

MANEJO DE ARCHIVOS ENTRADA/SALIDA

Fundamentos de Programación
FIEC04341

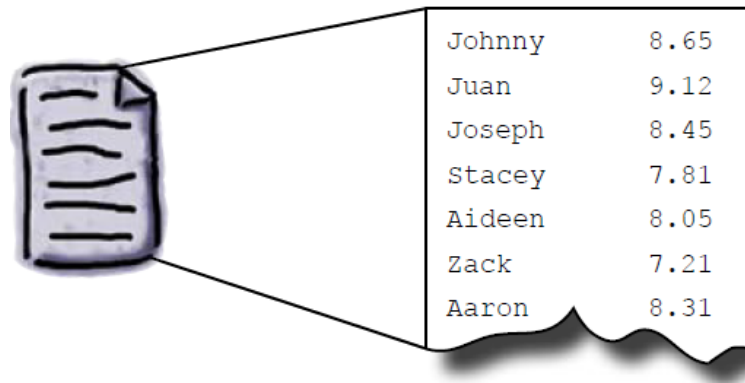
Agenda

- Conceptos básicos de archivos
- Operaciones de entrada utilizando archivos
- Operaciones de salida utilizando archivos

Conceptos Básicos de Archivos.

CONCEPTOS BÁSICOS DE ARCHIVOS

- La información que se guarda en las variables es temporal.
 - Todo lo que se almacena en variables se pierde cuando el programa termina.
 - Esta información es almacenada realmente en memoria principal (RAM).



Johnny	8.65
Juan	9.12
Joseph	8.45
Stacey	7.81
Aideen	8.05
Zack	7.21
Aaron	8.31

CONCEPTOS BÁSICOS DE ARCHIVOS

- Para poder guardar la información de forma permanente se debe almacenar en archivos; así se mantiene aún después de que el programa haya terminado su ejecución.
- Ejemplos:
 - Puntajes en los juegos, como el Buscaminas
 - Notas de estudiantes en una hoja electrónica
 - Voces grabadas usando el computador, etc.

ARCHIVOS DE TEXTO

- Los archivos pueden almacenar todo tipo de información, dependiendo de ello tienen un *tipo* el cual se reconoce en Windows por la *extensión*.
- En Python tenemos los archivos que almacenan información como secuencia de caracteres, es decir, en cadenas (caracteres ASCII).
- Estos son los archivos de texto plano (extensión *.txt*), cuyo acceso se realiza de forma secuencial. Además, tienen la ventaja de ser multi-plataforma.

ABRIR Y CERRAR ARCHIVOS

- Desde un programa en Python se puede crear archivos, o abrir archivos ya existentes para modificarlos. Se puede tomar datos del archivo y ponerlos en variables, para ser manipulados en el programa. Estas operaciones se manejan a través de funciones.
- Para poder trabajar con un archivo, se necesita:
 - Abrirlo con la función **open()**
 - Asignar el resultado de la apertura a una **variable**
 - Efectuar las operaciones de **lectura/escritura**
 - Cuando se ha terminado de trabajar con el archivo, se debe cerrar con la función **close()**

ABRIR Y CERRAR ARCHIVOS

variable = open(nombre_del_archivo, tipo_de_acceso)

archivo = open("mi_archivo.txt", "r")

archivo.close()

- Si en el **nombre_del_archivo** no se incluye la ruta donde está almacenado, Python asume el directorio de trabajo actual.
- Se debe proveer un modo de operación o **tipo_de_acceso**.
- El archivo debe cerrarse con la función **close()** una vez que se ha terminado de trabajar con él.

ABRIR Y CERRAR ARCHIVOS

Tipo de Acceso (Modo)	Descripción
"r"	Lectura. El archivo debe existir, caso contrario se generará un error.
"w"	Escritura. Si el archivo existe, su contenido es sobrescrito. Si no existe, se crea.
"a"	Añadir. Si el archivo existe, se agregan nuevos datos. Si no existe, se crea.
"r+"	Lectura y escritura.
"w+"	Escritura y lectura.
"a+"	Adición y lectura.

Operaciones de entrada y salida utilizando archivos.

OPERACIONES DE ENTRADA/SALIDA

The "each_line" variable is set to the next line from the file on each iteration. The for loop stops when you run out of lines to read.

```
result_f = open("results.txt")
for each_line in result_f:
    print(each_line)
result_f.close()
```

Open the file and give it a file handle.

Do something with the thing you've just read from the file. In this case, you print out the line. Notice that the for loop's code is indented.

Close the file (through the file handle) when you're done with it.

OPERACIONES DE ENTRADA

- read(n)** Lectura de un número específico de caracteres.
- readline()** Lectura de una línea de texto hasta encontrar el carácter de fin de línea '\n'
- readlines()** Lectura de todas las líneas del archivo, y las retorna en forma de lista.

```
print("\nReading one line at a time.")
text_file = open("read_it.txt", "r")
print(text_file.readline())
print(text_file.readline())
print(text_file.readline())
text_file.close()
```

```
print("\nReading the entire file into a list.")
text_file = open("read_it.txt", "r")
lines = text_file.readlines()
print(lines)
print(len(lines))
for line in lines:
    print(line)
text_file.close()
```

```
print("\nLooping through the file, line by line.")
text_file = open("read_it.txt", "r")
for line in text_file:
    print(line)
```

OPERACIONES DE SALIDA

write(str)

Escritura de una cadena. Crea una línea de texto en el archivo, e incluye el carácter de fin de línea ‘\n’

writelines(list)

Escritura de las cadenas de una lista hacia un archivo.

```
text_file.write("Line 1\n")
text_file.write("This is line 2\n")
text_file.write("That makes this line 3\n")
```

```
lines = ["Line 1\n",
        "This is line 2\n",
        "That makes this line 3\n"]
```

```
text_file.writelines(lines)
```

```
print("\nReading the newly created file.")
text_file = open("write_it.txt", "r")
print(text_file.read())
text_file.close()
```

OTRAS FUNCIONES CON ARCHIVOS

Detectar la posición actual del dispositivo de lectura del archivo

```
p=f.tell()
```

p contendrá un entero con la posición actual.

La posición inicial en el archivo es 0

Ubicar el dispositivo de lectura del archivo en una posición especificada

```
f.seek(d)
```

d es el desplazamiento contado a partir del inicio que es la posición 0

BIBLIOGRAFÍA

- Mike Dawson, Python Programming, Third Edition, 2010, CENGAGE Learning.
- Paul Barry, Head First Python, Second edition, 2010, O'Reilly
- Luis Rodríguez, Python Programación