Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Ciencia de los Datos

Periodo 2024B

Práctica: Laboratorio recuperación de datos

Alumno: Eric Carmen Soto

Profesor: Dr. Asdrubal López Chau

Zumpango, Estado de México, a 20 de agosto de 2024

Laboratorio1

August 21, 2024

1 Universidad Autónoma del Estado de México

1.1 Centro Universitario UAEM Zumpango

1.1.1 Ingeniería en Computación

UA: Ciencia de los Datos **Profesor:** Dr. Asdrubal López Chau

Alumno: Eric Carmen Soto

Fecha: 20 de Agosto de 2024

Reporte de: Laboratorio: Recuperación de datos

2 Creación y Manipulación de DataFrames con Pandas

2.1 1. Importar la Biblioteca Pandas

Primero, necesitamos importar la biblioteca `pandas` para trabajar con DataFrames.

[137]:
```
import pandas as pd
```

2.2 2. Crear un DataFrame

A continuación, vamos a crear un DataFrame con datos ficticios. `Un DataFrame es una estructura de datos bidimensional que puede almacenar datos de diferentes tipos.`

El código `df = pd.DataFrame(data)` crea un nuevo DataFrame en pandas usando los datos que se proporcionan en `data`.

[138]:
```
# Crear un DataFrame
data = {
    'Nombre': ['Ana', 'Luis', 'Pedro'],
    'Edad': [28, 34, 45]
}
df = pd.DataFrame(data)
```

2.3 3. Impresion del DataFrame

[139]:
```
# Mostrar el DataFrame
print(df)
print("\n\n\n\n")
```

```
  Nombre  Edad
0    Ana    28
1   Luis    34
2  Pedro    45
```

2.4 4. Leer archivo CSV

Un archivo CSV (Comma-Separated Values, en inglés) es un formato de archivo utilizado para almacenar datos tabulares en texto plano. En un archivo CSV, cada línea representa una fila de la tabla, y los valores en cada fila están separados por comas (o a veces por otros delimitadores, como punto y coma `;`).

Características clave de un archivo CSV:

1. **Formato de Texto Plano:** Los datos se almacenan en formato de texto sin formato, lo que facilita su visualización y edición con un editor de texto.
2. **Separadores:** Los valores en cada fila están separados por un delimitador, que por lo general es una coma `,`. Sin embargo, en algunos casos puede ser un punto y coma `;` o un tabulador `\t`.
3. **Encabezados:** La primera línea de un archivo CSV suele contener los nombres de las columnas, lo que facilita la comprensión de los datos.
4. **Compatibilidad:** Los archivos CSV son ampliamente compatibles con muchos programas de software, incluidos programas de hojas de cálculo como Microsoft Excel, Google Sheets, y herramientas de análisis de datos como pandas en Python.

- `pd.read_csv('bank.csv', delimiter=';')`: Lee un archivo CSV llamado `bank.csv`, usando el punto y coma (`;`) como delimitador de valores, y carga los datos en un DataFrame de pandas.
- `print(df.head())`: Muestra las primeras cinco filas del DataFrame `df` para visualizar una muestra rápida de los datos cargados.

Para mostrar más filas o toda la información del DataFrame, se puede usar:

- `df.head(n)`: Donde `n` es el número de filas que quieres ver. Por ejemplo, `df.head(10)` muestra las primeras 10 filas.
- `df.tail(n)`: Muestra las últimas `n` filas del DataFrame.

- `print(df)`: Imprime todo el DataFrame, aunque esto puede no ser ideal para archivos grandes.

[140]:
```
# Leer un archivo CSV
df = pd.read_csv('bank.csv', delimiter=';')
print(df.head())
```

```
   age          job  marital  education default  balance housing loan  \
0   30   unemployed  married    primary      no     1787      no   no
1   33     services  married  secondary      no     4789     yes  yes
2   35   management   single   tertiary      no     1350     yes   no
3   30   management  married   tertiary      no     1476     yes  yes
4   59  blue-collar  married  secondary      no        0     yes   no

    contact  day month  duration  campaign  pdays  previous poutcome   y
0  cellular   19   oct        79         1     -1         0  unknown  no
1  cellular   11   may       220         1    339         4  failure  no
2  cellular   16   apr       185         1    330         1  failure  no
3   unknown    3   jun       199         4     -1         0  unknown  no
4   unknown    5   may       226         1     -1         0  unknown  no
```

3 Realización de Solicitudes HTTP con Requests

3.1 1. Importar la Biblioteca Requests

Primero, necesitamos importar la biblioteca requests para realizar solicitudes HTTP.

[141]:
```
import requests
```

3.2 2. Realizar una Solicitud GET

3.3 ¿Qué es una API?

API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que diferentes aplicaciones se comuniquen entre sí. Una API define cómo se deben solicitar y recibir datos entre un cliente y un servidor. En otras palabras, es un puente que facilita la interacción entre software.

3.4 ¿Qué es un Endpoint?

Un **endpoint** es una URL específica en una API donde se puede acceder a ciertos datos o servicios. Cada endpoint representa un recurso particular o una acción en la API. Por ejemplo, en la API de GitHub, un endpoint podría ser `/users` para obtener información sobre los usuarios.

3.5 ¿Qué Hace el Método GET de Requests?

El método **GET** de la biblioteca `requests` se utiliza para solicitar datos de un servidor. Cuando haces una solicitud GET a un endpoint, estás pidiendo al servidor que te envíe la información correspondiente a ese recurso o acción. En el caso de la API pública de GitHub, al hacer una solicitud GET a `https://api.github.com`, estás solicitando información sobre la API de GitHub.

3.5.1 La informacion de esta API `https://api.github.com` se guarda en la variable `response`

[142]:
```
response = requests.get('https://api.github.com')
```

- `print(response.status_code)`: Muestra el código de estado HTTP de la respuesta, que indica el resultado de la solicitud (por ejemplo, 200 para éxito, 404 para no encontrado).

[143]:
```
print(response.status_code)  # Código de estado
```

200

- `print(response.json())`: Muestra el contenido de la respuesta en formato JSON, que suele ser un diccionario o lista de datos.

[144]:
```
print(response.json())       # Contenido JSON de la respuesta
```


```
{'current_user_url': 'https://api.github.com/user',
'current_user_authorizations_html_url':
'https://github.com/settings/connections/applications{/client_id}',
'authorizations_url': 'https://api.github.com/authorizations',
'code_search_url':
'https://api.github.com/search/code?q={query}{&page,per_page,sort,order}',
'commit_search_url':
'https://api.github.com/search/commits?q={query}{&page,per_page,sort,order}',
'emails_url': 'https://api.github.com/user/emails', 'emojis_url':
'https://api.github.com/emojis', 'events_url': 'https://api.github.com/events',
'feeds_url': 'https://api.github.com/feeds', 'followers_url':
'https://api.github.com/user/followers', 'following_url':
'https://api.github.com/user/following{/target}', 'gists_url':
'https://api.github.com/gists{/gist_id}', 'hub_url':
'https://api.github.com/hub', 'issue_search_url':
'https://api.github.com/search/issues?q={query}{&page,per_page,sort,order}',
'issues_url': 'https://api.github.com/issues', 'keys_url':
'https://api.github.com/user/keys', 'label_search_url': 'https://api.github.com/
search/labels?q={query}&repository_id={repository_id}{&page,per_page}',
'notifications_url': 'https://api.github.com/notifications', 'organization_url':
'https://api.github.com/orgs/{org}', 'organization_repositories_url':
'https://api.github.com/orgs/{org}/repos{?type,page,per_page,sort}',
'organization_teams_url': 'https://api.github.com/orgs/{org}/teams',
'public_gists_url': 'https://api.github.com/gists/public', 'rate_limit_url':
'https://api.github.com/rate_limit', 'repository_url':
'https://api.github.com/repos/{owner}/{repo}', 'repository_search_url': 'https:/
/api.github.com/search/repositories?q={query}{&page,per_page,sort,order}',
'current_user_repositories_url':
'https://api.github.com/user/repos{?type,page,per_page,sort}', 'starred_url':
'https://api.github.com/user/starred{/owner}{/repo}', 'starred_gists_url':
'https://api.github.com/gists/starred', 'topic_search_url':
'https://api.github.com/search/topics?q={query}{&page,per_page}', 'user_url':
'https://api.github.com/users/{user}', 'user_organizations_url':
```

```
'https://api.github.com/user/orgs', 'user_repositories_url':
'https://api.github.com/users/{user}/repos{?type,page,per_page,sort}',
'user_search_url':
'https://api.github.com/search/users?q={query}{&page,per_page,sort,order}'}
```

4 Uso de SQLite en Python

4.1 1. Importar la Biblioteca SQLite

Primero, necesitamos importar la biblioteca `sqlite3`, que proporciona una interfaz para trabajar con bases de datos SQLite en Python.

[145]:
```
import sqlite3
```

4.2 2. Conectar a una Base de Datos

A continuación, nos conectamos a una base de datos SQLite. Si la base de datos no existe, `sqlite3` la creará automáticamente.

[146]:
```
conn = sqlite3.connect('mibdatos.db')
```

4.3 3. Crear un Cursor

Un **cursor** es un objeto que permite ejecutar comandos SQL en la base de datos. Usamos el cursor para interactuar con la base de datos.

[147]:
```
cursor = conn.cursor()
```

4.4 4. Crear una Tabla

Creamos una tabla llamada `usuarios` si no existe ya. La tabla tiene tres columnas: `id` (clave primaria que se incrementa automáticamente), `nombre` (texto) y `edad` (entero).

[148]:
```
cursor.execute('''CREATE TABLE IF NOT EXISTS usuarios (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    nombre TEXT,
                    edad INTEGER)''')
```

[148]:
```
<sqlite3.Cursor at 0x21e08c9b6c0>
```

4.5 5. Insertar Datos

Insertamos dos registros en la tabla `usuarios`. Cada registro tiene un `nombre` y una `edad`.

[149]:
```
cursor.execute('''INSERT INTO usuarios (nombre, edad)
                  VALUES ('Ana', 28), ('Luis', 34)''')
```

[149]:
```
<sqlite3.Cursor at 0x21e08c9b6c0>
```

4.6 6. Consultar Datos

Consultamos todos los datos de la tabla `usuarios` y mostramos el resultado. `fetchall()` devuelve una lista de tuplas, cada una representando una fila de la tabla.

[150]:
```
cursor.execute('SELECT * FROM usuarios')
print(cursor.fetchall())
```

```
[(1, 'Ana', 28), (2, 'Luis', 34), (3, 'Ana', 28), (4, 'Luis', 34), (5, 'Ana',
28), (6, 'Luis', 34)]
```

4.7 7. Cerrar la Conexión

Finalmente, guardamos los cambios realizados en la base de datos con `commit()` y cerramos la conexión.

[151]:
```
conn.commit()
conn.close()
```

5 Extracción de Datos Web con BeautifulSoup

5.1 1. Importar las Bibliotecas

Primero, importamos las bibliotecas necesarias: `BeautifulSoup` para el análisis HTML y `requests` para realizar solicitudes HTTP.

[152]:
```
from bs4 import BeautifulSoup
import requests
```

5.2 2. Hacer una Solicitud GET

Realizamos una solicitud HTTP GET al sitio web que queremos analizar. En este caso, solicitamos la página de `https://es.wikipedia.org`.

[153]:
```
response = requests.get('https://es.wikipedia.org')
```

5.3 3. Analizar gramaticalmente el Contenido HTML

Convertimos el contenido HTML de la respuesta en un objeto BeautifulSoup para facilitar su análisis y manipulación. Usamos el analizador `html.parser`.

[154]:
```
soup = BeautifulSoup(response.text, 'html.parser')
```

5.4 4. Extraer y Mostrar el Título de la Página

Obtenemos el título de la página HTML y lo mostramos. `soup.title.text` extrae el texto del elemento `<title>` de la página.

[155]:
```
print(soup.title.text)
```

```
Wikipedia, la enciclopedia libre
```

5.5 5. Extraer Todos los Enlaces

Buscamos todos los elementos `<a>` en la página, que generalmente representan enlaces. Luego, imprimimos el valor del atributo **`href`** de cada enlace, que contiene la URL a la que apunta el enlace.

[156]:
```
for link in soup.find_all('a'):
    print(link.get('href'))
```

```
#bodyContent
/wiki/Wikipedia:Portada
/wiki/Portal:Comunidad
/wiki/Portal:Actualidad
/wiki/Especial:CambiosRecientes
/wiki/Especial:P%C3%A1ginasNuevas
/wiki/Especial:Aleatoria
/wiki/Ayuda:Contenidos
//donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&utm_m
edium=sidebar&utm_campaign=C13_es.wikipedia.org&uselang=es
/wiki/Wikipedia:Informes_de_error
/wiki/Wikipedia:Portada
/wiki/Especial:Buscar
/w/index.php?title=Especial:Crear_una_cuenta&returnto=Wikipedia%3APortada
/w/index.php?title=Especial:Entrar&returnto=Wikipedia%3APortada
/w/index.php?title=Especial:Crear_una_cuenta&returnto=Wikipedia%3APortada
/w/index.php?title=Especial:Entrar&returnto=Wikipedia%3APortada
/wiki/Ayuda:Introducci%C3%B3n
/wiki/Especial:MisContribuciones
/wiki/Especial:MiDiscusi%C3%B3n
/wiki/Wikipedia:Portada
/wiki/Wikipedia_discusi%C3%B3n:Portada
/wiki/Wikipedia:Portada
/w/index.php?title=Wikipedia:Portada&action=edit
/w/index.php?title=Wikipedia:Portada&action=history
/wiki/Wikipedia:Portada
/w/index.php?title=Wikipedia:Portada&action=edit
/w/index.php?title=Wikipedia:Portada&action=history
/wiki/Especial:LoQueEnlazaAqu%C3%AD/Wikipedia:Portada
/wiki/Especial:CambiosEnEnlazadas/Wikipedia:Portada
//commons.wikimedia.org/wiki/Special:UploadWizard?uselang=es
/wiki/Especial:P%C3%A1ginasEspeciales
/w/index.php?title=Wikipedia:Portada&oldid=149134229
/w/index.php?title=Wikipedia:Portada&action=info
/w/index.php?title=Especial:Acortador_de_URL&url=https%3A%2F%2Fes.wikipedia.org%
2Fwiki%2FWikipedia%3APortada
/w/index.php?title=Especial:QrCode&url=https%3A%2F%2Fes.wikipedia.org%2Fwiki%2FW
ikipedia%3APortada
https://www.wikidata.org/wiki/Special:EntityPage/Q5296
/w/index.php?title=Especial:Libro&bookcmd=book_creator&referer=Wikipedia%3APorta
```

da
/w/index.php?title=Especial:DownloadAsPdf&page=Wikipedia%3APortada&action=show-download-screen
/w/index.php?title=Wikipedia:Portada&printable=yes
https://commons.wikimedia.org/wiki/Main_Page
https://foundation.wikimedia.org/wiki/Home
https://www.mediawiki.org/wiki/MediaWiki
https://meta.wikimedia.org/wiki/Main_Page
https://outreach.wikimedia.org/wiki/Main_Page
https://wikisource.org/wiki/Main_Page
https://species.wikimedia.org/wiki/Main_Page
https://es.wikibooks.org/wiki/Portada
https://www.wikidata.org/wiki/Wikidata:Main_Page
https://www.wikifunctions.org/wiki/Wikifunctions:Main_Page
https://wikimania.wikimedia.org/wiki/Wikimania
https://es.wikinews.org/wiki/Portada
https://es.wikiquote.org/wiki/Portada
https://es.wikisource.org/wiki/Portada
https://es.wikiversity.org/wiki/Portada
https://es.wikivoyage.org/wiki/P%C3%A1gina_principal
https://es.wiktionary.org/wiki/Wikcionario:Portada
/wiki/Wikipedia:Bienvenidos
/wiki/Ayuda:Introducci%C3%B3n
/wiki/Especial:Buscar
/wiki/Especial:Estad%C3%ADsticas
/wiki/Wikipedia_en_espa%C3%B1ol
/wiki/Wikipedia:Caf%C3%A9
/wiki/Ayuda:C%C3%B3mo_puedes_colaborar
/wiki/Ayuda:Introducci%C3%B3n
/wiki/Ayuda:Contenidos
/wiki/Wikipedia:Contacto
/wiki/Urbanismo_de_Barcelona
/wiki/Archivo:Plano_BCN.jpg
/wiki/Historia_de_Barcelona
/wiki/Ciudad
/wiki/Espacio_p%C3%BAblico
/wiki/Arquitectura_de_Barcelona
/wiki/Parques_y_jardines_de_Barcelona
/wiki/Barcelona
/wiki/Antigua_Roma
/wiki/Ensanche_de_Barcelona
/wiki/Agregaciones_municipales_de_Barcelona
/wiki/Fortaleza_de_la_Ciudadela
/wiki/Llano_de_Barcelona
/wiki/Plan_Cerd%C3%A1
/wiki/Ildefonso_Cerd%C3%A1
/wiki/Antiguos_municipios_de_Barcelona
/wiki/Plan_Jaussely

/wiki/Exposici%C3%B3n_Universal_de_Barcelona_(1888)
/wiki/Exposici%C3%B3n_Internacional_de_Barcelona_(1929)
/wiki/XXXV_Congreso_Eucar%C3%ADstico_Internacional
/wiki/Juegos_Ol%C3%ADmpicos_de_Barcelona_1992
/wiki/F%C3%B3rum_Universal_de_las_Culturas_2004
/wiki/Universidad_de_Harvard
/wiki/Medalla_de_Oro_del_RIBA
/wiki/Real_Instituto_de_Arquitectos_Brit%C3%A1nicos
/wiki/Urbanismo_de_Barcelona
/wiki/Wikipedia:Art%C3%ADculos_destacados/%C3%8Dndice
/wiki/Wikipedia:Candidatos_a_art%C3%ADculos_destacados
/wiki/Golpe_de_Estado_de_Korn%C3%ADlov
/wiki/Benjamin_Britten
/wiki/Rent-A-Girlfriend
/wiki/Up_All_Night:_The_Live_Tour
/wiki/Idioma_espa%C3%B1ol
/wiki/DVD
/wiki/Boy_band
/wiki/Reino_Unido
/wiki/Irlanda
/wiki/One_Direction
/w/index.php?title=Bournemouth_International_Centre&action=edit&redlink=1
/wiki/Bournemouth
/wiki/Gira_musical
/wiki/Up_All_Night_Tour
/wiki/Sony_Music
/wiki/What_Makes_You_Beautiful
/wiki/Gotta_Be_You
/wiki/One_Thing
/wiki/Up_All_Night:_The_Live_Tour
/wiki/Wikipedia:Art%C3%ADculos_buenos/%C3%8Dndice
/wiki/Wikipedia:Selecci%C3%B3n_de_art%C3%ADculos_buenos/nominaciones
/wiki/Peugeot_206_WRC
/wiki/Talbot_Samba
/wiki/Anexo:%C3%81lbumes_tributo_a_Madonna
/wiki/Wikipedia:Recurso_del_d%C3%ADa
/wiki/Archivo:Miraflores_Locks_Panama_1914.JPG
/wiki/Esclusas_del_Canal_de_Panam%C3%A1
/wiki/Canal_de_Panam%C3%A1
/wiki/Wikipedia:Recurso_del_d%C3%ADa
/wiki/Wikipedia:Portal
/wiki/Archivo:Nuvola_apps_package_graphics.svg
/wiki/Portal:Arte
/wiki/Portal:Arquitectura
/wiki/Portal:Cine
/wiki/Portal:Danza
/wiki/Portal:Literatura
/wiki/Portal:M%C3%BAsica

/wiki/Portal:M%C3%BAsica_cl%C3%A1sica
/wiki/Portal:Pintura
/wiki/Portal:Teatro
/wiki/Archivo:Sciences_humaines.svg
/wiki/Portal:Ciencias_humanas_y_sociales
/wiki/Portal:Comunicaci%C3%B3n
/wiki/Portal:Deporte
/wiki/Portal:Derecho
/wiki/Portal:Econom%C3%ADa
/wiki/Portal:Feminismo
/wiki/Portal:Filosof%C3%ADa
/wiki/Portal:LGBT
/wiki/Portal:Ling%C3%BC%C3%ADstica
/wiki/Portal:Psicolog%C3%ADa
/wiki/Portal:Sociolog%C3%ADa
/wiki/Archivo:Science-symbol-2.svg
/wiki/Portal:Ciencias_naturales
/wiki/Portal:Astronom%C3%ADa
/wiki/Portal:Biolog%C3%ADa
/wiki/Portal:Bot%C3%A1nica
/wiki/Portal:F%C3%ADsica
/wiki/Portal:Medicina
/wiki/Portal:Matem%C3%A1tica
/wiki/Portal:Qu%C3%ADmica
/wiki/Archivo:Ambox_globe.svg
/wiki/Portal:Geograf%C3%ADa
/wiki/Portal:%C3%81frica
/wiki/Portal:Am%C3%A9rica
/wiki/Portal:Ant%C3%A1rtida
/wiki/Portal:Asia
/wiki/Portal:Europa
/wiki/Portal:Ocean%C3%ADa
/wiki/Portal:Pa%C3%ADses
/wiki/Archivo:History2.svg
/wiki/Portal:Historia
/wiki/Portal:Prehistoria
/wiki/Portal:Edad_Antigua
/wiki/Portal:Edad_Media
/wiki/Portal:Edad_Moderna
/wiki/Portal:Edad_Contempor%C3%A1nea
/wiki/Archivo:Vote3_final.png
/wiki/Portal:Pol%C3%ADtica
/wiki/Portal:Marxismo
/wiki/Portal:Nacionalismo
/wiki/Portal:Socialismo
/wiki/Portal:Terrorismo
/wiki/Archivo:P_religion_world.svg
/wiki/Portal:Religi%C3%B3n

/wiki/Portal:Ate%C3%ADsmo
/wiki/Portal:Budismo
/wiki/Portal:Cristianismo
/wiki/Portal:Iglesia_cat%C3%B3lica
/wiki/Portal:Islam
/wiki/Portal:Juda%C3%ADsmo
/wiki/Portal:Mitolog%C3%ADa
/wiki/Archivo:Tecno-rueda.svg
/wiki/Portal:Tecnolog%C3%ADa
/wiki/Portal:Biotecnolog%C3%ADa
/wiki/Portal:Exploraci%C3%B3n_espacial
/wiki/Portal:Inform%C3%A1tica
/wiki/Portal:Ingenier%C3%ADa
/wiki/Portal:Software_libre
/wiki/Portal:Videojuegos
/wiki/Portal:Actualidad
/wiki/Guerra_Israel-Gaza_(2023-presente)
/wiki/Invasi%C3%B3n_rusa_de_Ucrania
/wiki/Brote_de_viruela_s%C3%ADmica_de_2024
/wiki/Protestas_en_Venezuela_de_2024
/wiki/Elecciones_presidenciales_de_Venezuela_de_2024
/wiki/Torneo_de_Cleveland_2024
/wiki/Torneo_de_Monterrey_2024
/wiki/Torneo_de_Winston-Salem_2024
/wiki/Convenci%C3%B3n_Nacional_Dem%C3%B3crata_de_2024
/wiki/Campeonato_Mundial_de_Remo_de_2024
/wiki/Copa_Panamericana_de_Voleibol_Femenino_de_2024
/wiki/Vuelta_a_Espa%C3%B1a_2024
/wiki/Campeonato_Mundial_de_Voleibol_Femenino_Sub-17_de_2024
/wiki/Anexo:Fallecidos_en_2024
/wiki/Humberto_Maschio
/wiki/Rato_Tvrdi%C4%87
/wiki/Atsuko_Tanaka
/wiki/Archivo:Maria_Branyas_Morera_(117%C3%A8_aniversari).jpg
/wiki/Maria_Branyas_Morera
/wiki/Archivo:Klaus_Dockhorn_DDR-Schwimmer.jpg
/wiki/Klaus_Dockhorn
/wiki/Somaya_Ramadan
/wiki/Lalo_Gomes
/wiki/N%C3%A9stor_Salvador_Quintana
/wiki/Michel_Gu%C3%A9rard_(cocinero)
/wiki/Phil_Donahue
/wiki/Diletta_D%27Andrea
/wiki/Franciszek_Smuda
/wiki/Alain_Delon
/wiki/Johnny_%22Dandy%22_Rodr%C3%ADguez_Jr
/wiki/Zhou_Guangzhao
/wiki/Pierre_Cartier

/wiki/Clara_Mar%C3%ADa_Gonz%C3%A1lez_de_Amez%C3%BAa
/wiki/D%C3%ADa_Internacional_de_Conmemoraci%C3%B3n_de_las_V%C3%ADctimas_de_Actos_de_Violencia_Basados_en_la_Religi%C3%B3n_o_las_Creencias
/wiki/D%C3%ADa_Internacional_de_Conmemoraci%C3%B3n_y_Homenaje_a_las_V%C3%ADctimas_del_Terrorismo
/wiki/Categor%C3%ADa:Actualidad
/wiki/2024
/wiki/Categor%C3%ADa:2024
/wiki/Portal:Actualidad
/wiki/Categor%C3%ADa:Actualidad
/wiki/2024
/wiki/Categor%C3%ADa:2024
/wiki/Plantilla:Portada:Actualidad
/wiki/21_de_agosto
/wiki/Archivo:Hafsat_Abiola_Nigerian_activist.jpg
/wiki/1924
/wiki/Jack_Weston
/wiki/1974
/wiki/Amy_Fisher
/wiki/1974
/wiki/Hafsat_Abiola
/wiki/1974
/wiki/James_Patrick_Cannon
/wiki/1974
/wiki/Stadionul_Municipal_(Br%C4%83ila)
/wiki/20_de_agosto
/wiki/21_de_agosto
/wiki/22_de_agosto
/wiki/Plantilla:Efem%C3%A9rides
https://meta.wikimedia.org/wiki/Special:MyLanguage/Wikimedia_projects
/wiki/Fundaci%C3%B3n_Wikimedia
https://commons.wikimedia.org/wiki/Portada
https://commons.wikimedia.org/wiki/Portada
https://es.wiktionary.org/wiki/Wikcionario:Portada
https://es.wiktionary.org/wiki/Wikcionario:Portada
https://www.wikidata.org/wiki/Wikidata:Portada
https://www.wikidata.org/wiki/Wikidata:Portada
https://es.wikibooks.org/wiki/Portada
https://es.wikibooks.org/wiki/Portada
https://es.wikinews.org/wiki/Portada
https://es.wikinews.org/wiki/Portada
https://es.wikiquote.org/wiki/Portada
https://es.wikiquote.org/wiki/Portada
https://es.wikisource.org/wiki/Portada
https://es.wikisource.org/wiki/Portada
https://species.wikimedia.org/wiki/Portada
https://species.wikimedia.org/wiki/Portada
https://es.wikiversity.org/wiki/Portada

https://es.wikiversity.org/wiki/Portada
https://es.wikivoyage.org/wiki/es:P%C3%A1gina_principal
https://es.wikivoyage.org/wiki/es:P%C3%A1gina_principal
https://meta.wikimedia.org/wiki/Portada/Es
https://meta.wikimedia.org/wiki/Portada/es
https://es.wikipedia.org/w/index.php?title=Wikipedia:Portada&oldid=149134229
https://an.wikipedia.org/wiki/
https://ar.wikipedia.org/wiki/
https://ast.wikipedia.org/wiki/
https://ay.wikipedia.org/wiki/
https://bg.wikipedia.org/wiki/
https://bpy.wikipedia.org/wiki/
https://bs.wikipedia.org/wiki/
https://ca.wikipedia.org/wiki/
https://cbk-zam.wikipedia.org/wiki/
https://ceb.wikipedia.org/wiki/
https://ch.wikipedia.org/wiki/
https://cs.wikipedia.org/wiki/
https://da.wikipedia.org/wiki/
https://de.wikipedia.org/wiki/
https://el.wikipedia.org/wiki/
https://en.wikipedia.org/wiki/
https://eo.wikipedia.org/wiki/
https://et.wikipedia.org/wiki/
https://eu.wikipedia.org/wiki/
https://ext.wikipedia.org/wiki/
https://fa.wikipedia.org/wiki/
https://fi.wikipedia.org/wiki/
https://fr.wikipedia.org/wiki/
https://gl.wikipedia.org/wiki/
https://gn.wikipedia.org/wiki/
https://he.wikipedia.org/wiki/
https://hr.wikipedia.org/wiki/
https://hu.wikipedia.org/wiki/
https://id.wikipedia.org/wiki/
https://it.wikipedia.org/wiki/
https://ja.wikipedia.org/wiki/
https://ko.wikipedia.org/wiki/
https://la.wikipedia.org/wiki/
https://lad.wikipedia.org/wiki/
https://lmo.wikipedia.org/wiki/
https://lt.wikipedia.org/wiki/
https://ms.wikipedia.org/wiki/
https://mwl.wikipedia.org/wiki/
https://nah.wikipedia.org/wiki/
https://new.wikipedia.org/wiki/
https://nl.wikipedia.org/wiki/
https://nn.wikipedia.org/wiki/

```
https://no.wikipedia.org/wiki/
https://oc.wikipedia.org/wiki/
https://pap.wikipedia.org/wiki/
https://pl.wikipedia.org/wiki/
https://pt.wikipedia.org/wiki/
https://qu.wikipedia.org/wiki/
https://ro.wikipedia.org/wiki/
https://ru.wikipedia.org/wiki/
https://simple.wikipedia.org/wiki/
https://sk.wikipedia.org/wiki/
https://sl.wikipedia.org/wiki/
https://sr.wikipedia.org/wiki/
https://sv.wikipedia.org/wiki/
https://te.wikipedia.org/wiki/
https://th.wikipedia.org/wiki/
https://tl.wikipedia.org/wiki/
https://tr.wikipedia.org/wiki/
https://uk.wikipedia.org/wiki/
https://vi.wikipedia.org/wiki/
https://zh.wikipedia.org/wiki/
https://es.wikipedia.org/wiki/Wikipedia:Texto_de_la_Licencia_Creative_Commons_At
ribuci%C3%B3n-CompartirIgual_4.0_Internacional
https://creativecommons.org/licenses/by-sa/4.0/deed.es
https://foundation.wikimedia.org/wiki/Policy:Terms_of_Use/es
https://foundation.wikimedia.org/wiki/Policy:Privacy_policy/es
https://wikimediafoundation.org/es/
https://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy:Privacy_policy/e
s
/wiki/Wikipedia:Acerca_de
/wiki/Wikipedia:Limitaci%C3%B3n_general_de_responsabilidad
https://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy:Universal_Code_o
f_Conduct
https://developer.wikimedia.org
https://stats.wikimedia.org/#/es.wikipedia.org
https://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy:Cookie_statement
/es
//es.m.wikipedia.org/w/index.php?title=Wikipedia:Portada&mobileaction=toggle_vie
w_mobile
https://wikimediafoundation.org/
https://www.mediawiki.org
```

Este código muestra cómo usar BeautifulSoup y `requests` para hacer una solicitud web, analizar el HTML recibido, y extraer información útil como el título de la página y los enlaces presentes en ella.

6 IRIS

1. Entra al sitio de UCI Repository y dedica un momento a explorarlo: https://archive.ics.uci.edu/
2. Localiza el conjunto de datos "iris" en UCI Repository.
3. Busca en la página cómo importar conjuntos directamente de UCI Repository con Python.
4. Importa el conjunto de datos iris desde https://archive.ics.uci.edu/.
5. Imprime el conjunto de datos.

[157]:
```
import pandas as pd

path = './iris/iris.data'
df = pd.read_csv(path, header=None)#si hubiese un encabezado sería  df = pd.
 ↪read_csv(path)

#Agregamos nombres a las comlumnas:
names = ['sepal_length', 'sepal_width', 'petal_lenght', 'petal_width', 'class']
df.columns = names
print(df)
```

```
     sepal_length  sepal_width  petal_lenght  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
..            …            …             …            …             …
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

7 Bank

1. Dirígete al sitio de UCI Repository: https://archive.ics.uci.edu/
2. Localiza el conjunto de datos "Bank Marketing" en UCI Repository.
3. Descarga el conjunto de datos.
4. Importa el conjunto de datos en Python.
5. Imprime el conjunto de datos.

[158]:
```
import pandas as pd

path = './bank+marketing/bank/bank-full.csv'


# Lee el archivo CSV especificando el delimitador y el carácter de comillas
datos = pd.read_csv(path, delimiter=';', quotechar='"')

# Agrega nombres a las columnas si es necesario
names = ['edad', 'empleo', 'estado_civil', 'educación', 'impago', 'saldo',␣
 ↪'hipoteca', 'préstamo', 'contacto', 'día', 'mes', 'duración', 'campaña',␣
 ↪'pdays', 'previous', 'poutcome', 'suscripción']
datos.columns = names

# Imprime las primeras filas del DataFrame
print(datos)
```

```
       edad        empleo estado_civil  educación impago  saldo hipoteca  \
0        58    management      married   tertiary     no   2143       yes
1        44    technician       single  secondary     no     29       yes
2        33  entrepreneur      married  secondary     no      2       yes
3        47   blue-collar      married    unknown     no   1506       yes
4        33       unknown       single    unknown     no      1        no
…     …           …           …         …    …       …
45206    51    technician      married   tertiary     no    825        no
45207    71       retired     divorced    primary     no   1729        no
45208    72       retired      married  secondary     no   5715        no
45209    57   blue-collar      married  secondary     no    668        no
45210    37  entrepreneur      married  secondary     no   2971        no

      préstamo   contacto  día  mes  duración  campaña  pdays  previous  \
0           no    unknown    5  may       261        1     -1         0
1           no    unknown    5  may       151        1     -1         0
2          yes    unknown    5  may        76        1     -1         0
3           no    unknown    5  may        92        1     -1         0
4           no    unknown    5  may       198        1     -1         0
…        …         …  …  …       …      …    …       …
45206       no   cellular   17  nov       977        3     -1         0
45207       no   cellular   17  nov       456        2     -1         0
45208       no   cellular   17  nov      1127        5    184         3
45209       no  telephone   17  nov       508        4     -1         0
45210       no   cellular   17  nov       361        2    188        11

      poutcome suscripción
0      unknown          no
1      unknown          no
2      unknown          no
3      unknown          no
```

```
4      unknown          no
…        …         …
45206  unknown         yes
45207  unknown         yes
45208  success         yes
45209  unknown          no
45210    other          no

[45211 rows x 17 columns]
```

8 Para convertir un archivo JSON en un DataFrame de pandas, puedes usar el siguiente código:

1. **Importación de bibliotecas**:
   - `import requests`: Importa la biblioteca `requests`, que permite hacer solicitudes HTTP para obtener datos de una URL.
   - `import pandas as pd`: Importa la biblioteca `pandas`, que es útil para el manejo y análisis de datos en Python.
2. **Primer bloque de código**:
   - `url = 'https://jsonplaceholder.typicode.com/users'`: Define la URL de la API desde la cual se obtendrán los datos. En este caso, se trata de una API pública de ejemplo que proporciona datos de usuarios.
   - `respuesta = requests.get(url)`: Realiza una solicitud GET a la URL especificada para obtener datos.
   - `print(respuesta.json())`: Imprime la respuesta de la solicitud en formato JSON, lo que permite ver los datos obtenidos de la API.
3. **Segundo bloque de código**:
   - `respuesta = requests.get(url)`: Realiza nuevamente una solicitud GET a la misma URL para obtener datos.
   - `if respuesta.status_code == 200`: Verifica si la solicitud fue exitosa. El código de estado 200 indica que la solicitud fue procesada correctamente.
   - `datos = respuesta.json()`: Convierte la respuesta en formato JSON a un diccionario de Python para facilitar el manejo de los datos.
   - `df = pd.DataFrame(datos)`: Convierte el diccionario en un DataFrame de `pandas`, una estructura de datos tabular que facilita el análisis y manipulación de datos.
   - `print(df.head())`: Imprime las primeras filas del DataFrame para mostrar una vista previa de los datos.
   - `else: print(f"Error al obtener los datos: {respuesta.status_code}")`: Imprime un mensaje de error con el código de estado si la solicitud no fue exitosa.

[162]:
```python
import requests

url = 'https://jsonplaceholder.typicode.com/users'
respuesta = requests.get(url)
print(respuesta.json())

import pandas as pd
```

```
respuesta = requests.get(url)
if respuesta.status_code == 200:
    datos = respuesta.json()
    df = pd.DataFrame(datos)
    print(df.head())
else:
    print(f"Error al obtener los datos: {respuesta.status_code}")
```

[{'id': 1, 'name': 'Leanne Graham', 'username': 'Bret', 'email':
'Sincere@april.biz', 'address': {'street': 'Kulas Light', 'suite': 'Apt. 556',
'city': 'Gwenborough', 'zipcode': '92998-3874', 'geo': {'lat': '-37.3159',
'lng': '81.1496'}}, 'phone': '1-770-736-8031 x56442', 'website':
'hildegard.org', 'company': {'name': 'Romaguera-Crona', 'catchPhrase': 'Multi-
layered client-server neural-net', 'bs': 'harness real-time e-markets'}}, {'id':
2, 'name': 'Ervin Howell', 'username': 'Antonette', 'email':
'Shanna@melissa.tv', 'address': {'street': 'Victor Plains', 'suite': 'Suite
879', 'city': 'Wisokyburgh', 'zipcode': '90566-7771', 'geo': {'lat': '-43.9509',
'lng': '-34.4618'}}, 'phone': '010-692-6593 x09125', 'website': 'anastasia.net',
'company': {'name': 'Deckow-Crist', 'catchPhrase': 'Proactive didactic
contingency', 'bs': 'synergize scalable supply-chains'}}, {'id': 3, 'name':
'Clementine Bauch', 'username': 'Samantha', 'email': 'Nathan@yesenia.net',
'address': {'street': 'Douglas Extension', 'suite': 'Suite 847', 'city':
'McKenziehaven', 'zipcode': '59590-4157', 'geo': {'lat': '-68.6102', 'lng':
'-47.0653'}}, 'phone': '1-463-123-4447', 'website': 'ramiro.info', 'company':
{'name': 'Romaguera-Jacobson', 'catchPhrase': 'Face to face bifurcated
interface', 'bs': 'e-enable strategic applications'}}, {'id': 4, 'name':
'Patricia Lebsack', 'username': 'Karianne', 'email':
'Julianne.OConner@kory.org', 'address': {'street': 'Hoeger Mall', 'suite': 'Apt.
692', 'city': 'South Elvis', 'zipcode': '53919-4257', 'geo': {'lat': '29.4572',
'lng': '-164.2990'}}, 'phone': '493-170-9623 x156', 'website': 'kale.biz',
'company': {'name': 'Robel-Corkery', 'catchPhrase': 'Multi-tiered zero tolerance
productivity', 'bs': 'transition cutting-edge web services'}}, {'id': 5, 'name':
'Chelsey Dietrich', 'username': 'Kamren', 'email': 'Lucio_Hettinger@annie.ca',
'address': {'street': 'Skiles Walks', 'suite': 'Suite 351', 'city':
'Roscoeview', 'zipcode': '33263', 'geo': {'lat': '-31.8129', 'lng': '62.5342'}},
'phone': '(254)954-1289', 'website': 'demarco.info', 'company': {'name':
'Keebler LLC', 'catchPhrase': 'User-centric fault-tolerant solution', 'bs':
'revolutionize end-to-end systems'}}, {'id': 6, 'name': 'Mrs. Dennis Schulist',
'username': 'Leopoldo_Corkery', 'email': 'Karley_Dach@jasper.info', 'address':
{'street': 'Norberto Crossing', 'suite': 'Apt. 950', 'city': 'South Christy',
'zipcode': '23505-1337', 'geo': {'lat': '-71.4197', 'lng': '71.7478'}}, 'phone':
'1-477-935-8478 x6430', 'website': 'ola.org', 'company': {'name': 'Considine-
Lockman', 'catchPhrase': 'Synchronised bottom-line interface', 'bs': 'e-enable
innovative applications'}}, {'id': 7, 'name': 'Kurtis Weissnat', 'username':
'Elwyn.Skiles', 'email': 'Telly.Hoeger@billy.biz', 'address': {'street': 'Rex
Trail', 'suite': 'Suite 280', 'city': 'Howemouth', 'zipcode': '58804-1099',
'geo': {'lat': '24.8918', 'lng': '21.8984'}}, 'phone': '210.067.6132',

```
'website': 'elvis.io', 'company': {'name': 'Johns Group', 'catchPhrase':
'Configurable multimedia task-force', 'bs': 'generate enterprise e-tailers'}},
{'id': 8, 'name': 'Nicholas Runolfsdottir V', 'username': 'Maxime_Nienow',
'email': 'Sherwood@rosamond.me', 'address': {'street': 'Ellsworth Summit',
'suite': 'Suite 729', 'city': 'Aliyaview', 'zipcode': '45169', 'geo': {'lat':
'-14.3990', 'lng': '-120.7677'}}, 'phone': '586.493.6943 x140', 'website':
'jacynthe.com', 'company': {'name': 'Abernathy Group', 'catchPhrase':
'Implemented secondary concept', 'bs': 'e-enable extensible e-tailers'}}, {'id':
9, 'name': 'Glenna Reichert', 'username': 'Delphine', 'email':
'Chaim_McDermott@dana.io', 'address': {'street': 'Dayna Park', 'suite': 'Suite
449', 'city': 'Bartholomebury', 'zipcode': '76495-3109', 'geo': {'lat':
'24.6463', 'lng': '-168.8889'}}, 'phone': '(775)976-6794 x41206', 'website':
'conrad.com', 'company': {'name': 'Yost and Sons', 'catchPhrase': 'Switchable
contextually-based project', 'bs': 'aggregate real-time technologies'}}, {'id':
10, 'name': 'Clementina DuBuque', 'username': 'Moriah.Stanton', 'email':
'Rey.Padberg@karina.biz', 'address': {'street': 'Kattie Turnpike', 'suite':
'Suite 198', 'city': 'Lebsackbury', 'zipcode': '31428-2261', 'geo': {'lat':
'-38.2386', 'lng': '57.2232'}}, 'phone': '024-648-3804', 'website':
'ambrose.net', 'company': {'name': 'Hoeger LLC', 'catchPhrase': 'Centralized
empowering task-force', 'bs': 'target end-to-end models'}}]
   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…
```

9 Libro Fuente Ovejuna

1. Acceder al sitio https://www.gutenberg.org.
2. Una vez ahí, se busca esta obra: “Fuente Ovejuna”, autor: Lope de Vega (puedes elegir otra).
3. Se scribe el código de Python para descargar el contenido del libro.

[160]:
```python
import requests

# URL del libro en formato de texto plano
url = "https://www.gutenberg.org/cache/epub/60198/pg60198.txt"

# Realizar la solicitud HTTP para obtener el contenido del libro
response = requests.get(url)

# Verificar que la solicitud fue exitosa
if response.status_code == 200:
    # Almacenar el contenido del libro como una cadena
    book_content = response.text
    # Dividir el contenido en líneas
    lines = book_content.splitlines()

    # Imprimir las primeras 100 líneas
    for line in lines[:100]:
        print(line)
   # print(book_content)#imprime todo el libro


    """
    # Imprimir las primeras 500 líneas (o todo el contenido si hay menos de 200 líneas)
    for line in lines[:500]:
        print(line)
    """
else:
    print(f"Error {response.status_code}: No se pudo descargar el libro.")
```

```
The Project Gutenberg eBook of Fuente Ovejuna

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

Title: Fuente Ovejuna

Author: Lope de Vega

Release date: August 30, 2019 [eBook #60198]
                Most recently updated: August 28, 2020

Language: Spanish
```

Credits: Roberto Marabini, Carlos Colon, and the Online Distributed Proofreading Team (http://www.pgdp.net) from page images generously made available by Internet Archive (https://archive.org)

E-text prepared by Roberto Marabini, Carlos Colon, and the Online Distributed Proofreading Team (http://www.pgdp.net) from page images generously made available by Internet Archive (https://archive.org)

Note: Images of the original pages are available through
Internet Archive. See
https://archive.org/details/fuenteovejunacom00vega

Nota del Transcriptor:

Las letras itálicas se denotan con el caracter de _subrayado_.

Las versalitas (letras mayúsculas de tamaño igual a las minúsculas) han sido sustituidas por letras mayúsculas de tamaño normal

Coleccion Universal
N.os 5 y 6

LOPE DE VEGA

UCOMEDIA

Precio: Una peseta

MADRID, 1919

LOPE DE VEGA

```
FUENTE OVEJUNA

Comedia

Edición revisada por Américo Castro


[Ilustración]




Madrid, 1919

Talleres "Calpe", Ríos Rosas, 24.--MADRID


_Como tantas otras comedias de nuestro teatro, la presente--publicada
en 1619--se basa en un hecho histórico: en 1476, los vecinos de Fuente
Ovejuna (Córdoba) tomaron venganza, en la persona del comendador de la
Orden de Calatrava, de las vejaciones inhumanas que aquél les hacía
padecer. La crónica de Rades y Andrada narra prolijamente la justicia
hecha por los aldeanos y la pesquisa ordenada por los Reyes Católicos._

_Culmina en este drama el espíritu rústico y primitivo, que animó los
ternas populares, muy activos dentro del teatro de nuestro siglo_
XVII, _siglo próximo a la Edad Media tanto por su pensamiento como
por la forma de sentir la vida. Otras comedias de Lope poetizan la
caballerosidad o la altivez aldeana_ (El alcalde de Zalamea, Peribáñez,
El mejor alcalde, el rey, El villano en su rincón, etc.); FUENTE
OVEJUNA, _empero, tiene como héroe a toda una villa, cuya fisonomía va
concretándose en una firme progresión y acaba por adquirir tremenda e
indivisible personalidad_.
```

10 Data Frame de 10 libros

1. Acceder al sitio https://www.gutenberg.org y seleccionar 10 libros que desees incluir en tu DataFrame.
2. Descargar el contenido de los libros seleccionados.
3. Se escribe el código de Python para crear un DataFrame con los datos descargados. La primera columna del DataFrame debe contener el texto del libro, y la segunda columna debe contener el título del libro.
4. **Importación de bibliotecas**:

- `import requests`: Importa la biblioteca `requests`, que permite hacer solicitudes HTTP para obtener datos de una URL.
- `import pandas as pd`: Importa la biblioteca `pandas`, que es útil para el manejo y análisis de datos en Python.

5. **Lista de URLs de libros**:
   - `book_urls`: Una lista de URLs que apunta a los archivos de texto de libros en el sitio de Project Gutenberg.
6. **Crear listas para almacenar los textos y los títulos**:
   - `texts`: Lista para almacenar el contenido de los libros.
   - `titles`: Lista para almacenar los títulos de los libros.
7. **Descargar el contenido de cada libro y extraer el título**:
   - `for url in book_urls`: Recorre cada URL en la lista `book_urls`.
   - `response = requests.get(url)`: Realiza una solicitud GET para obtener el contenido del libro desde la URL.
   - `if response.status_code == 200`: Verifica si la solicitud fue exitosa (código de estado 200).
     - `book_content = response.text`: Almacena el contenido del libro como una cadena de texto.
     - `start_index = book_content.find('Title: ')`: Encuentra la posición inicial de la línea que contiene el título del libro.
     - `end_index = book_content.find('\n', start_index)`: Encuentra la posición final de la línea del título.
     - `title = book_content[start_index + len('Title: '):end_index].strip()`: Extrae y limpia el título del libro.
     - `texts.append(book_content)`: Añade el contenido del libro a la lista `texts`.
     - `titles.append(title)`: Añade el título del libro a la lista `titles`.
   - `else`: Si la solicitud no fue exitosa.
     - `texts.append(None)`: Añade `None` a la lista `texts`.
     - `titles.append(None)`: Añade `None` a la lista `titles`.
8. **Crear el DataFrame**:
   - `df = pd.DataFrame({ "Title": titles, "Text": texts })`: Crea un DataFrame de `pandas` con dos columnas, "Title" y "Text", utilizando las listas `titles` y `texts`.
9. **Mostrar el DataFrame**:
   - `print(df)`: Imprime el DataFrame para mostrar el resultado.

[161]:
```
import requests
import pandas as pd

# Lista de URLs de libros
book_urls = [
    "https://www.gutenberg.org/cache/epub/74282/pg74282.txt",
    "https://www.gutenberg.org/cache/epub/74273/pg74273.txt",
```

```
    "https://www.gutenberg.org/cache/epub/74274/pg74274.txt",
    "https://www.gutenberg.org/cache/epub/74275/pg74275.txt",
    "https://www.gutenberg.org/cache/epub/74276/pg74276.txt",
    "https://www.gutenberg.org/cache/epub/74277/pg74277.txt",
    "https://www.gutenberg.org/cache/epub/74278/pg74278.txt",
    "https://www.gutenberg.org/cache/epub/74279/pg74279.txt",
    "https://www.gutenberg.org/cache/epub/74280/pg74280.txt",
    "https://www.gutenberg.org/cache/epub/74281/pg74281.txt",
]

# Crear listas para almacenar los textos y los títulos
texts = []
titles = []

# Descargar el contenido de cada libro y extraer el título
for url in book_urls:
    response = requests.get(url)
    if response.status_code == 200:
        # Almacenar el contenido del libro como una cadena
        book_content = response.text

        # Buscar la línea que contiene el título
        start_index = book_content.find('Title: ')
        end_index = book_content.find('\n', start_index)
        title = book_content[start_index + len('Title: '):end_index].strip()

        # Almacenar el contenido y el título en las listas
        texts.append(book_content)
        titles.append(title)
    else:
        texts.append(None)
        titles.append(None)

# Crear el DataFrame
df = pd.DataFrame({
    "Title": titles,
    "Text": texts
})

# Mostrar el DataFrame
print(df)  # Imprime las primeras filas del DataFrame para ver el resultado
```

```
                                          Title  \
0                                  I miei racconti
1                                 Jylhänmäkeläiset
2                                       Onnen tie
3  Theology in romance: or the catechism and the …
4                              Arthur's inheritance
```

```
5                                Beauty and the beast
6                             Tuonen ahventa onkimassa
7                                          Lukukammio
8  Rambles in Germany and Italy in 1840, 1842, an…
9        Seven years in South Africa volume 1 (of 2)

                                                 Text
0  The Project Gutenberg eBook of I miei raccont…
1  The Project Gutenberg eBook of Jylhänmäkeläis…
2  The Project Gutenberg eBook of Onnen tie\r\n …
3  The Project Gutenberg eBook of Theology in ro…
4  The Project Gutenberg eBook of Arthur's inher…
5  The Project Gutenberg eBook of Beauty and the…
6  The Project Gutenberg eBook of Tuonen ahventa…
7  The Project Gutenberg eBook of Lukukammio\r\n…
8  The Project Gutenberg eBook of Rambles in Ger…
9  The Project Gutenberg eBook of Seven years in…
```

11 Diferencias entre Tipos de Datos Estructurados y No Estructurados

Datos Estructurados: - **Definición:** Son datos organizados en un formato fijo, lo que facilita su entrada, consulta y análisis. Se suelen almacenar en bases de datos relacionales y tienen una estructura predefinida. - **Ejemplos:** Tablas en bases de datos SQL, hojas de cálculo, y archivos CSV. - **Características:** - **Organización:** Están organizados en tablas, con filas y columnas. - **Facilidad de Consulta:** Se pueden consultar fácilmente mediante SQL u otros lenguajes de consulta. - **Ejemplo de uso:** Un sistema de gestión de ventas donde los datos de clientes, productos y transacciones están organizados en tablas relacionadas.

Datos No Estructurados: - **Definición:** Son datos que no siguen un formato o estructura predefinida, lo que dificulta su organización y análisis automático. - **Ejemplos:** Texto libre, correos electrónicos, imágenes, videos y publicaciones en redes sociales. - **Características:** - **Desorganización:** No tienen una estructura fija y no se organizan en tablas o columnas. - **Dificultad de Consulta:** Requieren procesamiento y análisis adicionales para extraer información útil. - **Ejemplo de uso:** Documentos de texto, comentarios en redes sociales o transcripciones de audio.

11.0.1 Ventajas de Usar Pandas para Ciencia de Datos

Pandas es una biblioteca en Python muy utilizada en ciencia de datos debido a varias razones:

- **Estructuras de Datos Flexibles:** Ofrece dos estructuras de datos principales, `Series` (para datos unidimensionales) y `DataFrame` (para datos bidimensionales), que permiten manejar y analizar datos de manera eficiente.
- **Manejo de Datos Faltantes:** Facilita la limpieza de datos, incluyendo el manejo y la imputación de valores faltantes.
- **Operaciones de Datos:** Proporciona herramientas para filtrar, agregar, agrupar y transformar datos, lo que simplifica el análisis de datos complejos.

- **Integración con Otras Herramientas:** Se integra bien con otras bibliotecas de Python como NumPy, Matplotlib y SciPy, lo que permite realizar análisis numéricos y visualización de datos.
- **Facilidad de Importación y Exportación:** Permite leer y escribir datos desde diversos formatos como CSV, Excel, SQL y JSON de manera sencilla.
- **Funcionalidades Avanzadas:** Incluye funcionalidades avanzadas para manejo de fechas y horarios, operaciones de pivote y agregaciones.

11.0.2 Resumen sobre Cómo Leer Datos desde Diversas Fuentes y Tipos de Archivos

1. Archivos CSV (Comma-Separated Values): - **Uso de Pandas:** `pandas.read_csv('archivo.csv', delimiter=',')` - **Descripción:** Los archivos CSV son simples archivos de texto donde cada línea representa una fila y los valores están separados por comas (o cualquier otro delimitador).

2. Bases de Datos SQL: - **Uso de Pandas:** `pandas.read_sql_query('SELECT * FROM tabla', con=conexion)` - **Descripción:** Permite leer datos directamente desde una base de datos SQL ejecutando consultas SQL. Se necesita una conexión a la base de datos.

3. Archivos JSON (JavaScript Object Notation): - **Descripción:** Los archivos JSON contienen datos en formato de texto que usa una estructura de pares clave-valor. Son comunes para intercambiar datos en aplicaciones web.