

</ Ayudantía 10: Archivos

/>



Rolando Rojas

</ Temas a tratar

{01}

Repaso: procesamiento de texto

{02}

Introducción a Archivos

{03}

Lectura de un archivo

{04}

Archivos CSV

{05}

Escritura de un archivo

{06}

Ejercicios
Preguntas
Tarea

</ Procesamiento de texto: caracteres especiales

- Dentro del formato de archivos, se encuentran caracteres especiales que se encargan de dar forma al archivo. Estos se encuentran ocultos para el usuario.
- Python los representa como un carácter especial, compuesto de un backslash y una letra. Por ahora veremos solo `\n` y `\t`.
- El backslash suele escribirse presionando AltGr + ' (tecla de apóstrofe), aunque depende de la distribución del teclado.

`\n`: define saltos de línea.

`\t`: define tabulador (tab).

</ Carácteres especiales: ejemplo

```
impar = 0
If impar % 2 == 0:
    print("es impar")
else:
    print("no es impar")
```

impar.txt (visto desde el usuario)

```
impar = 0\n
If impar % 2 == 0:\n
    \tprint("es impar")\n
else:\n
    \tprint("no es impar")\n
```

impar.txt (interpretado por python)

- El ejemplo ilustra cómo se representa cada línea de un texto en base a lo visto desde el usuario a través de un procesador de texto plano, y cuando este archivo es leído por Python.

</ Operadores

Los siguientes operadores sirven para manipular texto. Sea una variable de texto (str) llamado **texto**.

- **texto.strip()** elimina secuencias de escape (`\n`, `\t`) y espacios en ambos extremos de **texto**.
- **texto.split(separador)** retorna una lista de strings donde cada elemento es una subcadena de texto que termina un carácter antes del **separador**.
El **separador** es opcional, y por defecto es un espacio en blanco " ".
- **texto.replace(texto1, texto2, n)** retorna un string que reemplaza string **texto1** dentro de **texto** por **texto2** dentro de las primeras **n** ocurrencias.
n es opcional, y por defecto es -1, lo que implica que se reemplazan todas las ocurrencias de **texto1** dentro de **texto**.

</ Operadores: ejemplo

```
>>> print("\n\nHello\nWorld\n\n")  
  
Hello  
World  
  
>>> print("\n\nHello\nWorld\n\n".strip())  
Hello  
World  
>>> |
```

Ejemplo strip

```
>>> print("Nada puede salir sal".replace("sal", "mal"))  
Nada puede malir mal  
>>> print("ganso ganso ganso".replace("ganso", "pato", 2))  
pato pato ganso  
>>> |
```

```
>>> print("Lorem ipsum dolor sit amet".split())  
['Lorem', 'ipsum', 'dolor', 'sit', 'amet']  
>>> print("Juan;20;Valparaíso;Casa Central".split(";"))  
['Juan', '20', 'Valparaíso', 'Casa Central']  
...
```

Ejemplo split

</ Operadores

Una variable de clase list, llamado **lista**.

texto.join(lista) retorna un texto donde **lista** es unida en cada uno de sus elementos separados por **texto**.

Puede considerarse el equivalente inverso de **split()**.

```
>>> print("back to csv")
back to csv
>>> print(";".join(['Juan', '20', 'Valparaíso', 'Casa Central']))
Juan;20;Valparaíso;Casa Central
>>> |
```

</ String format

format puede ayudarnos a dar formato a un texto genérico. Para dar formato a un texto se debe tener un string que contiene paréntesis de llaves.

`"{} {} {}".format(texto1, texto2, texto3)` retorna un string donde a cada llave se le reemplaza en orden **texto1**, **texto2**, y **texto3** de manera automática.

`"{2} {0} {1}".format(texto1, texto2, texto3)` reemplaza las llaves con los textos entregados en el orden numérico que se definieron dichas llaves (empezando desde 0).

Nota: no se pueden combinar ambas técnicas, y no pueden saltarse números.

```
>>> print("{} {} {}".format("manzana", "plátano", "naranja"))
manzana plátano naranja

>>> print("{2} {0} {1}".format("manzana", "plátano", "naranja"))
naranja manzana plátano

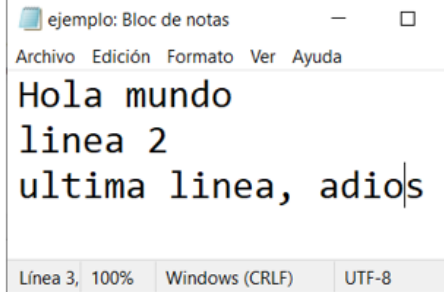
>>> print("muy buenos días señor {}".format("john"))
muy buenos días señor john
```


</ Introducción a Archivos

- En algún punto donde tengamos que resolver algún problema por medio de la programación, será necesario leer archivos con una gran cantidad de datos para procesarlos y convertirlos en información relevante para nosotros.
- En Python, una forma simple es la siguiente:
`variable_archivo = open(directorio)` almacena en una variable el archivo para ser leído.
 - **directorio** es la dirección relativa donde se encuentra el archivo. Esto quiere decir que, si está en la misma carpeta que el programa, basta con escribir el nombre del archivo más su extensión.
- Para leer el contenido del archivo se puede aplicar con un ciclo for, el cual lee cada línea del archivo.
`for linea in variable_archivo:`
- Finalmente, luego de leer la variable es necesario cerrar el archivo, ya que no puede volver a iterarse
`variable_archivo.close()`

</ Introducción a Archivos: lectura

```
archivo = open("ejemplo.txt")
for linea in archivo:
    linea = linea.strip()
    print(linea)
archivo.close()
```



ejemplo: Bloc de notas

Archivo Edición Formato Ver Ayuda

Hola mundo
línea 2
ultima linea, adios

Línea 3, 100% Windows (CRLF) UTF-8

```
===== RE:
Hola mundo
línea 2
ultima linea, adios
>>>
```

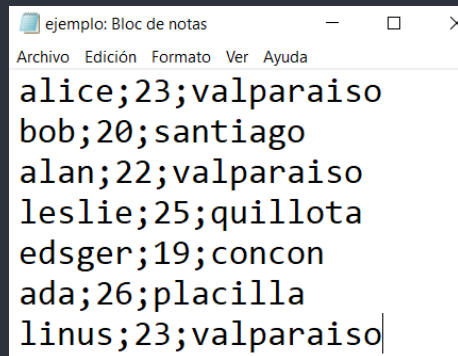
Conjunto de imágenes que muestran el script de código para leer archivo ejemplo.txt (izquierda), el contenido de ejemplo.txt (centro), y resultado de ejecución (derecha).

</ Ejemplo de aplicación: CSV

- Un archivo CSV (archivo separado por comas), es un archivo de formato estructurado que separa datos en algún separador, sea este una coma, punto y coma, o dos puntos.
- Se puede interpretar como filas (líneas) y columnas (separadores)
- Por esta razón es importante el uso de `split()`.

</ Lectura de un archivo CSV

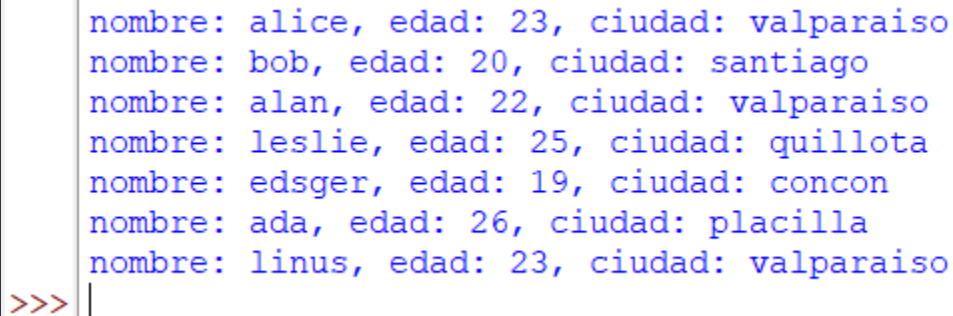
```
archivo = open("ejemplo.txt")
for linea in archivo:
    linea = linea.strip().split(";")
    print("nombre: {}, edad: {}, ciudad: {}".format(linea[0], linea[1], linea[2]))
archivo.close()
```



ejemplo: Bloc de notas

Archivo Edición Formato Ver Ayuda

alice;23;valparaiso
bob;20;santiago
alan;22;valparaiso
leslie;25;quillota
edsgen;19;concon
ada;26;placilla
linus;23;valparaiso



```
>>> nombre: alice, edad: 23, ciudad: valparaiso
nombre: bob, edad: 20, ciudad: santiago
nombre: alan, edad: 22, ciudad: valparaiso
nombre: leslie, edad: 25, ciudad: quillota
nombre: edsgen, edad: 19, ciudad: concon
nombre: ada, edad: 26, ciudad: placilla
nombre: linus, edad: 23, ciudad: valparaiso
>>> |
```

Algoritmo para lectura de CSV, por cada línea se separan los datos y se entregan en un formato legible en pantalla.

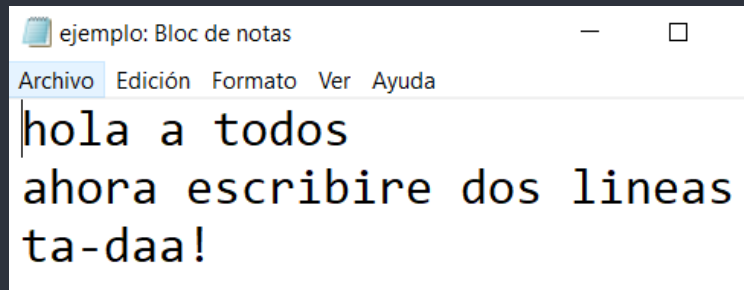
</ Escritura en un archivo – 'w'

- Para escribir un archivo el procedimiento para abrir el archivo es similar, con una sutil diferencia.
- **variable_archivo = open(nombre, "w")** almacena en variable_archivo la apertura de un archivo llamado nombre, el segundo parámetro "w" indica que es para escribir (write).
 - Si archivo no existe, se crea.
 - Si archivo existe, se **sobrescribe**, borrando todo contenido que tuvo previamente (!).
- **variable_archivo.write(texto)** escribe texto sobre el archivo.
 - Se puede usar más de una vez, por ejemplo, dentro de un ciclo.
 - No olvide añadir un \n si necesita delimitar un salto de línea.
- Para guardar cambios, se requiere cerrar el archivo escribiendo **variable_archivo.close()**
 - De no hacerlo toda escritura no será guardada.

</ Escritura en un archivo – 'w' - Ejemplo

```
archivo = open("ejemplo.txt", "w")
archivo.write("hola a todos\n")
archivo.write("ahora escribire dos lineas\nta-daa!")
archivo.close()
```

```
>>> ===== RESTART: C:\Users\Ignacio\Desktop\example.py
```



Ejemplo de ejecución de script de Python escribiendo un archivo.

</ Patrón de lectura/escritura [consideraciones]

- Cuando buscamos alterar un archivo en específico, aplicar 'w' sobre el mismo no es recomendable ya que se sobrescribe.
- La alternativa a esto sería leer el archivo, y por cada línea leída escribirlo a un archivo temporal "archivo"+"_temp". De esta forma se guardan los cambios en un archivo nuevo.
- Luego, se lee el archivo temporal y se sobrescribe el archivo original.

</ Ejercicio 1

Se averió la base de datos de ventas de la empresa. Todo lo que queda es un respaldo corrompido donde los campos de venta más iva se vieron afectados, y la declaración de renta se acerca. Dado que el jefe de la empresa no repara en gastos, le encargó a estudiantes de programación el desarrollar un algoritmo que sea capaz de “reparar” dicho respaldo calculando los valores faltantes.

El archivo de respaldo es un CSV que tiene el siguiente formato:

ID_venta;fecha;subtotal;total

Donde o total o subtotal puede que se hayan corrompido.

Por suerte, hay archivos del detalle de las ventas para cuando ambos fallen. El nombre del archivo es el mismo de la ID de venta en formato .txt.

Se identificará con una x el campo del archivo que se encuentre corrupto.

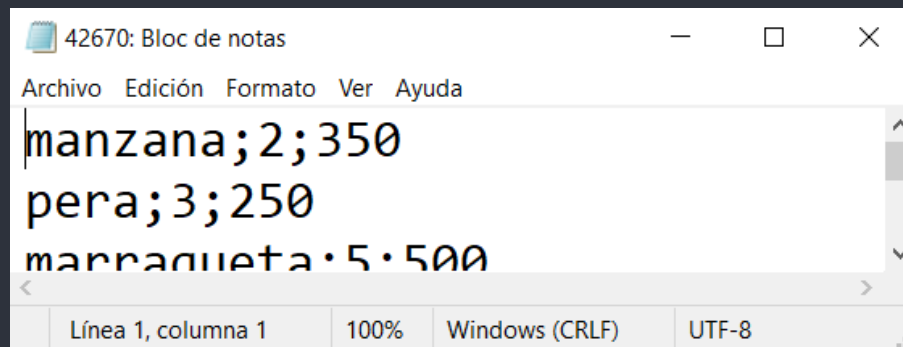
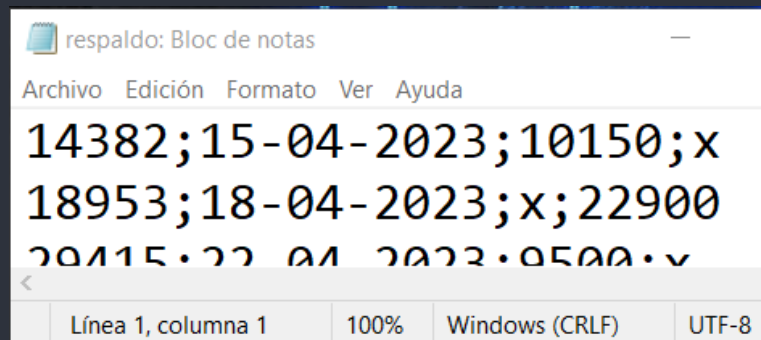
Cada archivo de detalle presenta el siguiente formato:

Item;precio;cantidad

Donde precio es el precio individual de cada item.

- Considere un IVA del 19%, redondear valor al entero.
- Puede guardar sus resultados en un archivo nuevo.
- Evite hardcodear nombres o valores en su código salvo por el factor IVA y los nombres de archivo que se mantendrán fijos.

</ Ejercicio 1: ejemplo de respaldo y detalle



</ Ejercicio 1: tips

- Necesito leer archivo de respaldo.
- Para tener cambios se debe crear un archivo de salida que almacene los datos ya procesados.
- Por cada línea identificar subtotal y total.
- Si al menos uno de ellos no está corrupto, se puede calcular con el IVA.
- Si ambos están corruptos, se necesita buscar el archivo de detalles.
- Por cada línea del archivo de detalles, calcular el subtotal de la venta con un acumulador.
- Guardar resultado calculado de cada línea en un acumulador (ejemplo: lista).
- Recorrer acumulador y guardar información en archivo de salida.

</ Ejercicio 1: Solución

```
#arreglo de strings para escribir en archivo
salida = []
respaldo = open("respaldo.txt")
for linea in respaldo:
    linea_s = linea.strip().split(";")
    if linea_s[2] != "x":
        # si subtotal no esta corrupto, se calcula total multiplicado por iva
        linea_s[3] = round(int(linea_s[2])*1.19)
        salida.append("{};{};{};{}\n".format(linea_s[0],linea_s[1],linea_s[2],str(linea_s[3])))
    elif linea_s[3] != "x":
        # si total no esta corrupto, se calcula subtotal dividiendo por iva
        linea_s[2] = round(int(linea_s[3])/1.19)
        salida.append("{};{};{};{}\n".format(linea_s[0],linea_s[1],str(linea_s[2]),linea_s[3]))
    else:
        #si ambos estan corruptos, se requiere leer archivo detalle
        detalle = open(linea_s[0]+".txt")
        acumulador = 0
        for linea_detalle in detalle:
            split_2 = linea_detalle.strip().split(";")
            #nos interesa cantidad por detalle
            acumulador += int(split_2[1]) * int(split_2[2])
        detalle.close()
        subtotal = acumulador
        total = round(subtotal * 1.19)
        salida.append("{};{};{};{}\n".format(linea_s[0],linea_s[1],str(subtotal),str(total)))
respaldo.close()

respaldo_arreglado = open("respaldo_arreglado.txt",'w')
for transaccion in salida:
    respaldo_arreglado.write(transaccion)
respaldo_arreglado.close()
```

Tarea



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



Ayudantía 10: Archivos



¿Dudas?

rolando.rojass@usm.cl



Rolando Rojas