Clase04_0Archivos

April 8, 2024

0.0.1 Seminario de Lenguajes - Python

0.1 Cursada 2024

0.0.1 Archivos. Formatos JSON y CSV

1 Pensemos en las siguientes situaciones

¿Qué estructura usamos si queremos:

- guardar los puntajes cada vez que jugamos a un juego determinado?,
- tener un banco de preguntas para que cada vez que realizamos el repaso de clase pueda acotar por temas?,
- manipular los Python plus de los estudiantes por turnos?

¿Qué tienen todas estas situaciones en común?

###

Necesitamos una estructura que permita que los datos puedan **persistir** cuando la ejecución del programa finalice

2 Algunas consideraciones antes de empezar

- Lo básico: ¿qué es un **archivo**?
- ¿Cómo podemos manipular los archivos desde un programa Python?

3 Manejo de archivos

- Existen funciones predefinidas.
- Si las operaciones fallan, se levanta una **excepción**.
- Los archivos se manejan como objetos que se crean usando la función open.

4 La función open

5 Veamos algunos ejemplos

6 Caso 1: usamos modo "w"

- ¿De **qué modo** se abre este archivo? ¿Qué significa?

- Luego de la instrucción, ¿**dónde se encuentra** archivo.txt?
- ¿De qué **tipo** es archivo.txt?
- ¿Cuándo puede dar un error esta sentencia?

[ ]:
```
file1 = open('archivo.txt', 'w')
```

7 Caso 2: usamos modo "x"

- ¿De **qué modo** se abre este archivo? ¿Qué significa?
- Luego de la instrucción, ¿**dónde se encuentra** archivo.txt?
- ¿De qué **tipo** es archivo.txt?
- ¿Cuándo puede dar un error esta sentencia?

[ ]:
```
file2 = open('archivo.txt', 'x')
```

8 Caso 3: usamos modo "a"

Veremos este caso más adelante...

9 Caso 4: ¿qué modo estamos usando?

- ¿De **qué modo** se abre este archivo? ¿Qué significa?
- ¿De qué **tipo** es archivo.txt?
- ¿Cuándo puede dar un error esta sentencia?

[ ]:
```
file3 = open('archivo.txt')
```

- En realidad la función open tiene más argumentos:

```
open(file, mode='r', buffering=-1, encoding=None,
     errors=None, newline=None, closefd=True, opener=None)
```

- **encoding**: sólo para modo texto. Por defecto, la codificación establecida en las configuraciones del sistema

```
notes = open("pp.xxx", "r+", encoding="UTF-8")
```

- En este caso, ¿debería llamarse pp.txt?
- ¿Cómo se cuál es la configuración por defecto?

[ ]:
```
import locale
locale.getlocale()
```

10 ¿Qué pasa cuando el archivo está en otro directorio?

Usamos pathlib descripta en la PEP 428: una interfaz orientada a objetos que permite el manejo de rutas.

[ ]:
```
from pathlib import Path
file_muchachos = Path('ejemplos') / "clase04" / "muchachos.txt"
file_muchachos
```

11 Archivos de texto vs. binarios

[ ]:
```
the_best_song_file = open(file_muchachos,"rb")
lyrics = the_best_song_file.read()
the_best_song_file.close()
```

[ ]:
```
print(lyrics)
```

[ ]:
```
type(lyrics)
```

- El tipo **bytes** es una secuencia inmutable de bytes.
- Solo admiten caracteres ASCII.

[ ]:
```
print(lyrics.decode('UTF-8')[:228])
```

- El tipo **str** de Python utiliza el estándar Unicode para representar caracteres.
- Generalmente la codificación predeterminada es **UTF-8**.
  - UTF significa **Unicode Transformation Format**, y el **8** significa que se utilizan valores de 8 bits en la codificación.

12 Una rosa azul ..

[ ]:
```
file_rose = Path('ejemplos') / "clase04" / "rosa_azul.jpg"
rose = open(file_rose,"rb")
rose = rose.read()
```

[ ]:
```
rose
```

13 ¿Cómo almacenamos datos en un archivo?

El caso más sencillo: vamos a guardar solo texto plano.

[ ]:
```
file_muchachos = Path('ejemplos') / "clase04" / "muchachos1.txt"
```

[ ]:
```
f = open(file_muchachos, 'w')
cad1 = "De los pibes de Malvinas"
cad2 = "Que jamás olvidaré."
f.write("En Argentina nací")
f.write("Tierra del Diego y Lionel")
f.write(cad1)
print(f.write(cad2))
f.close()
```

- **write(cadena):** escribe *cadena* en el archivo y retorna cantidad de caracteres escritos.
- **close():** cierra el archivo.

14 Ahora si: abrimos con modo "a"

- ¿De **qué modo** se abre este archivo? ¿Qué significa?
- Luego de la instrucción, ¿**dónde se encuentra** archivo.txt?
- ¿De qué **tipo** es archivo.txt?
- ¿Cuándo puede dar un error esta sentencia?

[ ]:
```
f = open(file_muchachos, "a")
#f.write("*****************************")
#f.write("HOLA QUE TAL")
f.close()
```

¿Qué pasa si no existe?

[ ]:
```
file_nuevo_muchachos = Path('ejemplos') / "clase04" / "muchachosNUEVO.txt"
```

[ ]:
```
f = open(file_nuevo_muchachos, "a")
```

15 ¿Cómo leemos los datos guardados?

[ ]:
```
f = open(file_muchachos, 'r')
x = f.read(4)
print(f.read())
x
```

- **read(cantidad_bytes):** lee *cantidad_bytes* del archivo.
- Si cantidad_bytes es <0 o no está, lee hasta fin de archivo.
- Retorna "" si EOF.

16 DESAFIO 1

Tarea para el hogar: probar el siguiente ejemplo que muestran otras formas de leer caracteres desde un archivo de texto.

[ ]:
```
file_muchachos = Path('ejemplos') / "clase04" / "muchachos.txt"
```

[ ]:
```
def by_characters():
    f = open(file_muchachos,"r")
    for x in f.read():
        print(x)
    f.close()

def by_lines():
    f = open(file_muchachos,"r")
```

```
    print(f.readlines())
    f.close()

def other_by_lines():
    f = open(file_muchachos,"r")
    for line in f:
        print(line)
    f.close()
```

[ ]:
```
def main():
    print('Leo caracteres')
    by_characters()
    print('-' * 20)
    print('Leo lineas')
    by_lines()
    print('-' * 20)
    print('Otra forma')
    other_by_line()


if __name__ == "__main__":
    other_by_lines()
```

17 ¿Qué pasa si necesito guardar información que tiene una estructura?

- Pensemos en estos ejemplos:
  - Los puntajes cada vez que juego a un juego. Información tipo: nombre jugador, puntaje, fecha.
  - El banco de preguntas: tema, enunciado, respuesta correcta.
  - Los Python plus de los estudiantes por turnos: turno, nombre, apellido, num_legajo, cantidad_puntos, etc.
- En estos casos también podría usar un archivo de texto: ¿cómo se les ocurre?

18 Algunas posibilidades

```
'Tema: Photograph - Intérprete: Ed Sheeran - Año: 2015'
---
Tema: Photograph
Intérprete: Ed Sheeran
Año: 2015
---
'Photograph-Ed Sheeran-2015'
'Photograph**Ed Sheeran**2015'
```

- ¿Pros y contras?

19 Hay otras formas mejores...

20 JSON (JavaScript Object Notation)

- Es un **formato de intercambio de datos** muy popular. Por ejemplo:

```
{ "Tema": "Photograph",
  "Intérprete": "Ed Sheeran",
  "Año": 2015}
o
 [{"Tema": "Unintended",
  "Intérprete": "Muse",
  "Año": 1999},
  {"Tema": "Photograph",
  "Intérprete": "Ed Sheeran",
  "Año": 2015}]
```

- +Info
- Veamos este ejemplo: https://sampleapis.com/api-list/movies

Y éste es solo uno de los tantos ejemplos ...

21 Módulo json

- Python tiene un módulo que permite trabajar con este formato.
- Para usarlo, debemos importarlo.

[ ]:
```
import json
```

- Permite serializar objetos.
  - serializamos con: **dumps()** y **dump()**.
  - deserializamos con: **loads()** y **load()**.
- Más info en: https://docs.python.org/3/library/json.html

22 DESAFIO 2

> Queremos guardar en un archivo info de la música que más me gusta.

Vamos a generar un archivo con la siguiente estructura: - nombre de la banda o intérprete, - ¿es_solista? - ciudad y país en donde procede, - una referencia a su trabajo.

23 Empecemos por los datos

¿Qué estructura de Python les parece mejor para almacenar los datos antes de querer guardarlos en un archivo?

```
[ ]: data = [
        {"nombre": "William Campbell", "es_solista": False, "ciudad": "La Plata",␣
     ↪"pais": "Argentina", "ref": "www.instagram.com/williamcampbellok"},
        {"nombre": "Marcos Fava", "es_solista": True, "ciudad": "La Plata", "pais":␣
     ↪"Argentina", "ref": "https://open.spotify.com/intl-es/artist/
     ↪2risap4rX5qNM54Gdiiuf1"},
        {"nombre": "Ed Sheeran", "es_solista": True, "ciudad": "Halifax", "pais":␣
     ↪"Reino Unido", "ref": "https://open.spotify.com/intl-es/artist/
     ↪6eUKZXaKkcviH0Ku9w2n3V"},
        {"nombre": "INXS", "es_solista": False, "ciudad": "Sidney", "pais":␣
     ↪"Australia","ref": "https://open.spotify.com/intl-es/artist/
     ↪1eClJfHLoDI4rZe5HxzBFv?si=b-EIpUpNTqqhmM26k6mB4w"},
        {"nombre": "La Ley", "es_solista": False, "ciudad": "Santiago", "pais":␣
     ↪"Chile", "ref": "https://open.spotify.com/intl-es/artist/
     ↪1ZVoRDO29AlDXiMkRLMZSK?si=j22YWc1wQByyODfT25Md7w"},
        {"nombre": "Héroes del Silencio", "es_solista": False, "ciudad":␣
     ↪"Zaragoza", "pais": "España", "ref": "https://open.spotify.com/intl-es/
     ↪artist/3qAPxVwIQRBuz5ImPUxpZT?si=6o_JpVtBQSy44N0lzn_73A"},
        {"nombre": "Panza", "es_solista": False, "ciudad": "Buenos Aires", "pais":␣
     ↪"Argentina", "ref": "https://open.spotify.com/intl-es/artist/
     ↪5FamANEkN5PMHyr3UuvpDd?si=o7dnBnOlRbWq70s39fsBTw"},
        {"nombre": "Adele", "es_solista": True, "ciudad": "Tottenham", "pais":␣
     ↪"Reino Unido", "ref": "https://open.spotify.com/intl-es/artist/
     ↪4dpARuHxo51G3z768sgnrY?si=co6Jp1hyS76SjQfbCEY5Rw"},
        ]
```

```
[ ]: file_route = Path('ejemplos') / "clase04" / "musica.json"
     music_file = open(file_route, "w")
```

Ahora, vamos a guardar nuestros datos en el archivo:

```
[ ]: import json

     json.dump(data, music_file)
     music_file.close()
```

24 DESAFIO 3

Queremos visualizar los datos sobre mi música que previamente guardamos.

```
[ ]: #Recordemos que:
     #file_route = Path('ejemplos') / "clase04" / "musica.json"

     file = open(file_route, "r")
     dat = json.load(file)
     print(data)
```

```
file.close()
```

- ¿De qué tipo de **data**? Abramos el archivo y observemos el dato booleano: ¿algo raro?

[ ]:
```
pretty_data = json.dumps(data, indent=4)
print(pretty_data)
```

- ¿De qué tipo es **pretty_data**?

[ ]:
```
type(pretty_data)
```

25 CSV: ¿más formatos?

- CSV (Comma Separated Values).
- Es un formato muy común para importar/exportar desde/hacia hojas de cálculo y bases de datos.
- Ejemplo:

```
nombre,es_solista,ciudad,pais,referencia
William Campbell,False,La Plata,Argentina,www.instagram.com/williamcampbellok
Marcos Fava,True,La Plata,Argentina,https://open.spotify.com/intl-es/artist/2risap4rX5qNM54Gdi
Ed Sheeran,True,Halifax,Reino Unido,https://open.spotify.com/intl-es/artist/6eUKZXaKkcviH0Ku9w
INXS,False,Sidney,Australia,https://open.spotify.com/intl-es/artist/1eClJfHLoDI4rZe5HxzBFv?si=
La Ley,False,Santiago,Chile,https://open.spotify.com/intl-es/artist/1ZVoRDO29AlDXiMkRLMZSK?si=
Héroes del Silencio,False,Zaragoza,España,https://open.spotify.com/intl-es/artist/3qAPxVwIQRBu
Panza,False,Buenos Aires,Argentina,https://open.spotify.com/intl-es/artist/5FamANEkN5PMHyr3Uuv
Adele,True,Tottenham,Reino Unido,https://open.spotify.com/intl-es/artist/4dpARuHxo51G3z768sgnr
```

- +Info: https://docs.python.org/3/library/csv.html
- PEP 305

26 Datasets

- Hay muchos datasets disponibles de muchas temáticas:
- Datos de gestión de gobiernos:
  - Har un portal de datos de Argentina, de CABA, de La Plata
  - Datos del Banco mundial
  - Portal Kagle
  - IMDB
  - Y muchos más...
- Muchos son datos abiertos, pero otros... no tanto...
- ¡PRESTAR ATENCIÓN a la licencias y requisitos para su uso!
- Muchos en formato csv y otros en json

27 Nuestra música en Spotify

Vamos a trabajar con el archivo: songs_normalize.csv - Tenemos una copia de la descripción del dataset en donde se explica a qué se refiere cada campo.

[ ]:
```
import csv
file_route = Path('ejemplos') / "clase04" / "songs_normalize.csv"
```

[ ]:
```
file = open(file_route, "r")
csv_reader = csv.reader(file, delimiter=',')
```

[ ]:
```
#encabezado = csvreader.__next__()
header = next(csv_reader)
print(header)
```

[ ]:
```
file.close()
```

28 El módulo csv

- Hay que importarlo.
- **csv.reader**: crea un objeto **iterador** que nos permite recorrer las líneas del archivo.
- ¿Por qué incluimos el paŕametro **delimiter**? ¿Dialectos?

```
csv_reader = csv.reader(file, delimiter=',')
```

29 DESAFIO 4

Busquemos cuántos temas hay de Adele en el dataset de Spotify.

Comencemos nuevamente el proceso.

[ ]:
```
file = open(file_route, "r")
csv_reader = csv.reader(file, delimiter=',')

#header = csvreader.__next__()
header = next(csv_reader)
print(header)
```

[ ]:
```
for line in csv_reader:
    if line[0] == "Adele":
        print(f"{line[1]:<40} {line[17]}")
file.close()
```

- ¿Cómo se accede a cada dato?
- ¿De qué tipo son header y line?

[ ]:
```
type(header)
```

30 Otra solución ...

[ ]:
```
file = open(file_route, "r")
csv_reader = csv.reader(file, delimiter=',')

adele_songs = filter(lambda x:  x[0] == "Adele", csv_reader)
for elem in adele_songs:
        print(f"{elem[1]:<40} {elem[17]}")

file.close()
```

31 Otra forma de acceder: csv.DictReader

[ ]:
```
file = open(file_route, "r")
csv_reader = csv.DictReader(file, delimiter=',')

adele_songs = filter(lambda x:  x["artist"] == "Adele", csv_reader)

for elem in adele_songs:
        print(f"{elem['song']:<40} {elem['genre']}")

file.close()
```

32 Hagamos la siguiente modificación a nuestro código:

Agregar un título al listado: “Adele en Spotify”.

[ ]:
```
file = open(file_route, "r")
csv_reader = csv.DictReader(file, delimiter=',')
```

[ ]:
```
adele_songs = filter(lambda x:  x["artist"] == "Adele", csv_reader)
print("Adele en Spotify")
for elem in adele_songs:
        print(f"\t{elem['song']:<40} {elem['genre']}")
#file.close()
```

No nos gusta y queremos agregar una tabulación para que que quede mejor. Modifiquemos y volvamos a ejecutar...

¿Qué observamos?

33 Recordemos que ...

csv_reader es un **iterador** por lo que cuando llegamos al final de la iteración ya no vuelve a comenzar.

En un ejemplo más adelante veremos otra forma de trabajar teniendo en cuenta esto.

34 Creamos un archivo csv con nuestra música

- **csv.writer:** retorna un objeto que convierte los datos con los que trabajamos en el programa en cadenas con el formato delimitadas con el separador correspondiente.

[ ]:
```
file_route_json = Path('ejemplos') / "clase04" / "musica.json"
file_route_csv = Path('ejemplos') / "clase04" / "musica.csv"
```

[ ]:
```
#import csv
#import json

file_json = open(file_route_json)
file_csv = open(file_route_csv, "w")
```

[ ]:
```
music = json.load(file_json)
writer = csv.writer(file_csv)
```

¿Qué contiene music?¿De qué tipo es?

[ ]:
```
type(music)
```

[ ]:
```
# Grabo el encabezado con los nombres de las columnas
writer.writerow(["nombre", "es_solista", "ciudad", "pais", "referencia"])
#Grabo los datos
for elem in music:
    writer.writerow( [elem["nombre"], elem["es_solista"], elem["ciudad"],␣
 ↪elem["pais"], elem["ref"] ])

file_json.close()
file_csv.close()
```

Exploremos el archivo .... musica.csv

35 Ahora lo leemos desde Python

[ ]:
```
#Recordemos que
#file_route_csv = Path('ejemplos') / "clase04" / "musica.csv""
file_cvs = open(file_route_csv, "r")
csv_reader = csv.reader(file_cvs, delimiter=',')
for line in csv_reader:
    print(line)
file_csv.close()
```

¿Notan algo respecto a los datos? Observemos el campo booleano.

36 JSON vs. CSV

- ¿Con qué tipo de datos trabajan? (¿texto o binario?)
- ¿En qué casos usamos archivos en formato json?
- ¿En qué casos usamos archivos en formato csv?

37 La sentencia with

La sentencia with crea un objeto denominado **runtime context** o **contexto de tiempo de ejecución** que permite ejecutar un grupo de sentencias bajo el control de un **administrador de contexto** o **context manager**. ¿Qué es esto?

> Un administrador de contexto permite asignar y liberar recursos cuando se desee.

El ejemplo típico se da en el acceso a arhivos, ya que son recursos externos a nuestros programas que requieren una gestión adecuada.

[ ]:
```
with open(file_route_csv)  as file_csv:
    csv_reader = csv.reader(file_csv, delimiter=',')
    header, data = next(csv_reader), list(csv_reader)

print(header)
for line in data:
    print(line)
```

- ¿Dónde se cierra el archivo?
- ¿Por qué trabajamos data como una lista?

38 DESAFIO 5

Dado el conjunto de datos de Spotify, queremos:

1- Guardar en otro archivo, en formato json, las canciones que tienen asignado más de un género.

2- Los cinco (5) artistas con más canciones en el dataset durante el año 2019.

[ ]:
```
# Soluciñn
```

39 Exploremos el siguiente dataset de películas

[ ]:
```
file_route = Path("ejemplos") / "clase04" / "mymoviedb.csv"

with open(file_route, "r") as file_csv:
    csv_reader = csv.reader(file_csv)
    header, data = next(csv_reader), list(csv_reader)
```

[ ]:
```
header
```

[ ]:
```
data[:1]
```

40 DESAFIO 6

Queremos ver qué películas tienen más de 9900 votos.

Deberíamos trabajar con la columna Vote_Count (columna 4)

[ ]:
```
movies = list(filter(lambda x: x[4] > "9900", data))
for movie in movies[:5]:
    print(movie[1])
```

- ¿Cuál es el problema?
- ¿Por qué convertimos a list?

Probar en casa: convertir a int x[4]. **Spolier:** da error

41 DESAFIO 7

Queremos ver los idiomas de las películas.

[ ]:
```
#header
```

[ ]:
```
languages = map(lambda x: x[6], data)
print(list(languages))
```

41.1 ¿Nos interesa que se repita?

[ ]:
```
languages = set(map(lambda x: x[6], data))
```

[ ]:
```
print(languages)
```

##

Notamos que hay datos que no son correctos.

#

Más adelante veremos cómo hay otras formas de procesar estos datasets

42 DESAFIO 8

Queremos descargar el poster de las Spiderman.

[ ]:
```
data[:1]
```

[ ]:
```
spiderman = data[:1][0]
poster_spiderman = spiderman[8]
poster_spiderman
```

[ ]:
```
import requests
image_route = Path("ejemplos") / "clase04" / "poster_spiderman.jpg"

image = requests.get(poster_spiderman)
with open(image_route, 'wb') as f:
    f.write(image.content)
```

El módulo request permite realizar peticiones HTTP.

En este caso, estamos realizando un GET de un recurso ubicado en la URL dada: 'https://image.tmdb.org/t/p/original/1g0dhYtq4irTY1GPXvft6k4YLjm.jpg'

43 DESAFIO 9

Escribir un programa que permita agregar al dataset de mi música favorita, su música favorita. Para esto tener en cuenta que: - deben utilizar la sentencia with; - deben ubicar los archivos originales en una carpeta denominada "claudia_music" y los generados por ustedes en otra carpeta denomina "<su_nombre>_music". Por ejemplo:"sofía_music"; - deben comenzar con el archivo trabajado en clase y luego agregar a éste los datos correspondientes a su música; - deben trabajar con ambos formatos y reflexionar si es posible utilizar la opción"a" en la apertura del archivo.

Los que deseen pueden compartirme la solución. Los archivos también están disponibles en Drive

44 Seguimos la próxima ...