Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Ciencia de los Datos

Periodo 2024B

Laboratorio 1

Alumno:

Alvaro Jesus Castro Pizaña

Profesor:

Dr. Asdrúbal López Chau

Zumpango, Estado de México a 21 del 08 de 2024

laboratorio1

August 20, 2024

0.1 Explicación de Bibliotecas

0.1.1 1. `pandas`

`pandas` es una biblioteca de Python que generalmente se usa para la manipulacion y analisis de datos. Ademas, permite trabajar con `DataFrames`.

Ejemplo de uso:

```
import pandas as pd

# crear un df
data = {'Nombre': ['Ana', 'Juan', 'Pedro'], 'Edad': [23, 35, 29]}
df = pd.DataFrame(data)

# imprimir las primeras 5 filas del df
print(df.head())
```

0.1.2 2. `requests`

`requests` es una biblioteca que permite realizar peticiones HTTP y se usa generalmente para pedir datos de APIs y procesarlos.

Ejemplo de uso:

```
import requests

# realizar una peticion
url = 'https://jsonplaceholder.typicode.com/users'
respuesta = requests.get(url)

# si la respuesta fue OK
if respuesta.status_code == 200:
    data = respuesta.json()
    print(data)
else:
    print(f"Error en la solicitud: {respuesta.status_code}")
```

0.1.3 3. `sqlite3`

`sqlite3` permite interactuar con bases de datos SQLite desde Python y facilita el manejo de BBDD sin la necesidad de configurar un servidor.

Ejemplo de uso:

```
import sqlite3

# conexion a una bd
conn = sqlite3.connect('mi_base_datos.db')
cursor = conn.cursor()

# crear una tabla
cursor.execute('''CREATE TABLE IF NOT EXISTS usuarios (id INTEGER PRIMARY KEY, nombre TEXT, eda
cursor.execute("INSERT INTO usuarios (nombre, edad) VALUES ('Pedro', 30)")

# guardar los cambios
conn.commit()
# Consultar datos
cursor.execute("SELECT * FROM usuarios")
print(cursor.fetchall())

conn.close() # Cerrar la conexion
```

0.1.4 4. beautifulsoup4

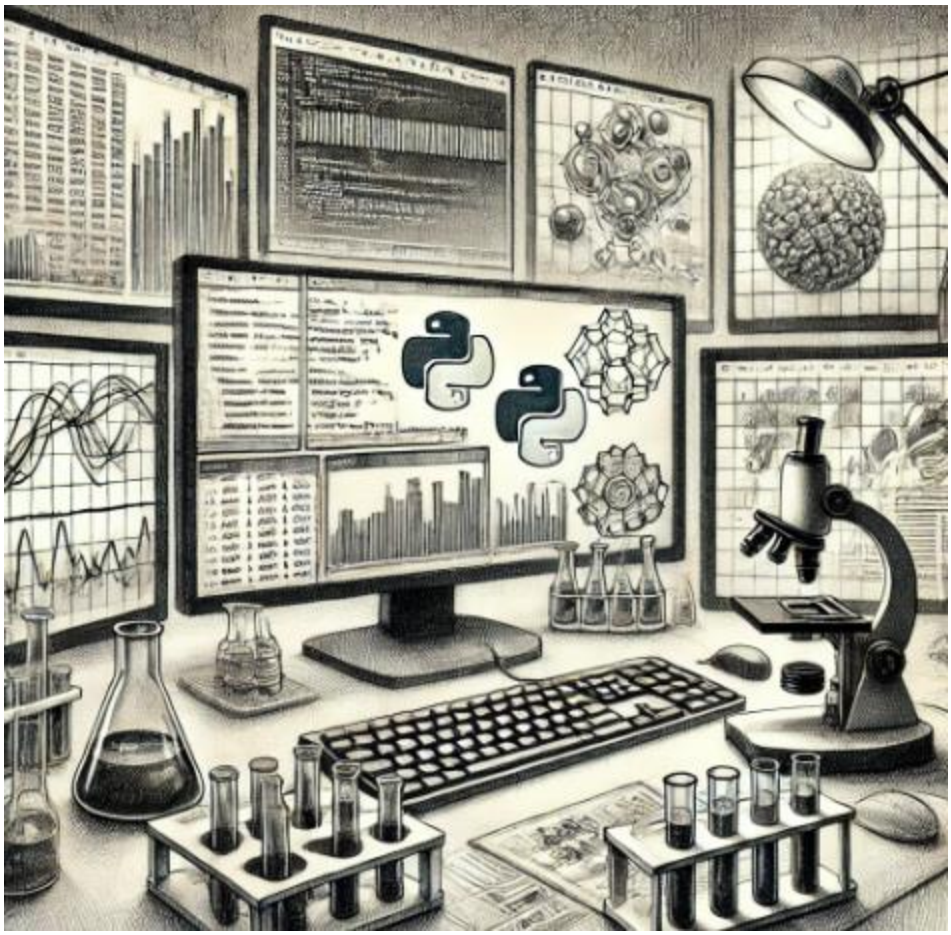
`beautifulsoup4` es utilizada para hacer web scraping, lo que significa extraer datos de sitios web al analizar y manipular elementos.

Ejemplo de uso:

```
from bs4 import BeautifulSoup
import requests

# obtener contenido HTML de una pag web
url = 'https://example.com'
respuesta = requests.get(url)
html_contenido = respuesta.content

# se analiza el contenido HTML
sopa = BeautifulSoup(html_contenido, 'html.parser')

# se extrae el titulo
titulo = sopa.title.string
print(f"Titulo de la pagina: {titulo}")
```

1 Inicio del laboratorio

[4]:
```
!pip install ucimlrepo
```



[5]:
```
# librerias que se utilizaran
import pandas as pd
from ucimlrepo import fetch_ucirepo
import requests
from bs4 import BeautifulSoup
```

1.0.1 Ejemplo de cómo cargar los datos de iris.data de UCI repository

[2]:
```
# se busca por el id correspondiente
iris = fetch_ucirepo(id=53)
# se obtienen los respectivos datos y categorias
X = iris.data.features
y = iris.data.targets
print(X.head())
```

```
   sepal length  sepal width  petal length  petal width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
```

```
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
```

1.0.2 Importar el conjunto de datos iris desde un archivo local

[3]:
```
# desde un archivo
ruta = "./datos/iris/iris.data"
# sin cabecera
df_iris = pd.read_csv(ruta, header=None)
# con nombres
nombres = ["sepal_length","sepal_width","petal_length","petal_width","class"]
df_iris.columns = nombres
print(df_iris.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

1.0.3 Importar el conjunto de datos Bank Marketing desde un archivo local Bank marketing

[9]:
```
#aqui debe ir la ruta donde estan los datos
ruta_bank = "C:/Users/Hp245-User/Desktop/datos/bank_marketing/bank-full.csv"
df_bank = pd.read_csv(ruta_bank, delimiter=";")
print(df_bank.head())
```

```
   age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
3   47   blue-collar  married    unknown      no     1506     yes   no
4   33       unknown   single    unknown      no        1      no   no

   contact  day month  duration  campaign  pdays  previous poutcome   y
0  unknown    5   may       261         1     -1         0  unknown  no
1  unknown    5   may       151         1     -1         0  unknown  no
2  unknown    5   may        76         1     -1         0  unknown  no
3  unknown    5   may        92         1     -1         0  unknown  no
4  unknown    5   may       198         1     -1         0  unknown  no
```

1.0.4 Obtener datos JSON desde una API

[10]:
```
# se establece la uri y se hace la solicitud
url = 'https://jsonplaceholder.typicode.com/users'
respuesta = requests.get(url)
```

```
"""si la respuesta es OK o 200 se hace construye el df
o simplemente se imprime el tipo de error"""
if respuesta.status_code == 200:
    data = respuesta.json()
    df_jsonplaceholder = pd.DataFrame(data)
    print(df_jsonplaceholder.head())
else:
    print(f"Error al obtener los datos {respuesta.status_code}")
```

```
   id              name   username                      email  \
0   1     Leanne Graham       Bret          Sincere@april.biz
1   2      Ervin Howell  Antonette          Shanna@melissa.tv
2   3  Clementine Bauch   Samantha         Nathan@yesenia.net
3   4  Patricia Lebsack   Karianne  Julianne.OConner@kory.org
4   5  Chelsey Dietrich     Kamren   Lucio_Hettinger@annie.ca

                                             address                  phone  \
0  {'street': 'Kulas Light', 'suite': 'Apt. 556',…  1-770-736-8031 x56442
1  {'street': 'Victor Plains', 'suite': 'Suite 87…    010-692-6593 x09125
2  {'street': 'Douglas Extension', 'suite': 'Suit…         1-463-123-4447
3  {'street': 'Hoeger Mall', 'suite': 'Apt. 692',…      493-170-9623 x156
4  {'street': 'Skiles Walks', 'suite': 'Suite 351…          (254)954-1289

         website                                            company
0  hildegard.org  {'name': 'Romaguera-Crona', 'catchPhrase': 'Mu…
1  anastasia.net  {'name': 'Deckow-Crist', 'catchPhrase': 'Proac…
2    ramiro.info  {'name': 'Romaguera-Jacobson', 'catchPhrase': …
3       kale.biz  {'name': 'Robel-Corkery', 'catchPhrase': 'Mult…
4   demarco.info  {'name': 'Keebler LLC', 'catchPhrase': 'User-c…
```

1.0.5 Descargar y mostrar contenido de un libro desde Gutenberg

[11]:
```
"""
Nuevamente se hace la solicitud y se muestra si todo sale bien, caso contrario␣
 ↪de avisa el status code
status code !=200
"""
uri = "https://www.gutenberg.org/cache/epub/60198/pg60198.txt"
respuesta_libro = requests.get(uri)
if respuesta_libro.status_code == 200:
    contenido = respuesta_libro.content
    print(contenido[:500])  # Mostramos los primeros 500 caracteres del␣
 ↪contenido
else:
    print(f"Error al obtener el contenido: {respuesta_libro.status_code}")
```

```
b'\xef\xbb\xbfThe Project Gutenberg eBook of Fuente Ovejuna\r\n    \r\nThis
ebook is for the use of anyone anywhere in the United States and\r\nmost other
```

parts of the world at no cost and with almost no restrictions\r\nwhatsoever. You may copy it, give it away or re-use it under the terms\r\nof the Project Gutenberg License included with this ebook or online\r\nat www.gutenberg.org. If you are not located in the United States,\r\nyou will have to check the laws of the country where you are located\r\nbefore using this eB'

1.0.6 Crear un DataFrame de 10 libros del sitio Gutenberg

[12]:
```
# primero busqué 10 libros con sus uris y titulos
libros = {
    "https://www.gutenberg.org/cache/epub/74279/pg74279.txt": "Lukukammio",
    "https://www.gutenberg.org/cache/epub/74278/pg74278.txt": "Tuonen ahventa␣
 ↪onkimassa",
    "https://www.gutenberg.org/cache/epub/74277/pg74277.txt": "Beauty and the␣
 ↪beast : An old tale new-told, with pictures",
    "https://www.gutenberg.org/cache/epub/74276/pg74276.txt": "The Door",
    "https://www.gutenberg.org/cache/epub/74275/pg74275.txt": "The Story of the␣
 ↪Champions of the Round Table",
    "https://www.gutenberg.org/cache/epub/74274/pg74274.txt": "The Blue␣
 ↪Balloon",
    "https://www.gutenberg.org/cache/epub/74273/pg74273.txt": "The Invisible␣
 ↪Man",
    "https://www.gutenberg.org/cache/epub/74272/pg74272.txt": "The Black Cat",
    "https://www.gutenberg.org/cache/epub/74271/pg74271.txt": "The Great␣
 ↪Gatsby",
    "https://www.gutenberg.org/cache/epub/74270/pg74270.txt": "Moby Dick"
}

# por cada libro se hace su solicitud
datos = []
for url, title in libros.items():
    response = requests.get(url)
    # si fue exitoso añadimos el contenido a la lista de datos o el codigo de␣
 ↪error
    if response.status_code == 200:
        datos.append([response.text, title])
    else:
        datos.append([f"Error {response.status_code}", title])
# al final solo queda añadir en el df añadir
df_libros = pd.DataFrame(datos, columns=["Texto", "Título"])
print(df_libros.head())
```

```
                                               Texto  \
0  The Project Gutenberg eBook of Lukukammio\r\n…
1  The Project Gutenberg eBook of Tuonen ahventa…
2  The Project Gutenberg eBook of Beauty and the…
3  The Project Gutenberg eBook of Arthur's inher…
```

```
4  The Project Gutenberg eBook of Theology in ro…

                                                 Título
0                                            Lukukammio
1                              Tuonen ahventa onkimassa
2  Beauty and the beast : An old tale new-told, w…
3                                             The Door
4      The Story of the Champions of the Round Table
```

1.1 Tipos de Datos Estructurados y No Estructurados

1.1.1 1. Datos Estructurados

Los **datos estructurados** son aquellos que están organizados en un formato predefinido y fácilmente interpretable por máquinas. Un ejemplo típico son las bases de datos relacionales, donde los datos se organizan en tablas con filas y columnas. Estos datos suelen tener un esquema fijo, como nombres de columnas y tipos de datos específicos (por ejemplo, enteros, cadenas de texto, etc.).

1.1.2 2. Datos No Estructurados

Los **datos no estructurados** no siguen un formato específico y, por lo tanto, son más difíciles de organizar, procesar y analizar. Estos incluyen archivos de texto, correos electrónicos, imágenes, videos, audios y publicaciones en redes sociales.

1.2 ¿Por qué es conveniente usar `pandas` para Ciencia de Datos?

`pandas` es una de las bibliotecas más importantes ya que permite la manipulación de datos en Python.

Una de sus principales caracteristicas son su alta **facilidad al usar** grandes volúmenes de datos. Pues el tipo `DataFrames` permiten manejar datos tabulares de manera eficiente y realizar operaciones complejas con poco código. Otra ventaje es que se pueden realizar **operaciones** como filtrar, agrupar, agregar y transformar datos. Lo que lo hace ideal para realizar análisis , preparar datos o manipular grandes conjuntos de datos de forma rápida. Y por ultimo se permite una gran cantidad de **formatos** ya que se puede leer y escribir datos desde/para una amplia gama de formatos, incluidos CSV, Excel, SQL, JSON, etc.

1.3 ¿Cómo se pueden leer datos desde diversas fuentes y tipos de archivos?

`Pandas` generalmete tiene diversas formas de poder realizar la lectura de archivos diferentes y hasta ahora hemos visto la funcion que permite leer de uno de los formatos de datos más utilizados. Cargando archivos CSV utilizando `pd.read_csv()`, lo que convierte el archivo en un `DataFrame` para su manipulación.

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

[ ]: