Cambia el nombre de las columnas.
drop(): Elimina columnas o filas.
fillna(): Rellena los valores nulos.

```
df.rename(columns={'old_name': 'new_name'})
df.drop('column_name', axis=1)
df['column_name'].fillna(value)
```

Agrupación y Agregación de Datos

Pandas proporciona la función groupby() para agrupar los datos y funciones de agregación como sum(), mean(), etc., para calcular estadísticas sobre cada grupo.

```
df.groupby('column_name').sum()
df.groupby('column_name').mean()
```

Fusión, Unión y Concatenación de Datos

Python permite fusionar, unir y concatenar datos utilizando las funciones merge(), join() y concat(), respectivamente.

```
pd.merge(df1, df2, on='common_column')
df1.join(df2, on='common_column')
pd.concat([df1, df2])
```

Operaciones con Series de Tiempo

Pandas ofrece varias funciones para trabajar con datos de series de tiempo:

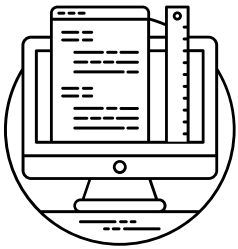
to_datetime(): Convierte una cadena a un objeto datetime.
resample(): Cambia la frecuencia de las series de tiempo.

```
pd.to_datetime(df['column_name'])
df.resample('D').mean()
# Re-muestreo a nivel de día y calcular la media.
```

Escritura de Datos

Finalmente, pandas proporciona funciones para escribir datos en diferentes formatos. La forma más común de escribir datos es en un archivo CSV.

```
df.to_csv('new_filename.csv', index=False)
```



Recomendaciones

Siempre es recomendable explorar los datos antes de empezar a procesarlos. Esto incluye entender las características de los datos, identificar cualquier valor nulo o perdido y comprender la distribución de los datos.

Durante el filtrado o la selección de datos, asegúrese de entender cómo estas operaciones afectan la estructura de los datos.

Al manipular los datos, recuerde que pandas usualmente retorna un nuevo DataFrame. Por lo tanto, si desea que los cambios se apliquen al DataFrame original, necesitará asignar el resultado de vuelta al DataFrame original.

Al agrupar y agregar datos, tenga en cuenta el contexto y asegúrese de que las operaciones de agregación tengan sentido en ese contexto.

Al unir datos, tenga cuidado con los valores duplicados o perdidos que podrían introducirse en los datos.

Cuando trabaje con series de tiempo, tenga en cuenta la zona horaria y asegúrese de que todos los datos estén en la misma zona horaria.

Finalmente, siempre es una buena práctica guardar los datos procesados para futuras referencias o análisis.

Espero que esta guía le sirva como punto de partida para el procesamiento de datos en Python. Recuerde que Python es una herramienta poderosa y flexible, y hay muchas más funciones y técnicas disponibles para el procesamiento de datos que las que se han discutido aquí.