IT Academy by KIBERNUM

Extracción de datos desde archivos y fuentes externas con Python: CSV, Excel y Web

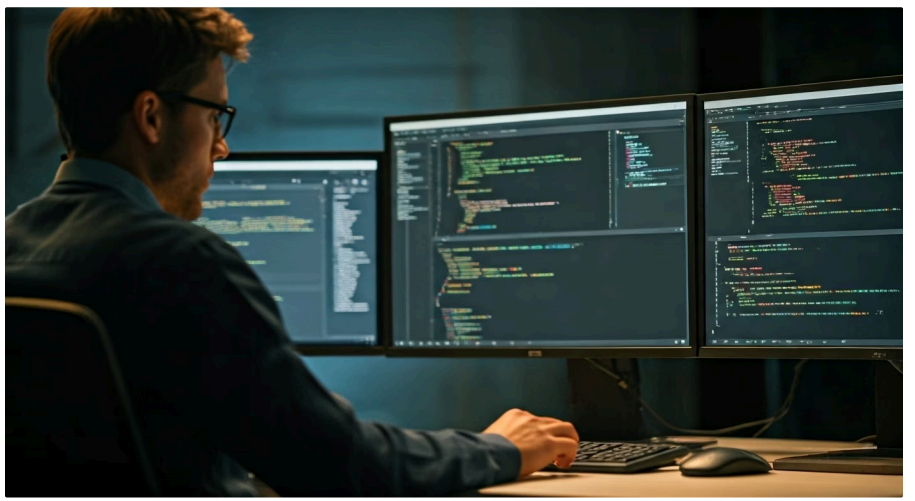
La etapa de obtención y extracción de datos es crucial en cualquier pipeline de ingeniería de datos y ciencia de datos. Una buena parte de los problemas en proyectos reales no reside en el modelado, sino en la obtención, integración y validación de los datos provenientes de diversas fuentes. Python, por su flexibilidad y ecosistema de librerías, ha facilitado enormemente este proceso, permitiendo leer, transformar y exportar datos desde y hacia múltiples formatos de archivos y servicios web, con apenas unas líneas de código.

R **por Kibernum Capacitación S.A.**

Introducción a la extracción de datos

Este documento explora en profundidad las técnicas y librerías clave para la extracción de datos desde archivos CSV, archivos Excel, y páginas web, cubriendo las operaciones de lectura, escritura y transformación de datos, y ejemplificando con situaciones y desafíos típicos de la industria.

La capacidad de extraer datos de diversas fuentes es fundamental para cualquier proyecto de análisis o ingeniería de datos. Python se ha convertido en la herramienta preferida por su versatilidad y potentes librerías.



Archivos CSV: Concepto y uso en ingeniería de datos

¿Qué es un archivo CSV?

Un CSV (Comma Separated Values) es un formato de archivo de texto plano ampliamente utilizado para almacenar datos tabulares, donde cada línea representa una fila y cada valor está separado por una coma (o, en ocasiones, por punto y coma u otros delimitadores). Su simplicidad lo hace ideal para el intercambio de información entre sistemas heterogéneos, así como para exportar e importar datos desde hojas de cálculo, bases de datos y plataformas web.

> Ejemplo de contenido típico de un CSV:

```
nombre,edad,ciudad
Ana,23,Madrid
Luis,35,Barcelona
Sofía,28,Valencia
```

Este formato es legible, portátil y soportado por casi todas las herramientas de análisis de datos.

Leyendo un archivo CSV con Python y Pandas

La librería Pandas provee la función read_csv(), que permite cargar datos desde archivos CSV de manera eficiente y con opciones avanzadas de control sobre delimitadores, encoding, manejo de valores faltantes, y conversión de tipos de datos.

Ejemplo básico de lectura:

```
import pandas as pd

df = pd.read_csv('datos_clientes.csv')
print(df.head())
```

Esto crea un DataFrame con los datos del archivo.

Parámetros útiles en la lectura de CSV:

- sep=',': Define el delimitador, por defecto la coma.
- header=0: Indica la fila que contiene los nombres de las columnas.
- index_col=None: Permite establecer una columna como índice.
- na_values=['NA', '']: Especifica cómo se identifican los valores nulos.
- encoding='utf-8': Define el tipo de codificación, útil para archivos con caracteres especiales.

Ejemplos avanzados de CSV en Python

Ejemplo avanzado:

```
df = pd.read_csv('clientes.csv', sep=';', encoding='latin1', na_values=['NA', ''], index_col='ID')
```

Leyendo grandes archivos CSV

En situaciones reales, los archivos pueden ser muy grandes y no caber en memoria. Pandas permite leerlos por partes (chunks):

```
for chunk in pd.read_csv('grande.csv', chunksize=10000):
    print(chunk['columna_interes'].mean())
```

Esto es clave para aplicaciones de big data y procesamiento distribuido.

Lectura del archivo CSV

Importar pandas y leer el archivo con los parámetros adecuados

Procesamiento de datos

Filtrar, transformar y limpiar los datos según necesidades

Exportación de resultados

Guardar los datos procesados en un nuevo archivo CSV

Escribiendo un archivo CSV

Guardar los datos transformados o procesados en CSV es igual de sencillo, usando el método to_csv() del DataFrame:

```
df.to_csv('salida.csv', index=False, sep=',', encoding='utf-8')
```

- index=False evita guardar el índice como columna.
- Se pueden personalizar delimitadores, codificación y representación de nulos (na_rep).

Exportar solo ciertas columnas:

```
df[['nombre', 'ingresos']].to_csv('clientes_exportados.csv', index=False)
```

Ejemplo integrado: Filtrar, transformar y exportar

Supón que se requiere extraer clientes de Madrid con ingresos mayores a 3000:

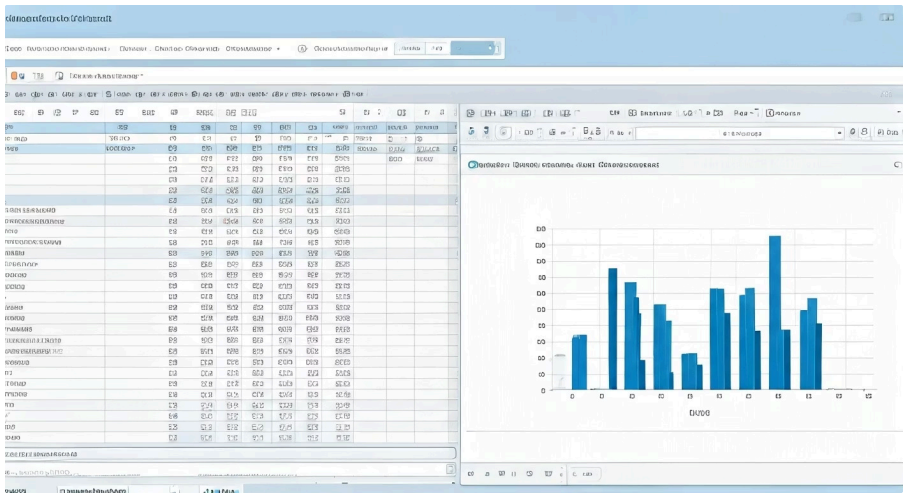
```
df = pd.read_csv('clientes.csv')
filtro = (df['ciudad'] == 'Madrid') & (df['ingresos'] > 3000)
df_filtrado = df[filtro]
df_filtrado.to_csv('clientes_madrid_alto.csv', index=False)
```

Esto ilustra la potencia de Pandas para extracción, filtrado y almacenamiento en pipelines de datos.

Archivos Excel: Características y manipulación en Python

¿Qué es un archivo Excel y por qué es relevante?

Los archivos Excel (extensiones .xls, .xlsx) son omnipresentes en el mundo empresarial y científico. Permiten almacenar datos tabulares, gráficos, fórmulas, y más, en hojas múltiples dentro de un mismo archivo. Son frecuentemente utilizados como formato de intercambio y almacenamiento intermedio, y muchos sistemas legacy exportan o reciben información en este formato.



Librerías utilitarias para trabajar con Excel

Aunque existen varias librerías en Python para manipular archivos Excel, Pandas es la más conveniente para tareas de análisis. Por debajo, utiliza motores como openpyxl, xlrd o xlsxwriter según la operación y el tipo de archivo.

Nota: A partir de versiones recientes, Pandas ya no utiliza xlrd para archivos .xlsx (solo para .xls). Para .xlsx recomienda openpyxl. Sin embargo, por requerimiento académico, se menciona la librería xlrd y su uso.

Leyendo un archivo Excel con Pandas

Pandas ofrece la función read_excel() para leer hojas de cálculo. Permite seleccionar la hoja de interés, definir rangos, establecer índices, y convertir datos automáticamente.

Ejemplo básico:

```
df = pd.read_excel('clientes.xlsx', sheet_name='Hoja1')
print(df.head())
```

- sheet_name puede ser el nombre o el número de la hoja.

Parámetros importantes:

- header: Fila donde se encuentran los nombres de columnas.
- usecols: Especifica columnas a leer (por nombres o posiciones).
- nrows: Número de filas a importar (útil para muestreo o archivos grandes).

Leer varias hojas a la vez:

```
datos = pd.read_excel('clientes.xlsx', sheet_name=None) # Devuelve un diccionario {hoja: DataFrame}
```

Leer con xlrd (formato antiguo .xls):

```
import xlrd
book = xlrd.open_workbook('archivo.xls')
sheet = book.sheet_by_name('Hoja1')
for row_idx in range(sheet.nrows):

print(sheet.row_values(row_idx))
```

Este enfoque es más bajo nivel y menos flexible que Pandas, pero importante para compatibilidad.

Escribiendo un archivo Excel

Para exportar datos procesados o generados, Pandas utiliza el método to_excel():

```
df.to_excel('resultado.xlsx', index=False, sheet_name='DatosLimpios')
```

- Por defecto, utiliza el motor openpyxl para .xlsx y xlsxwriter para características avanzadas (como formato o gráficos).
- Permite exportar múltiples hojas con ExcelWriter:

```
with pd.ExcelWriter('varias_hojas.xlsx') as writer:
    df1.to_excel(writer, sheet_name='Ventas')
    df2.to_excel(writer, sheet_name='Clientes')
```

Ejemplo integrado: Extracción, procesamiento y exportación Excel

Imagina que necesitas extraer la hoja de ventas de 2024, filtrar ventas superiores a $10,000 y guardar el resultado:

```
df = pd.read_excel('ventas_2024.xlsx', sheet_name='Ventas')
ventas_altas = df[df['monto'] > 10000]
ventas_altas.to_excel('ventas_altas_2024.xlsx', index=False)
```

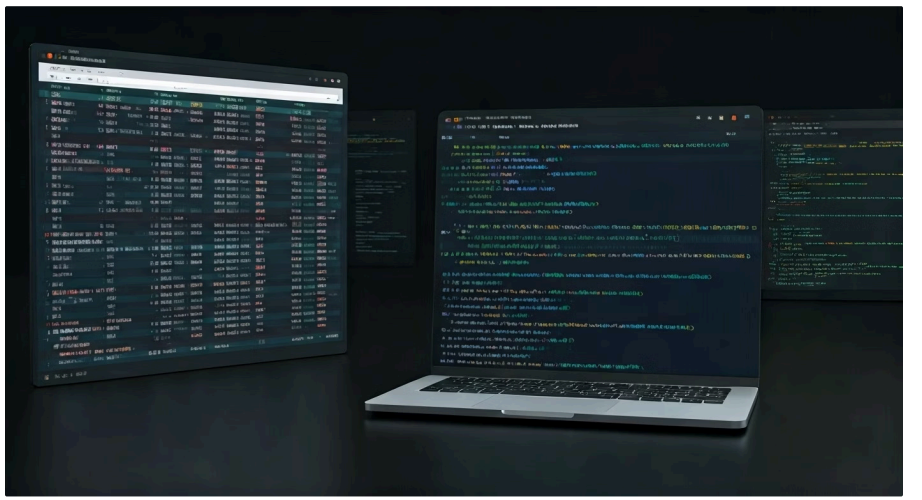
Leer tablas web con Pandas

Cada vez es más común extraer datos de la web para análisis y monitoreo (web scraping). Pandas facilita esta tarea con read_html(), que utiliza internamente BeautifulSoup y lxml para parsear HTML y buscar automáticamente tablas <table> en páginas web.

Ejemplo de lectura de tablas web:

```
url = 'https://es.wikipedia.org/wiki/Anexo:Países_por_PIB'
tablas = pd.read_html(url)
print(f"Se encontraron {len(tablas)} tablas.")
print(tablas[0].head()) # Imprime la primera tabla encontrada
```

- El resultado es una lista de DataFrames, uno por cada tabla HTML encontrada.
- Ideal para extraer rankings, precios, indicadores económicos, datos de portales abiertos, etc.



Consideraciones éticas y técnicas en web scraping

- Legalidad: Verificar términos de uso del sitio, respeto a robots.txt.
- Frecuencia: No saturar los servidores web.
- Validación: Las estructuras HTML pueden cambiar, requiere robustez y validaciones.

Comparativa y ventajas de cada formato/fuente

Formato/Fuente	Ventajas	Limitaciones	Ejemplo de aplicación
CSV	Simple, portable, universal, eficiente	No soporta fórmulas, sin metadatos complejos	Intercambio rápido de datos tabulares, logs, exportaciones automáticas
Excel	Multitabla, soporta fórmulas, metadatos, formatos	Mayor peso, complejidad, dependencias de librerías	Informes empresariales, seguimiento contable, datos con formatos especiales
Web (HTML)	Acceso a información pública, datos dinámicos	Requiere validación y parsing, cambios de estructura, cuestiones legales	Seguimiento de precios, rankings, scraping de portales abiertos

La elección del formato adecuado dependerá del contexto específico del proyecto, las fuentes disponibles y los requisitos de procesamiento.

Desafíos comunes en la obtención y extracción de datos

Encoding y formatos regionales

Archivos exportados en distintos sistemas pueden usar distintos encodings, delimitadores o convenciones de decimales/comas.

Datos faltantes y tipos incorrectos

Es frecuente que los datos presenten errores, nulos o mezclas de tipos inesperadas.

Archivos corruptos o con hojas múltiples

Pandas permite manejar estos casos con parámetros específicos, pero exige pruebas y validación.

Automatización

Es habitual integrar estas técnicas en pipelines programados o notebook de análisis reproducibles.

Superar estos desafíos requiere conocimiento técnico, experiencia y un enfoque metódico para garantizar la calidad e integridad de los datos extraídos.

Actividades y retos prácticos integrados

Análisis de datos del INE

Descargar y analizar datos del INE (España) o del Banco Mundial en formato CSV y Excel, aplicando filtros, generando resúmenes y exportando resultados.

Comparativa de ventas

Comparar las ventas por región a partir de archivos Excel multi-hoja, integrando los datos en un solo DataFrame.

Web scraping

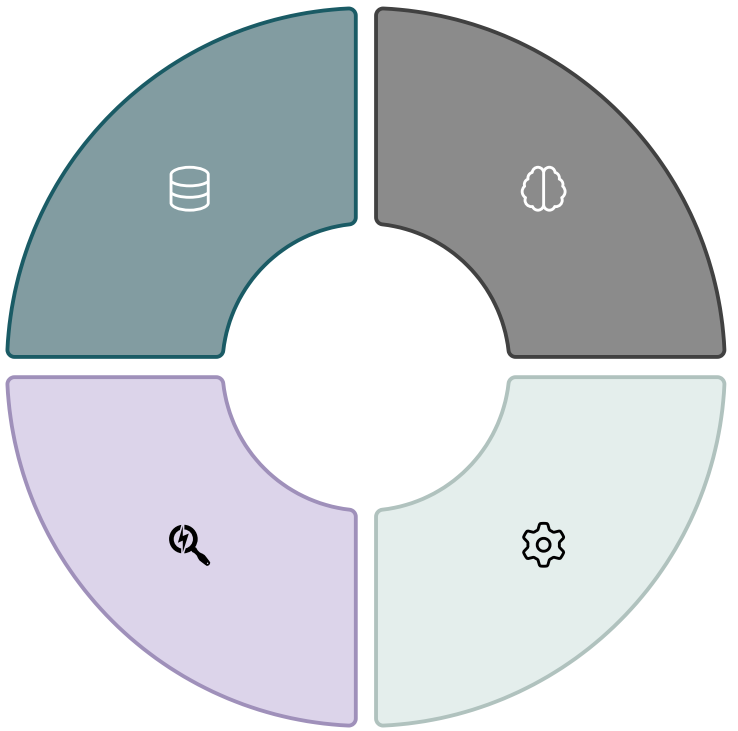
Web scraping de una tabla de Wikipedia: Extraer la tabla de población por país y calcular estadísticas básicas sobre los 10 países más poblados.

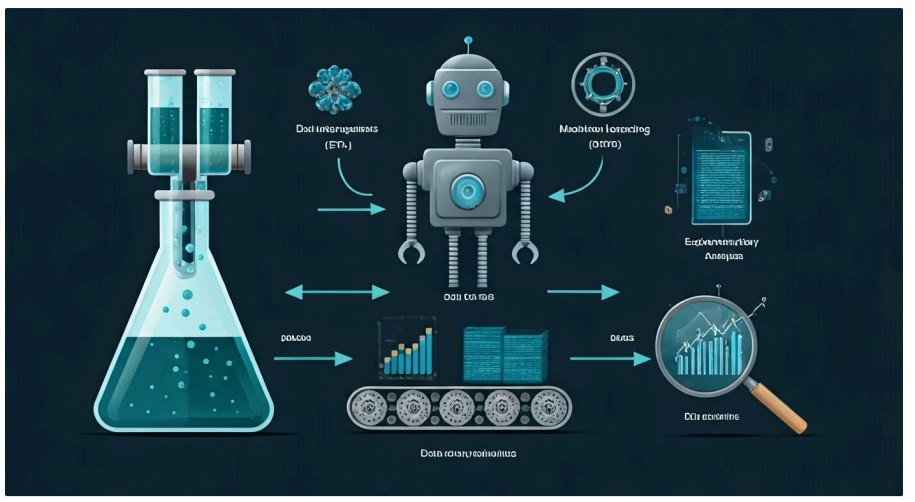
Automatización

Programar una función que lea automáticamente todos los archivos CSV de un directorio, los concatene y genere un resumen agregado.

Estos ejercicios prácticos permiten aplicar los conceptos aprendidos y desarrollar habilidades esenciales para el trabajo con datos en entornos reales.

Relación con otras áreas de la ingeniería de datos y ciencia de datos





ETL

Estas técnicas constituyen el pilar de la fase de extracción en procesos ETL, previos al análisis avanzado o la carga en bases de datos.

Machine learning

La calidad de los modelos depende fuertemente de la correcta y robusta extracción de los datos.

Data pipelines

La integración de lectura/escritura desde múltiples fuentes es base para flujos automáticos en la industria (Apache Airflow, Luigi, pipelines cloud, etc.).

Análisis exploratorio

La flexibilidad para cargar, limpiar y transformar datos desde cualquier formato es requisito fundamental para el análisis moderno.

Recursos y conclusión

Recursos y documentación recomendados

- Wes McKinney, Python for Data Analysis, O'Reilly Media.
- Documentación oficial de Pandas: https://pandas.pydata.org/docs/
- Python for Data Science for Dummies (Wiley).
- BeautifulSoup y lxml para scraping avanzado: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Conclusión

La capacidad de extraer datos desde archivos CSV, Excel o fuentes web, usando Python y sus librerías utilitarias, es una competencia central para cualquier profesional en formación o activo en ingeniería de datos, análisis estadístico, ciencia de datos o inteligencia artificial. Dominar estos métodos permite construir flujos de datos robustos, reproducibles y adaptables a los desafíos cambiantes de la industria y la investigación, sentando las bases para una exploración, análisis y modelado exitosos.

Documentación de Pandas | **Guía de BeautifulSoup**