

Ayudantía 6 - Archivos

November 19, 2022

Profesor: Cristian Sepúlveda - cristian.sepulvedas@usach.cl

Ayudante: John Serrano - john.serrano@usach.cl

1 ¿Que son los Archivos de Texto?

Son archivos cuya extensión es **.txt** y contienen una cantidad específica de texto. Si bien hemos utilizado `input()` para las entradas, a veces queremos pasar información por otro medio distinto a la consola, es ahí donde entran a jugar un papel clave los Archivos de Texto. Lo mismo pasa para la salida, si queremos mostrar resultados por otro medio que no sea la consola, podemos dar un resultado en un Archivo de Texto. Un Archivo de Texto es de tipo de dato **FILE**.



ejemplo1.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Hola! Soy un archivo de texto.
Debes procesarme mediante Python.

Es importante siempre guardar los archivos **.py** y **.txt** con los que trabajemos en la misma carpeta, para así evitar problemas de directorios.

2 ¿Como puedo abrir un archivo .txt en Python?

Para abrir un archivo en Python, se debe utilizar la función nativa **open()**, la cual recibe dos entradas: * El nombre del archivo en formato String, incluye su extensión (Por ejemplo, "archivo1.txt") * Un modo de procesamiento del archivo, también en formato String.

Para este curso, se consideran tres modos de procesamiento. Los cuales son: * **r**: Modo de lectura del archivo. Solo se lee el archivo, pero no se modifica de ninguna forma. * **w**: Modo de escritura del archivo. Si ya existe un archivo con el nombre indicado, se borra todo el contenido y se escribe el nuevo contenido, o bien, si no existe el archivo con el nombre indicado, se crea un nuevo archivo con el contenido indicado y el nombre. * **a**: Modo de concatenación. Se mantiene el contenido original del archivo y **directamente al final** del archivo se agrega el nuevo contenido. Independiente de que metodo utilicemos para procesar el archivo, siempre se deben cerrar los archivos de texto. Para ello, se utiliza el metodo **nombre_archivo.close()**

3 El modo lectura

Para leer el contenido de un archivo, se puede utilizar el método **nombre_archivo.readline()**, donde **nombre_archivo** es el nombre del Archivo de Texto. Este método lo que hace es leer una línea del Archivo de Texto y cada vez que se lee una línea, el cursor del archivo se cambia a la siguiente línea.

```
[3]: archivo = open("ejemplo1.txt", "r") # Se lee el archivo de texto "ejemplo1.txt"
linea = archivo.readline()             # Se lee la primera línea. El cursor se
    ↪mueve a la siguiente línea.
print("Línea 1:", linea)                # Se imprime la primera línea del archivo.

linea = archivo.readline()             # Se lee la segunda línea. El cursor se
    ↪mueve al final del archivo (No hay más líneas)
print("Línea 2:", linea)                # Se imprime la segunda línea del archivo.

archivo.close()                        # Se cierra el archivo de texto.
```

Línea 1: Hola! Soy un archivo de texto.

Línea 2: Debes procesarme mediante Python.

Línea 3:

Línea 4:

Utilizando un ciclo While, podemos guardar todo el contenido del archivo de una variable.

```
[4]: archivo = open("ejemplo1.txt", "r") # Se lee el archivo de texto "ejemplo1.txt"
texto_archivo = ""
linea = archivo.readline()
while linea != "":
    texto_archivo += linea
    linea = archivo.readline()
archivo.close()
print(texto_archivo)
```

Hola! Soy un archivo de texto.

Debes procesarme mediante Python.

Pero lo anterior se puede hacer de una manera mucho más fácil y simple. Podemos utilizar el método **nombre_archivo.readlines()** para así leer inmediatamente todo el contenido del archivo. Pero debemos tener en consideración que **readline()** entrega un **String**, en cambio, **readlines()** entrega una **lista de Strings**. Sin embargo, ambos métodos incluyen en el contenido el salto de línea, representado por el string "

```
[3]: archivo = open("ejemplo2.txt", "r") # Se lee el archivo de texto "ejemplo1.txt"
lineas = archivo.readlines()
archivo.close()
```

```
print(lineas)
```

```
['Hoy es Noviembre\n', 'El mes antes de Navidad\n', 'Cuyo numero es 10+1.']
```

4 Modo de escritura

Para escribir en un Archivo de Texto, se debe utilizar el metodo **nombre_archivo.write(lineas)**, donde **lineas** es el contenido del archivo a escribir, en formato String. **Solo 1 String**.

```
[9]: archivo = open("nuevo.txt", "w")
      archivo.write("He creado un Arcgggghivo\nY es de texto.")
      archivo.close()
      print(linea)
```

```
['He creado un Arcgggghivo\n', 'Y es de texto.']
```

Recordemos que si el archivo no existe, entonces se creará. Si ya existe, se borrará todo el contenido y luego se agregará el contenido indicado.

5 Modo de concatenación

Similar al modo de escritura, nuevamente podemos utilizar el metodo **nombre_archivo.write(lineas)**. Aplican las mismas reglas, pero debemos tener en consideración que el nuevo contenido se agregará **inmediatamente después del final del archivo original**.

```
[11]: archivo = open("nuevo.txt", "a")
      archivo.write("\nHe creado un nuevo Archivo de Texto")
      archivo.close()
```

6 Ciclo For in

El ciclo For in es una nueva forma de iterar bastante util cuando se trabaja con archivos. Sigue la siguiente estructura: * For **variable** in **elemento_iterable** * **variable** puede ser cualquier nombre. Usualmente y por buenas practicas, el nombre debe hacer alusión al contenido del elemento iterable. * **elemento_iterable** es cualquier elemento que se puede recorrer, como por ejemplo, listas.

```
[13]: lista = [3,4,1,2]

      for numero in lista:
          numero = numero + 100
          print(numero)
```

```
103
104
101
102
```

For in también es una herramienta muy potente cuando estamos trabajando con **listas bidimensionales**. Supongamos que tenemos una matriz 3x3 y queremos imprimir todos sus elementos.

```
[7]: matriz = [[9,5,2], [3,1,2], [3,4,5]]
      for fila in matriz:
          for columna in fila:
              print(columna)
```

```
9
5
2
3
1
2
3
4
5
```

7 EJERCICIOS

Pyuk, el Dios de la reprobación es dueño de la PyNote, un archivo de texto capaz de reprobar alumnos con solo escribir su nombre en él. Las primeras cuatro líneas del archivo son encabezados y luego cada línea tiene el nombre, la nota final y la causa de reprobación, en algunos casos, solo estará escrito el nombre.

```
PyNote.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
PyNote
=====
Aquel que su nombre sea escrito, reprobara.
=====
Hernesto Yagami#19:Se quedara dormido para la POR.
Juan Manzana#10:Copiara en el LAB.
Fabiola Misa#39:Entregara el archivo .py de otro alumno.
Aquiles Baeza
Lorenzo Lawliet#24:Usara diccionarios en la PEP2.
Nate Rivera#20:Tendra problemas estomacales.
Naruto Izimiki#0:Se lo merece.
Mario Verde#39:No escribira su nombre en la PEP2.
Juan Nieves#34:Lo acuchillaran camino a la universidad.
Wilfredo Watari#23:Pisara el escudo del patio de los perros y GGWP.
```

1. Dado que Pyuk reprueba alumnos por monton, suele olvidar los alumnos que ha reprobado. Desarrolle la función **reprobado(nombre,archivo)** que retorna la causa de reprobación. Si no hay causa de reprobación debe retornar el string **"Perdio las ganas de aprobar"**. Por ultimo, si el nombre no aparece en la lista, retornar **False**.
2. Desarrolle la función **agregar(lista, archivo)**, que reciba una lista con el nombre, nota (int) y la causa de reprobación o solo el nombre de un nuevo alumno. La función debe agregar una línea con la información de la lista al final de la PyNote.
3. Las reglas de la PyNote dicen que si un nombre es tachado antes de reprobar, este aprobará todos sus ramos. Escriba la función **tachar(nombre, archivo)** que modifique el nombre de la persona en el archivo, tachando todas sus letras con un guion (-). **Considere que el nombre siempre estara en el archivo.**

[]: