python™

Archivos

CTIC - UNI

Universidad Nacional de Ingeniería

Introducción

- Los SO almacenan los datos de forma estructurada en unas unidades básicas de almacenamiento a las que llamamos archivos. Los programadores necesitan trabajar con archivos ya sea para leer datos de los mismos o para crearlos.

- Los lenguajes de programación suelen incluir librerías que contienen funciones para el tratamiento de archivos.

- En Python contamos con una serie de funciones básicas, incluidas en su librería estándar para leer y escribir datos en archivos.

¿Qué es un archivo?

- Un archivo es una colección de datos que se guarda en un dispositivo de almacenamiento secundario, como un disco duro o una unidad de disco USB.
- Acceder a un archivo significa establecer una conexión entre el archivo y el programa y mover datos entre los dos.

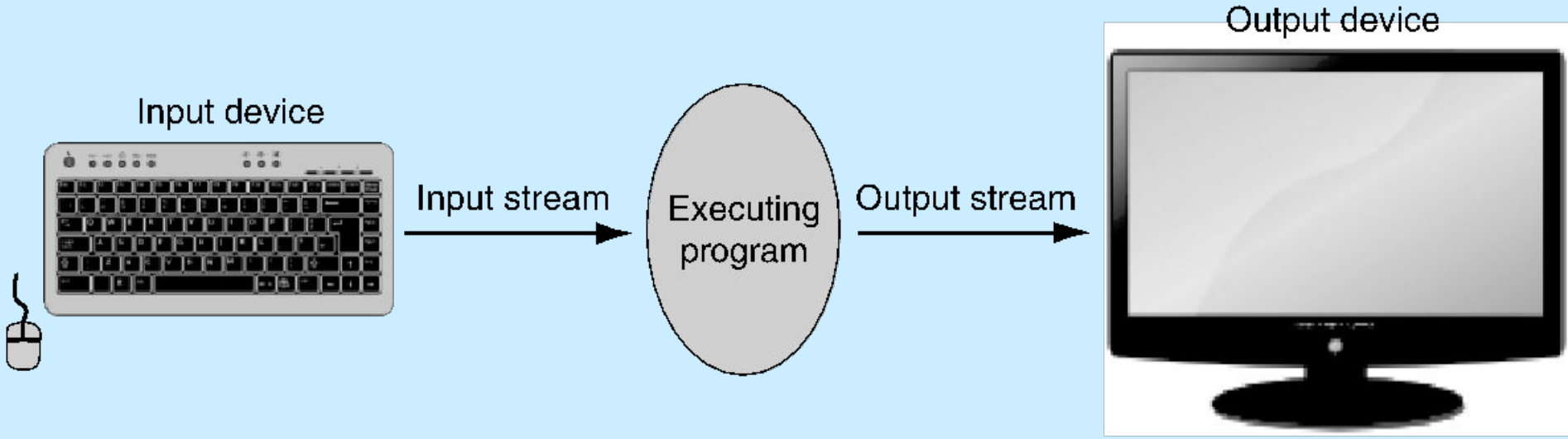
Dos tipos de archivos

Los archivos vienen en dos tipos generales:

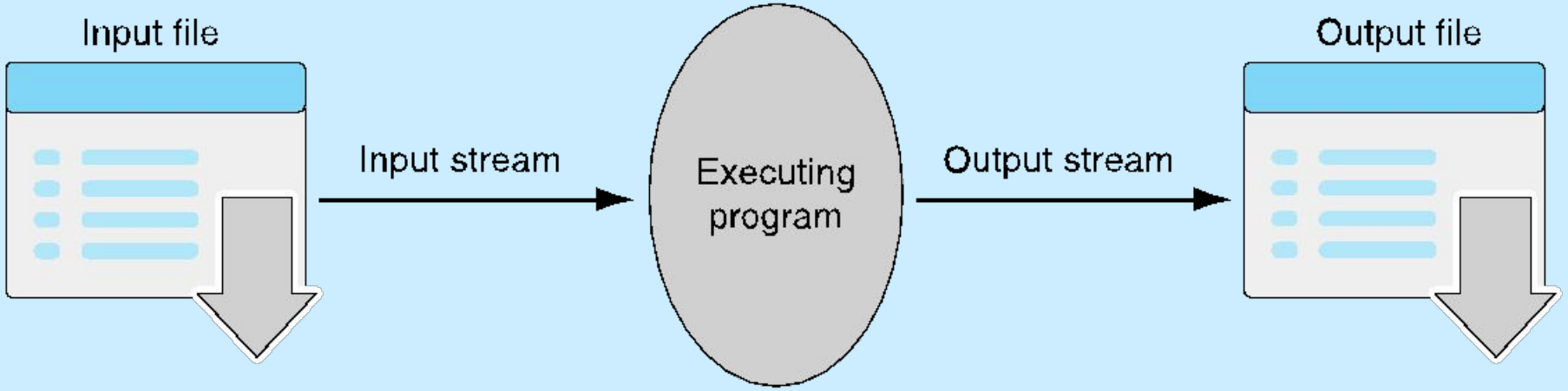
- *Ficheros de texto*. Archivos donde se traducen caracteres de control como "/n". Esto es generalmente legible por humanos
- *Ficheros binarios*. Toda la información se toma directamente sin traducción. No es legible y contiene información no legible.

Objetos archivo o Flujo

- Al abrir un archivo, crea un objeto archivo o un flujo de archivo que es una conexión entre la información del archivo en el disco y el programa.
- El flujo contiene un búfer de la información del archivo y proporciona la información al programa.



a) Standard input and output



b) File input and output

FIGURE 5.1 Input-output streams.

Buffering

- Leer desde un disco es muy lento. Por lo tanto, la computadora leerá una gran cantidad de datos de un archivo con la esperanza de que, si necesita los datos en el futuro, se almacenará en el objeto archivo.
- Esto significa que el objeto archivo contiene una copia de la información del archivo llamada caché.

Operaciones con Archivos

- Python incorpora en su librería estándar una estructura de datos específica para trabajar con archivos.

- Básicamente, esta estructura es un stream que referencia al archivo físico del sistema operativo.

- Entre las operaciones que Python permite realizar con ficheros encontramos las más básicas que son: apertura, creación, lectura y escritura de datos.

Archivos - Apertura

- La principal función que debemos conocer cuando trabajamos con ficheros se llama **open()** y se encuentra integrada en la librería estándar del lenguaje.

- **mi_fich = open(nombreFich,modo)**
- Abre un archivo para un modo, ya sea (r,w,a)

Ejemplo

- mi_fich= open(“nuevo.txt” , “w”)

Ejemplo de apertura para lectura

```
mi_fich = open("mi_archivo.txt", "r")
```

- `mi_fich` es el objeto archivo. Contiene el búfer de información. La función open crea la conexión entre el *archivo de disco* y el *objeto archivo*. La primera cadena entre comillas es el nombre del archivo en el disco, la segunda es el modo para abrirla (aquí, "r" significa leer)

¿Dónde está el archivo de disco?

- Cuando se abre, el nombre del archivo puede venir en una de dos formas:
- `"fichero.txt"` asume que el nombre del archivo es fichero.txt y está ubicado en el directorio actual del programa
- `"c:\misdatos\fichero.txt"` es el nombre de archivo completo e incluye la información del directorio

Diferentes modos

Mode	How Opened	File Exists	File Does Not Exist
'r'	read-only	Opens that file	Error
'w'	write-only	Clears the file contents	Creates and opens a new file
'a'	write-only	File contents left intact and new data appended at file's end	Creates and opens a new file
'r+'	read and write	Reads and overwrites from the file's beginning	Error
'w+'	read and write	Clears the file contents	Creates and opens a new file
'a+'	read and write	File contents left intact and read and write at file's end	Creates and opens a new file

TABLE 5.1 File Modes

Cuidado con los modos de escritura

- Tenga cuidado si abre un archivo con el modo `'w'`. Establece el contenido de un archivo existente para que esté vacío, destruyendo cualquier dato existente.
- El modo `'a'` es más agradable, lo que le permite escribir al final de un archivo existente sin cambiar el contenido existente.

Los archivos de texto usan cadenas

- Si está interactuando con archivos de texto (que es todo lo que haremos en esta sesión), recuerde que todo es una cadena
  - todo lo leído es una cadena
  - si su intención es escribir en un archivo, solo puede escribir una cadena.

Métodos para Leer el contenido de archivos

- Python nos ofrece los métodos para la operación de lectura:
- ObjetoArch**.read()**
- Lee el contenido de todo el archivo y lo almacena como un solo string
- ObjetoArch**.readline()**
- Lee la siguiente línea del archivo.
- ObjetoArch**.readlines()**

  Lee todo el contenido del archivo y lo almacena en una lista de strings.

Obtener el contenido del archivo

- Una vez que tenga un objeto archivo:
- ObjetoArch.read() - lee todo el contenido del archivo como una cadena y lo devuelve. Puede tomar un argumento opcional entero para limitar la lectura a N bytes, es decir ObjetoArch.read(N)
- ObjetoArch.readline() - entrega la siguiente línea como una cadena.

Más lecturas de archivo

- ObjetoArch.readLines() - devuelve una lista única de todas las líneas del archivo.

Ejemplo:

```
for linea in ObjetoArch:
```

- iterador para recorrer las líneas de un archivo.

escritura en un archivo

Una vez que se ha creado un objeto archivo, abierto para escritura, puede usar el comando de impresión

- se agrega `file=ObjetoArchivo` en el comando print

```
#Abre el archivo para escritura:
#si el archivo no existe, lo crea
#si el archivo existe, sobreescribe su contenido

fich_temp=open("temp.txt","w")
print("primera linea",file=fich_temp)
print("segunda linea",file=fich_temp)
fich_temp.close()
```

Archivos - Escritura

- La operación básica de escritura en un archivo se hace en Python a través del método write.

- ObjetoArch**.write(texto)**
- Escribe un texto a un archivo.

- ObjetoArch**.close()**
- Cierra un archivo

close

Cuando el programa termina de realizar las operaciones con un archivo, debemos cerrar el archivo:

- Se debe vaciar el contenido del búfer de la computadora al archivo.
- Se debe abandonar la conexión al archivo.
- **`close`** es un método de un objeto archivo

```
ObjetoArchivo.close()
```

- Todos los archivos deben estar cerrados.!


Ejercicio
Revertir las líneas de un archivo

```
#Programa que lee el contenido del fichero entrada.txt
#linea por linea, en cada iteracion
#revierte cada linea y la copia en el archivo salida.txt

fich_entrada=open("entrada.txt","r")
fich_salida=open("salida.txt","w")

for linea in fich_entrada:
    acum=""
    linea=linea.strip()
    for simb in linea:
        acum=simb + acum
    print(acum, file=fich_salida) #guarda en el archivo de salida

    #Envia los resultados a pantalla para que observemos el progreso
    print("Linea:{:12s} invertida es: {:s}".format(linea,acum))
fich_entrada.close()
fich_salida.close()
```