Clase4_Archivos

April 4, 2022

0.0.1 Seminario de Lenguajes - Python

0.1 Cursada 2022

0.1.1 Archivos. Formatos JSON y CSV

1 Repasamos lambda

2 Veamos el desafío de la clase anterior

Una posible solución:

[ ]:
```
cadena = "casa"
cadena_encriptada = map(lambda x: chr(ord(x) + 1), cadena)
#cadena_encriptada
```

¿cadena_encriptada es una cadena? ¿De qué tipo es?

[ ]:
```
cadena_encriptada = ''.join(cadena_encriptada)
cadena_encriptada
```

+Info

3 Otra forma: función reduce

[ ]:
```
from functools import reduce

cadena = ["a", "e", "i"]
vocales = reduce((lambda x, y: x + y), cadena)
vocales
```

[ ]:
```
producto = reduce((lambda x, y: x * y), [1, 2, 3, 4])
producto
```

4 Otro ejemplo

Sacado de https://realpython.com/python-reduce-function/

[ ]:
```
def my_add(a, b):
    result = a + b
    print(f"{a} + {b} = {result}")
    return result
```

[ ]:
```
my_add(5, 5)
```

[ ]:
```
numbers = [0, 1, 2, 3, 4]
reduce(my_add, numbers)
```

5 Les dejé algo para investigar en el video del fin de semana....

5.1 ¿Cuándo un módulo se denomina ___main___?

6 Recordemos con este ejemplo cuál es la situación:

```
#módulo funciones
def uno():
    print("uno")
    print(f"El nombre de este módulo es {__name__}")
#uno()

#uso_funciones
import funciones
funciones.uno()
```

7 El módulo ___main___

- Las instrucciones ejecutadas en el nivel de llamadas superior del intérprete, ya sea desde un script o interactivamente, se consideran parte del módulo llamado **___main___**, por lo tanto tienen su propio espacio de nombres global.

```
#módulo funciones
def uno():
    print("uno")
    print(f"El nombre de este módulo es {__name__}")

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

8 Veamos este otro ejemplo:

```
# modulo utiles
def vocales(cadena):
    print(list(filter(lambda l: l.lower() in "aeiou", cadena)))

# modulo uso_utiles
```

```
import utiles

utiles.vocales("Holaaaa!!!!!!")
```

- Primero: ¿qué hace?

8.0.1 ¿Y si queremos invocar el módulo utiles (e invocar a la función vocales) desde la línea de comandos?¿Cómo les pasamos la cadena a analizar?

[ ]:
```
import sys
type(sys.argv)
```

- ¿De qué tipo es argv?
- ¿Qué valores contiene?

```
# modulo utiles
def vocales(cadena):
    print(list(filter(lambda l: l.lower() in "aeiou", cadena)))

if __name__ == "__main__":
    import sys
    vocales(sys.argv[1])
```

9 Paquetes

- Veamos el ejemplo de la documentación oficial de paquetes

```
import sound.effects.echo
from sound.effects import echo
```

9.0.1 ¿Qué contiene el archivo ___init___.py?

10 ¿Qué pasa si tenemos la siguiente sentencia?

```
from sound import *
```

- **___all___: es una variable que contiene una lista con los nombres de los módulos que deberían poder importarse cuando se encuentra la sentencia** from package import ***.

```
#Por ejemplo, en sound/effects/__init__.py

__all__ = ["echo", "surround", "reverse"]
```

- Si **___all___** no está definida, **from sound.effects import *** no importa los submódulos dentro del paquete **sound.effects** al espacio de nombres.
- Un artículo para leer luego.
- **Ejemplo:** analicemos esta librería: console-menu

11 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 juguemos al juego de repaso las pueda acotar por temas?,
- manipular los Python Plus de los estudiantes por turnos?.

¿Qué tienen todas estas situaciones en común?

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

12 Algunas consideraciones antes de empezar

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

13 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.
- **Tarea para el hogar**. Investigar: ¿qué diferencias hay entre un archivo de texto y uno binario?

14 Veamos este ejemplo

[ ]:
```
archi1 = open('archivo.txt', 'w')
```

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

15 ¿Y este otro ejemplo?

[ ]:
```
archi2 = open('archivo.txt', 'x')
```

- Y en este caso, ¿de qué modo se abre este archivo?
- ¿Cuándo puede dar un error esta sentencia?

16 ¿Y en este caso?

[ ]:
```
archi3 = 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,
```

- **encoding**: sólo para modo texto. Por defecto, la codificación establecida en las configuraciones del sistema
- **errors**: sólo en modo texto. Es una cadena que dice qué hacer ante un error en la codificación/decodificación. ("strict", "ignore", ..)
- **newline**: sólo modo texto. Puede ser: None, '','\n', '\r', y '\r\n'.

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

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

16.1 ¿Qué pasa si el archivo no está en la misma carpeta y tenmos que utilizar la ruta completa ?

[ ]:
```
import os

ruta = os.path.dirname(os.path.realpath("."))
ruta
```

[ ]:
```
ruta_completa = os.path.join(ruta, "ejemplo","clase4", "archivo.txt")
ruta_completa
```

17 ¿Cómo almacenamos datos en un archivo?

- El caso más sencillo: guardanto texto en un archivo.

[ ]:
```
f = open('archivo.txt', 'w')
f.write('Hola, ')
print(f.write('Mundo!'))

f.close()
```

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

18 ¿Cómo leemos los datos guardados?

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

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

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

[ ]:
```python
def leo_caracteres():
    f = open("imagine.txt","r")
    for x in f.read():
        print(x)
    f.close()

def leo_lineas():
    f = open("imagine.txt","r")
    print(f.readlines())
    f.close()

def otra_forma():
    f = open("imagine.txt","r")
    for linea in f:
        print(linea)
    f.close()

def main():
    print('Leo caracteres')
    leo_caracteres()
    print('-' * 20)
    print('Leo lineas')
    leo_lineas()
    print('-' * 20)
    print('Otra forma')
    otra_forma()


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

19 ¿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_alumno, cantidad_puntos, etc.
- En estos casos también podría usar un archivo de texto: ¿cómo se les ocurre?

20 Algunas posibilidades

```
'equipo: Astralis - e-sport: CSGO - pais: Dinamarca'

---
equipo:Astralis
e-sport:CSGO
pais:Dinamarca
---

'Astralis-CSGO-Dinamarca'

'Astralis*CSGO*Dinamarca*'
```

- ¿Pros y contras?

21 Hay otras formas mejores...

22 JSON (JavaScript Object Notation)

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

```
{"equipo": "Astralis",
 "e-sport": "CSGO",
 "pais": "Dinamarca"}
o
[{"equipo": "Astralis",
  "e-sport": "CSGO",
  "pais": "Dinamarca"},
  {"equipo": "9z",
  "e-sport": "CSGO",
  "pais": "Argentina"}]
```

- +Info
- Veamos este ejemplo: https://developers.mercadolibre.com.ar/es_ar/categorias-y-publicaciones#close

23 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()**.
  - desserializamos con: **loads()** y **load()**.

- Más info en: https://docs.python.org/3/library/json.html

23.0.1 Veamos este ejemplo

- Generamos un archivo con bandas de distintas ciudades:
  - Tenemos: nombre de la banda, ciudad en la que se generó y una referencia a su trabajo.
  - Empecemos por La Plata...

[ ]:
```
import json

archivo = open("bandas.txt", "w")
datos = [
    {"nombre": "William Campbell", "ciudad": "La Plata", "ref": "www.instagram.
 ↪com/williamcampbellok"},
    {"nombre": "Buendia", "ciudad": "La Plata", "ref":"https://buendia.bandcamp.
 ↪com/"},
    {"nombre": "Lúmine", "ciudad": "La Plata", "ref": "https://www.instagram.
 ↪com/luminelp/"}]
json.dump(datos, archivo)
archivo.close()
```

- ¿De qué tipo es la variable datos?

[ ]:
```
# Ahora accedemos a los datos guardados
import json

archivo = open("bandas.txt", "r")
datos = json.load(archivo)
print(datos)
#datos_a_mostrar = json.dumps(datos, indent=4)
#print(datos_a_mostrar)
archivo.close()
```

- ¿De qué tipo de datos? ¿Y datos_a_mostrar?

24 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,ciudad,ref
William Campbell,La Plata,www.instagram.com/williamcampbellok
Buendia,La Plata,https://buendia.bandcamp.com/
Lúmine,La Plata,https://www.instagram.com/luminelp/
```

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

25 Datasets

- Hay muchos datasets disponibles de muchas temáticas:
- En nuestro país:
  - Datos de Argentina
  - Datos de CABA
  - Datos de La Plata
- Otros:
  - https://data.world/
  - https://www.kaggle.com/
  - https://www.imdb.com/interfaces/
  - Y muchos más...

Muchos son datos abiertos, pero otros... no tanto...

¡PRESTAR ATENCIÓN a la licencias y requisitos para su uso!

26 ¿Qué vemos en netflix?

Vamos a trabajar con el archivo: netflix_titles.csv

[ ]:
```
import csv

ruta = os.path.dirname(os.path.realpath("."))
ruta_archivo = os.path.join(ruta, "teorias", "ejemplos","clase4",␣
 ↪"netflix_titles.csv")
ruta_archivo

archivo = open(ruta_archivo, "r")
csvreader = csv.reader(archivo, delimiter=',')

#encabezado = csvreader.__next__()
encabezado = next(csvreader)
print(encabezado)

archivo.close()
```

27 El módulo csv

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

)
```
csvreader = csv.reader(archivo, delimiter=',')
```

28 Leemos el contenido completo

[ ]:
```
archivo = open(ruta_archivo, "r")
csvreader = csv.reader(archivo, delimiter=',')

#encabezado = csvreader.__next__()
encabezado = next(csvreader)
print(encabezado)

for linea in csvreader:
    if linea[1] == "TV Show" and linea[5] == "Argentina":
        print(f"{linea[2]:<40} {linea[3]}")

archivo.close()
```

¿De qué tipo es línea?

29 Otra solución ...

[ ]:
```
archivo = open(ruta_archivo, "r")
csvreader = csv.reader(archivo, delimiter=',')

shows_ar = filter(lambda x:  x[5] == "Argentina" and x[1] == "TV Show",␣
 ↪csvreader)
for elem in shows_ar:
    print(f"{elem[2]:<40} {elem[3]}")

print(shows_ar)
archivo.close()
```

30 Creamos nuestro archivo csv de bandas de 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.

[ ]:
```
import csv
import json

archivo = open("bandas.txt")
archivo_csv = open("bandas.csv", "w")

bandas = json.load(archivo)

writer = csv.writer(archivo_csv)
writer.writerow(["Nombre", "Ciudad de procedencia", "Refencias"])
for banda in bandas:
```

```
    writer.writerow([banda["nombre"], banda["ciudad"], banda["ref"]])

archivo.close()
archivo_csv.close()
#type(writer)
```

31 Lo leemos

[ ]:
```
archivo_cvs = open("bandas.csv", "r")
csvreader = csv.reader(archivo_cvs, delimiter=',')

for linea in csvreader:
    print(linea)

archivo_csv.close()
```

32 Otra forma de acceder: csv.DictReader

[ ]:
```
archivo_cvs = open("bandas.csv", "r")
csvreader = csv.DictReader(archivo_cvs, delimiter=',')

for linea in csvreader:
    print(linea["Nombre"])

archivo_csv.close()
```

33 Desafío 1

- Dado el conjunto de datos con series y películas de Netflix, queremos:

  1- guardar en otro archivo las peliculas agregadas en el año 2021.

  2- los cinco (5) países con más producciones en Netflix.

34 Desafío 2

- Implementar un programa que muestre un menú a través del cual se puedan visualizar los resultados del desafío 1.
  - Pueden usar la librería console-menu analizada en clase.
  - Pueden agregar más opciones con los ejemplos mostrados en la clase

34.1 Los que quieran, compartir con @clauBanchoff

- Recuerden mandarme mensaje para revisar.