Archivos

IITA 2023

Primero...un repaso:

- Python...¿Qué es?
- SOFTWARE
- HARDWARE
- LENGUAJE DE PROGRAMACIÓN
- PROGRAMA
- TIPO DE DATO
- VARIABLE

- Estructuras:
  - Selectivas
    - Simples
    - Múltiples
  - Repetitivas

- Funciones:
  - Ámbito de las variables

- Listas
  - Operaciones
- Tuplas
- Diccionarios

- Módulos y librerías
  - LE de Python
  - PIP

Archivos

- Un archivo es una la unidad que utilizamos para guardar información de manera no volátil, es decir, para guardar la información que manejamos incluso después de que apagamos la máquina.
- Hasta ahora, todas las variables y datos que fuimos manejando se perdían una vez que el programa terminaba su ejecución.

  → Usando archivos, podemos grabar esta información en un medio de almacenamiento (como el tan conocido disco duro de hoy en día) y **leerla/escribirla**/ejecutarla llamando a dichos archivos

Archivos: Tipos

- Podemos dividir los archivos en dos grandes tipos:
  - **BINARIOS:** contienen información que no es traducible a texto, incluso si parte de su información si se pueda traducir
    - .exe
    - .dll
    - .jpg
    - .mp4
  - **ASCII o de texto:** toda la información que contienen puede ser traducida a texto usando un esquema de codificación como ASCII. Ejemplos:
    - .py
    - .txt
    - .csv
    - .xml

Algunos ejemplos más:

Common extensions that are binary file formats:

- Images: jpg, png, gif, bmp, tiff, psd, ...
- Videos: mp4, mkv, avi, mov, mpg, vob, ...
- Audio: mp3, aac, wav, flac, ogg, mka, wma, ...
- Documents: pdf, doc, xls, ppt, docx, odt, ...
- Archive: zip, rar, 7z, tar, iso, ...
- Database: mdb, accde, frm, sqlite, ...
- Executable: exe, dll, so, class, ...

Common extensions that are text file formats:

- Web standards: html, xml, css, svg, json, ...
- Source code: c, cpp, h, cs, js, py, java, rb, pl, php, sh, ...
- Documents: txt, tex, markdown, asciidoc, rtf, ps, ...
- Configuration: ini, cfg, rc, reg, ...
- Tabular data: csv, tsv, ...

Archivos: ¿Para qué sirven?

- Si bien ya mencionamos ciertos usos, podemos destacar también los siguientes:

→ Pipeline: comunicar procesos

→ Modelar (procesos, recursos, conexiones, funciones , etc)

→ Persistencia en los datos

→ Permiten almacenar lo que conocemos como programas para que luego se transformen en procesos

Archivos: Python

- **Abrir un archivo**

file=open(nombre_archivo,modo)

→La función open() devuelve un *objeto archivo*.

→El parámetro “modo”, puede ser:

- “r”: Read, lo abre como sólo lectura
- “r+”: Lo abre como lectura/escritura
- “a”: Append, agrega información
- “w”: Write, escribe un archivo (crea uno nuevo si es que no existe o ‘sobreescribe’ uno con el mismo nombre)

Archivos: Python

- **Operaciones sobre archivos**

→Usando file, podemos:

→*file.**read**(cantidad)*

→*Cantidad es un argumento numérico, cuando se omite este argumento, o es negativa, el contenido del archivo será leído y devuelto en su totalidad*

→*file.**readline**()*

→ *Lee una sola línea del archivo*

→*file.**readlines**()*

→*Lee todas las líneas de un archivo*

→*También se puede usar list(file)*

Archivos: Python

- **Operaciones sobre archivos**

→Usando file, podemos:

→*file.**write**(“Esto es una prueba\n”)*

→*Write escribe el contenido dado al archivo y nos devuelve la cantidad de caracteres escritos.*

→*Para ejecutarla, debemos abrir el archivo en el modo adecuado (‘w’, ‘r+’, ‘a’, etc.)*

→*file.**tell**()*

→*Devuelve un entero que indica la posición actual en el archivo*

→*file.**close**()*

→Cierra y libera cualquier recurso del sistema tomado por el archivo abierto

Archivos: Python

- Es importante cerrar un archivo una vez finalizado nuestro uso del mismo. Una buena práctica es usar la declaración **with** cuando manejamos objetos archivo. Tiene la ventaja que el archivo es cerrado apropiadamente luego de que el bloque termina, incluso si se produjo un error.

```
>>> with open('archivodetrabajo', 'r') as f:
...     read_data = f.read()
>>> f.closed
True
```

- Si no estamos llamando al archivo con with , debemos llamar a f.close() para cerrar el archivo y liberar los recursos de la máquina que está ocupando. Si no hacemos esto, en aplicaciones grandes donde los archivos abiertos pueden ser muchos o de gran tamaño, podemos llegar a ocupar toda la memoria de la maquina con archivos que ya no necesitamos.

A practicar...

- 1- Escribir un programa que abra un archivo, lea todas sus líneas y cuente cuantas líneas existen en el mismo

- 2- Utilizar Python para escribir un archivo de texto que tenga 11 líneas, en cada una escribir lo que deseen y cerrar el archivo. Luego mostrar el contenido del archivo.

- 3- [Escribir una función] que cuente cuantos caracteres existen dentro del archivo creado en el punto anterior

A practicar...

- 4- Escriba un programa que pida al usuario su nombre. Cuando este lo ingrese, muestre un mensaje de bienvenida en la pantalla, y agregue una línea donde registre la visita del usuario a un archivo llamado libro_invitados.txt. Asegúrese de que cada registro figure en una nueva línea, y de que cada nueva entrada sea grabada en el mismo archivo, incluso entre múltiples ejecuciones del programa

  (No hay problema con los repetidos)

5- Escribir un programa que copie los contenidos de un archivo A y los vuelque en un archivo B