DESCUBRIENDO EL PODER DE LA PROGRAMACION

CURSO INICIAL DE PYTHON

Jimmy Chung
Alexander Solis

CONTENIDO DEL CURSO

1. Introducción, Instalación y Conceptos básicos
2. Variables, Expresiones, Funciones y Operadores
3. Condicionales y Ciclos – Patrones de sintaxis válidos
4. Listas, tuplas y diccionarios
5. Errores y Excepciones
6. Objetos, Clases y funciones en Python
7. Guardar y Recuperar datos de archivos

Clase 6: Archivos CSV, Excel y librería Pandas

Funciones «built-in»

Funciones «built-in»: Incorporadas por defecto en el propio lenguaje

`abs()`	`delattr()`	`hash()`	`memoryview()`	`set()`
`all()`	`dict()`	`help()`	`min()`	`setattr()`
`any()`	`any()`	`hex()`	`next()`	`slice()`
`ascii()`	`divmod()`	`id()`	`object()`	`sorted()`
`bin()`	`enumerate()`	`input()`	`oct()`	`staticmethod()`
`bool()`	`eval()`	`int()`	`open()`	`str()`
`breakpoint()`	`exec()`	`isinstance()`	`ord()`	`sum()`
`bytearray()`	`filter()`	`issubclass()`	`pow()`	`super()`
`bytes()`	`float()`	`iter()`	`print()`	`tuple()`
`callable()`	`format()`	`len()`	`property()`	`type()`
`chr()`	`frozenset()`	`list()`	`range()`	`vars()`
`classmethod()`	`getattr()`	`locals()`	`repr()`	`zip()`
`compile()`	`globals()`	`map()`	`reversed()`	`__import__()`
`complex()`	`hasattr()`	`max()`	`round()`	

https://docs.python.org/es/3/library/functions.html?highlight=built

Archivos CSV

Trabajar con archivos CSV (valores separados por comas) es una tarea común en el procesamiento de datos en Python. CSV es un formato de archivo que se utiliza para almacenar datos tabulares en forma de texto plano, donde cada línea del archivo representa una fila y los valores de cada columna están separados por comas.

Para trabajar con archivos CSV en Python, podemos utilizar el módulo `csv` que proporciona funciones y clases para leer y escribir archivos CSV.

Archivos CSV: Abrir y Leer contenido

Para leer un archivo CSV, primero debemos importar el módulo csv y luego utilizar la función reader() para crear un objeto lector de CSV. Luego, podemos iterar sobre las filas del archivo utilizando un bucle for y acceder a los valores de cada columna.

```
import csv

with open('datos.csv', 'r') as archivo:
    lector_csv = csv.reader(archivo)
    for fila in lector_csv:
        print(fila)
```

```
import csv

with open('datos.csv', 'r') as archivo:
    lector_csv = csv.reader(archivo)
    encabezados = next(lector_csv)
    for fila in lector_csv:
        print(fila)
```

Si el archivo CSV tiene encabezados de columna, podemos utilizar el método next() del lector para omitir la primera línea que contiene los encabezados

Archivos CSV: Escribir contenido

Para escribir en un archivo CSV, podemos utilizar la función writer() para crear un objeto escritor de CSV. Luego, podemos utilizar el método writerow() para escribir una fila en el archivo.

```python
import csv

datos = [
    ['Juan', 'Perez', '25'],
    ['Maria', 'Gomez', '30'],
    ['Pedro', 'Lopez', '28']
]

with open('datos.csv', 'w', newline='') as archivo:
    escritor_csv = csv.writer(archivo)
    for fila in datos:
        escritor_csv.writerow(fila)
```

Proyecto CSV

Tienda de libros.

Eres el dueño de una tienda de libros y deseas analizar las ventas mensuales para determinar cuál es el libro más vendido cada mes. Tienes un archivo CSV que contiene la información de las ventas en cada línea con el siguiente formato: "mes, libro, cantidad".

PIP y las librerías externas

pip es el sistema de gestión de paquetes estándar para Python. Es una herramienta que facilita la instalación, actualización y gestión de librerías y paquetes de software en Python. Pip se utiliza para descargar paquetes desde el Python Package Index (PyPI).

Antes de utilizar pip, primero debemos verificar que está instalado, para ello utilizamos el comando pip --versión en una terminal del sistema, este comando nos mostrará la versión actual de pip.

- Instalar una librería: pip install nombre_librería.
- Actualizar una librería: pip install --upgrade nombre_librería
- Desinstalar una librería: pip uninstall nombre_libreria

Instalación de Python y **pip**: https://www.youtube.com/watch?v=GLcbD-mCAO8

PIP y las librerías externas

Librerías de Python más importantes en diferentes áreas

Análisis de datos y ciencia de datos:

- Pandas: Librería para el análisis y manipulación de datos estructurados.
- NumPy: Librería para la computación científica y manipulación de matrices y arreglos.
- Matplotlib: Librería para la visualización de datos en gráficos y gráficas.
- SciPy: Librería para la computación científica y resolución de problemas matemáticos.

Aprendizaje automático (Machine Learning):

- Scikit-learn: Librería para el aprendizaje automático y minería de datos.
- TensorFlow: Librería de aprendizaje automático y redes neuronales de código abierto.
- Keras: Librería de alto nivel para la construcción y entrenamiento de modelos de aprendizaje profundo.
- PyTorch: Librería de aprendizaje automático basada en tensores y redes neuronales.

PIP y las librerías externas

Librerías de Python más importantes en diferentes áreas

Procesamiento de imágenes y visión por computadora:

- OpenCV: Librería para el procesamiento de imágenes y visión por computadora.
- Pillow: Librería para el procesamiento de imágenes y manipulación de formatos de imagen.
- scikit-image: Librería para el procesamiento de imágenes y extracción de características.

Desarrollo web:

- Flask: Microframework para el desarrollo de aplicaciones web.
- Django: Framework web completo y de alto nivel.
- Requests: Librería para realizar peticiones HTTP.

Archivos Excel

Existen varias librerías populares para trabajar con archivos Excel en Python, estas son algunas de las mas conocidas:

- openpyxl: Permite leer, escribir y manipular archivos Excel (.xlsx)

  Instalación: pip install openpyxl
- xlsxwriter: Librería para crear y escribir archivos Excel en formato .xlsx

  Instalación: pip install xlsxwriter
- pandas: Muy utilizada para análisis y manipulación de datos

  Instalación: pip install pandas

Archivos Excel con openpyxl

Primer paso instalar openpyxl: pip install openpyxl

Leer datos de un archivo Excel:

```python
import openpyxl

# Abrir el archivo Excel
libro = openpyxl.load_workbook('datos.xlsx')

# Seleccionar una hoja
hoja = libro['Hoja1']

# Leer datos de celdas
valor_celda = hoja['A1'].value
print(valor_celda)

# Iterar sobre filas
for fila in hoja.iter_rows(min_row=2, values_only=True):
    print(fila)
```

Archivos Excel con openpyxl

Escribir datos en un archivo Excel:

```python
import openpyxl

# Crear un nuevo libro de Excel
libro = openpyxl.Workbook()

# Seleccionar una hoja
hoja = libro.active

# Escribir datos en celdas
hoja['A1'] = 'Nombre'
hoja['B1'] = 'Edad'
hoja['A2'] = 'Juan'
hoja['B2'] = 25

# Guardar el libro como un nuevo archivo
libro.save('nuevo.xlsx')
```

Archivos Excel con openpyxl

Modificar datos existentes en un archivo Excel:

```python
import openpyxl

# Abrir el archivo Excel
libro = openpyxl.load_workbook('datos.xlsx')

# Seleccionar una hoja
hoja = libro['Hoja1']

# Modificar el valor de una celda
hoja['A1'].value = 'Nuevo valor'

# Guardar los cambios en el archivo
libro.save('datos_modificados.xlsx')
```

Referencias bibliográficas

Documentación oficial Python.

https://docs.python.org/es/3/tutorial/index.html

https://docs.python.org/es/3/library/exceptions.html#exception-context

https://docs.python.org/es/3/library/exceptions.html#concrete-exceptions

https://docs.python.org/es/3/library/functions.html#open

https://docs.python.org/es/3/glossary.html#term-file-object

Referencias bibliográficas

Librerías Python.

Openpyxl: https://openpyxl.readthedocs.io/en/stable/

Xlsxwriter: https://xlsxwriter.readthedocs.io/