

- **Año:** [2024]
- **Alumno/a:** [Antonella Robiolio]
- **Legajo:** [42904775]

## Archivos de texto plano

[gutenberg.org](https://www.gutenberg.org) (llamado así por el inventor de la imprenta moderna) es el sitio web del Proyecto Gutenberg que se dedica a la distribución y creación de eBooks. En este sitio se encuentra el [Don Quijote](#). El Don Quijote se puede leer en diferentes formatos, ¿no? Para la tarea, se va a usar el Don Quijote de texto plano. La siguiente celda descarga el .txt y lo guarda en el sistema para luego su futura lectura:

```
import requests
URL = "https://www.gutenberg.org/cache/epub/2000/pg2000.txt"
response = requests.get(URL)
data = response.text
print(data)
```



Mostrar el resultado oculto

Haz doble clic (o ingresa) para editar

## Consignas

**NOTA:** Para estas consignas, no se puede usar módulos externos a Python

Del archivo descargado, calcular la cantidad de líneas:

```
num_lines = len(data.splitlines())
print("El archivo tiene", num_lines, "lineas.")
```



El archivo tiene 38055 líneas.

Del archivo descargado, calcular la cantidad de veces que se dice:

- 'Quijote'
- 'mancha'
- 'españa'

Considerar que, por ejemplo, 'QuiJoTe' es lo mismo que 'quijote' (no *case sensitive*).

```
data_lower = data.lower()
palabras = ["quijote", "mancha", "españa"]
cont = {}
```

```
for palabra in palabras:
    cont[palabra] = 0
    for otra_palabra in data_lower.split():
        if palabra in otra_palabra:
            cont[palabra] += 1
```

```
for palabra, cantidad in cont.items():
    print(f"La palabra '{palabra}' aparece {cantidad} veces.")
```



La palabra 'quijote' aparece 2249 veces.  
La palabra 'mancha' aparece 196 veces.  
La palabra 'españa' aparece 78 veces.


¿Quien es el autor y titulo del libro? Responderlo usando las herramientas utilizadas en la celda anterior (Tip: abrir el archivo de texto y ver como son las primeras líneas).

Haz doble clic (o ingresa) para editar

```
titulo = ""
autor = ""
for linea in data.splitlines():
    if "Title:" in linea:
        titulo = linea.split(":")[1].strip()
    elif "Author:" in linea:
        autor = linea.split(":")[1].strip()
    if titulo and autor:
```

```
break


print("Título:", titulo)
print("Autor:", autor)
```

 Título: Don Quijote  
Autor: Miguel de Cervantes Saavedra

## ✓ Archivos .csv

*Robert* De Niro actuó en mas de 100 películas y Rotten Tomatoes es un sitio web en el que se pueden leer críticas y ratings a películas hechos por críticos y usuarios. La siguiente línea descarga un archivo .csv en el que tenemos películas de De Niro junto a su rating en Rotten Tomatoes:

```
!wget https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv
```

 --2024-11-15 23:43:26-- <https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv>  
Resolving people.sc.fsu.edu (people.sc.fsu.edu)... 144.174.0.22  
Connecting to people.sc.fsu.edu (people.sc.fsu.edu)|144.174.0.22|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2391 (2.3K) [text/csv]  
Saving to: 'deniro.csv'

```
deniro.csv          100%[=====>]    2.33K  --.-KB/s   in 0s

2024-11-15 23:43:27 (687 MB/s) - 'deniro.csv' saved [2391/2391]
```


## ✓ Consignas

**NOTA:** *Apartir de aca, se pueden usar módulos externos de Python*

Sabemos que el archivo tiene en su primera línea los 'headers'. ¿Cuáles son? Imprimirlos.

```
import csv

with open('deniro.csv', 'r') as file:
    csv_reader = csv.reader(file)
    headers = next(csv_reader)
    print(headers)
```

 ['Year', ' "Score"', ' "Title"']


Convertir la información del archivo .csv a un diccionario.

```
movies = []

with open('deniro.csv', 'r') as file:
    csv_reader = csv.DictReader(file)
    for row in csv_reader:
        movies.append(row)
```

¿Cuántas películas hay registradas en el archivo .csv?

```
num_movies = len(movies)
print(f'Hay {num_movies} películas registradas.')
```


 Hay 87 películas registradas.

Imprimir las películas mejor y peor rankeadas del archivo (imprimir sus respectivos rankings)

```
for movie in movies:
    movie[' "Score"'] = float(movie[' "Score"'].strip(''))
```

```
best_movie = max(movies, key=lambda x: x[' "Score"'])
worst_movie = min(movies, key=lambda x: x[' "Score"'])
```

```
print('Película mejor rankeada: {} con ranking {}'.format(best_movie[' "Title"', best_movie[' "Score"']))
print('Película peor rankeada: {} con ranking {}'.format(worst_movie[' "Title"', worst_movie[' "Score"']))
```

 Película mejor rankeada: "Dear America: Letters Home From Vietnam" con ranking 100.0  
Película peor rankeada: "Godsend" con ranking 4.0

## ✓ Base de datos - Relacionales

SQLite es una biblioteca de C que provee una base de datos ligera basada en disco que no requiere un proceso de servidor separado y permite acceder a la base de datos usando una variación no estándar del lenguaje de consulta SQL. Algunas aplicaciones pueden usar SQLite para almacenamiento interno. También es posible prototipar una aplicación usando SQLite y luego transferir el código a una base de datos más grande como PostgreSQL u Oracle.

Realizamos el `import` e indicamos el archivo en el que vamos a guardar la base de datos:

```
import sqlite3
conn = sqlite3.connect('db.db')
```

La siguiente línea de código agrega filas a la misma. Por favor, ejecutar la siguiente celda para la consigna:

```
!wget https://people.sc.fsu.edu/~jburkardt/data/csv/snakes_count_10000.csv
```

```
cur = conn.cursor()
cur.execute("DROP TABLE IF EXISTS t;")
cur.execute("CREATE TABLE t (GameNumber, GameLength);")
```

```
with open('snakes_count_10000.csv','r') as csv_file:
    lines = csv_file.readlines()[1:]
    lines = [l.split(',') for l in lines]
    to_db = [(l[0].strip(), l[1].strip()) for l in lines]
```

```
cur.executemany("INSERT INTO t (GameNumber, GameLength) VALUES (?, ?);", to_db)
conn.commit()
```

```
🔗 --2024-11-16 00:12:36-- https://people.sc.fsu.edu/~jburkardt/data/csv/snakes_count_10000.csv
Resolving people.sc.fsu.edu (people.sc.fsu.edu)... 144.174.0.22
Connecting to people.sc.fsu.edu (people.sc.fsu.edu)|144.174.0.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89011 (87K) [text/csv]
Saving to: 'snakes_count_10000.csv.22'

snakes_count_10000. 100%[=====] 86.92K --.-KB/s in 0.06s

2024-11-16 00:12:36 (1.34 MB/s) - 'snakes_count_10000.csv.22' saved [89011/89011]
```

## ✓ Consignas

¿Cómo luce la base de datos? Imprimirla.

```
cur.execute("SELECT * FROM t;")
rows = cur.fetchall()
for row in rows:
    print(row)
```

🔗 [Mostrar el resultado oculto](#)

¿Cuántas filas hay en la base de datos?

```
conn = sqlite3.connect('db.db')
cur = conn.cursor()
cur.execute("SELECT COUNT(*) FROM t;")
row_count = cur.fetchone()[0]
print("Número de filas en la base de datos:", row_count)
```

🔗 Número de filas en la base de datos: 10000

¿Cuál es el máximo `GameLength` y en qué `GameNumber` se realizó?

```
cur.execute("SELECT MAX(GameLength), GameNumber FROM t;")
max_game_length_row = cur.fetchone()
print(f"El máximo GameLength es {max_game_length_row[0]} y se realizó en el GameNumber {max_game_length_row[1]}")
```

🔗 El máximo GameLength es 99 y se realizó en el GameNumber 363.

## ✓ API Requests

Por lo general, conseguimos datos a través de internet. Aunque el concepto de Application Programming Interface (API) es muy amplio y muy poco concreto, en el contexto de los servidores, es un conjunto de 'endpoints' que los servidores ofrecen para poder leer/procesar/escribir información en estos mismos.

[En este repositorio](#) van a encontrar una lista de APIs públicas (pueden elegir otras que no son de ese repositorio). Para la consigna, se recomienda usar aquellas APIs que no requieren autenticación. Si se elige una API con autenticación, por favor, adjuntar las keys o pasarl las credenciales necesarias para su futura ejecución También, pueden usar varias URLs.

## ✓ Consigna

Hacer una request GET e imprimir uno de los campos de la respuesta:

```
import requests
url = "https://catfact.ninja/fact"

try:

    response = requests.get(url)
    if response.status_code == 200:
        try:
            data = response.json()
            print("Dato curioso sobre gatos:", data['fact'])
        except requests.exceptions.JSONDecodeError:
            print("Error: la respuesta no es un JSON válido.")
    else:
        print(f"Error en la solicitud: {response.status_code}")
except requests.exceptions.RequestException as e:
    print(f"Error al realizar la solicitud: {e}")
```

🔗 Dato curioso sobre gatos: Siamese kittens are born white because of the heat inside the mother's uterus before birth. This heat keeps

Hacer una request GET y, apartir de los campos en la respuesta, generar otro campo (por ejemplo, si la respuesta solo incluye un timestamp y una posición del registro, elaborar la velocidad del registro)

```
url = "https://catfact.ninja/fact"

try:
    # Realizar la solicitud GET
    response = requests.get(url)

    if response.status_code == 200:
        try:

            data = response.json()

            cat_fact = data.get('fact', '')

            word_count = len(cat_fact.split())

            print("Dato curioso sobre gatos:", cat_fact)
            print("Número de palabras en el hecho:", word_count)

        except ValueError:
            print("Error: la respuesta no es un JSON válido.")
    else:
        print(f"Error en la solicitud: {response.status_code}")
except requests.exceptions.RequestException as e:
    print(f"Error al realizar la solicitud: {e}")
```

🔗 Dato curioso sobre gatos: A cat can't climb head first down a tree because every claw on a cat's paw points the same way. To get down  
Número de palabras en el hecho: 31

## ✓ Streaming data

Ahora vamos a hacer algo muy parecido a lo anterior, pero vamos a usar APIs que nos ofrecen datos en tiempo real. Pueden encontrar APIs de este estilo [en este repositorio](#) (pueden elegir otras que no son de ese repositorio)

## ✓ Consigna

Hacer un plot con `matplotlib` de una variable de una API de datos streaming en el que el eje 'y' sea la variable y el eje 'x' el tiempo.

```
import requests
import matplotlib.pyplot as plt
from datetime import datetime
import time

def obtener_pm10(ciudad, token):
    url = f"https://api.waqi.info/feed/{ciudad}?token={token}"
    respuesta = requests.get(url)
    datos = respuesta.json()
    if datos['status'] == 'ok':
        pm10 = datos['data']['iaqi'].get('pm10', {}).get('v', None)
        return pm10
    else:
        print("Error al obtener los datos:", datos['data'])
        return None

# Reemplaza 'TU_TOKEN' con el token que obtuviste de WAQI
token = '6f3fd800cea89d7508ac36d6fe3616867c80e5f8'
ciudad = 'Buenos Aires' # Puedes cambiar a la ciudad que desees monitorear

# Listas para almacenar los datos
tiempos = []
niveles_pm10 = []

plt.ion() # Activar modo interactivo de Matplotlib
fig, ax = plt.subplots()

# Recolectar datos durante 60 segundos
tiempo_inicio = time.time()
duracion = 60 # Duración en segundos

while time.time() - tiempo_inicio < duracion:
    pm10 = obtener_pm10(ciudad, token)
    if pm10 is not None:
        tiempo_actual = datetime.now().strftime("%H:%M:%S")
        tiempos.append(tiempo_actual)
        niveles_pm10.append(pm10)

    # Limpiar y actualizar la gráfica
    ax.clear()
    ax.plot(tiempos, niveles_pm10, marker='o')
    ax.set_title(f"Niveles de PM10 en Tiempo Real - {ciudad}")
    ax.set_xlabel("Tiempo")
    ax.set_ylabel("PM10 (µg/m³)")
    plt.xticks(rotation=45)
    plt.tight_layout()

    plt.pause(1) # Pausa de 1 segundo antes de la siguiente actualización

plt.ioff() # Desactivar modo interactivo
plt.show()
```



Niveles de PM10 en Tiempo Real - Buenos Aires

