

Programación de Computadores

Archivos

Jonatan Gómez Perdomo, Ph. D.

`jgomezpe@unal.edu.co`

Arles Rodríguez, Ph.D.

`aerodriguezp@unal.edu.co`

Camilo Cubides, Ph.D. (c)

`eccubidesg@unal.edu.co`

Research Group on Artificial Life – Grupo de investigación en vida artificial – (Alife)

Computer and System Department

Engineering School

Universidad Nacional de Colombia

Contenido

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Agenda

- 1 **Introducción**
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Definición

- La **memoria** del computador permite almacenar datos (valores y programas) con los que el computador puede ejecutar una tarea.
- Existen dispositivos externos a la memoria donde se pueden guardar datos que pueden ser usados por el computador. Estos pueden ser discos duros, cintas, memorias SD, USB, entre otras.
- Un **archivo** es una unidad lógica de almacenamiento de datos en dispositivos externos (no memoria principal). Se dice unidad lógica porque todos los bytes almacenados en un archivo se ven como una sola unidad que tiene un inicio y un fin, sin importar lo que esos bytes esten representando.
- El **tamaño** de un archivo es el número de bytes que almacena.



Tipos

Aunque existen muchos tipos de archivos, los programadores los clasifican en dos tipos básicos:

- **Texto:** Cuando cada byte representa un ASCII (en algunos casos UTF-8 o UTF-16) como los archivos `.txt`, los de páginas web `.html`, los de código fuente en muchos lenguajes de programación `.py`, `.c`, `.java`, `.cpp`.
- **Binario:** Cuando cada byte representa algo diferente o requiere un conjunto de bytes para representar algo, como los archivos de imagen `.jpg`, `.png`, de documento `.doc`, `.pdf`, `.xls`, `.ods`, comprimidos `.rar`, `.zip`, entre otros.



Operaciones

Existen tres operaciones que se pueden realizar con un archivo desde un programa:

- **Crear:** Crear un archivo en un dispositivo externo desde el programa.
- **Abrir:** darle permiso al programa de acceder al contenido almacenado por el archivo en diferentes **modos**, ya sea leer (traer datos del archivo al programa), escribir (guardar datos del programa en el archivo), adicionar (agregar datos del programa al final del archivo), y/o moverse.
- **Cerrar:** Remover al programa el permiso de leer, escribir, adicionar, y/o moverse en el contenido del archivo.
- **Eliminar:** Remover desde el programa un archivo de un dispositivo externo.



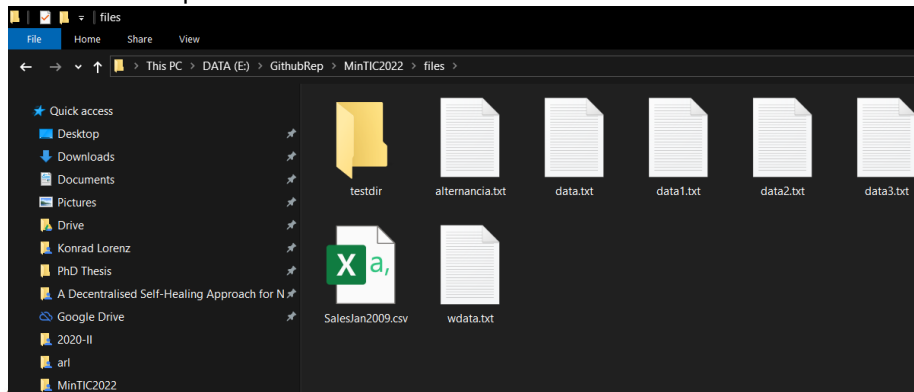
Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Ubicación de archivos en los ejemplos

En el siguiente ejemplo se abrirán archivos ubicados en la carpeta files que fue creada previamente y contiene algunos ejemplos. Así luce la carpeta files en el computador:



Direccionamiento

Es posible trabajar con direccionamiento absoluto o relativo en python. En los ejemplos siguientes se realizará direccionamiento relativo a partir de la carpeta donde se ejecuta el código. En esta carpeta hay una carpeta llamada files y por ejemplo, allí se encuentra el archivo 'data.txt'.

Para dar soporte a los diferentes sistemas operativos como windows y trabajar con direccionamiento absoluto se toma como nombre del archivo la ruta completa del archivo:

```
"C:\ThisFolder\workData.txt"
```

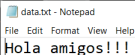
```
"C:/ThisFolder/workData.txt"
```



Abrir y leer un archivo (open y with)

`open(ruta del archivo, 'r')` toma un nombre de archivo y modo como argumentos. `r` abre el archivo en modo de lectura (read).

Adicionalmente y como buena práctica de programación en python se utilizará `with` que permite cerrar el archivo automáticamente y asegura un uso propio del archivo. El siguiente código abre el archivo `data.txt` ubicado en la carpeta `files` y muestra su contenido:



```
data.txt - Notepad
File Edit Format View Help
Hola amigos!!!
```

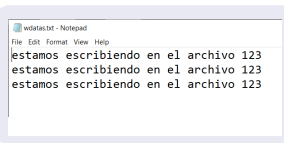
```
with open('files/data.txt', 'r') as f:
    data = f.read()
    print(data)
```



Escribir y crear un archivo (open y with)

`open(ruta del archivo, 'w')` toma un nombre de archivo y modo como argumentos. `w` abre el archivo en modo de escritura creando el archivo si no existe (write) ó sobrescribiendo el archivo si existe. El siguiente código crea el archivo `wdata.txt` en la carpeta `files`:

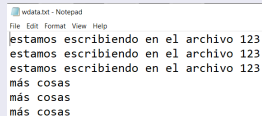
```
with open('files/wdata.txt', 'w') as f:  
    data = 'estamos escribiendo en el archivo 123'  
    f.write(data)  
    f.write(data)  
    f.write(data)
```



Agregar contenido a un archivo existente (open y with)

`open(ruta del archivo, 'a')` toma un nombre de archivo y modo como argumentos. `a` abre el archivo en modo de adjuntar creando el archivo si no existe (append) ó escribiendo al final del archivo si existe. El siguiente código escribe al final del archivo `wdata.txt`:

```
with open('files/wdata.txt', 'a', encoding='utf-8') as f:  
    data = 'más cosas'  
    f.write(data)  
    f.write(data)  
    f.write(data)
```



wdata.txt - Notepad
File Edit Format View Help
estamos escribiendo en el archivo 123
estamos escribiendo en el archivo 123
estamos escribiendo en el archivo 123
más cosas
más cosas
más cosas



Otros modos de apertura de archivos

Otros modos de abrir archivos en python son:

- 'r+' – (read/write): Este modo es usado cuando se desean hacer cambios al archivo y a la vez leer información. El puntero del archivo se ubica al principio del archivo.
- 'a+' – (append and Read): Se abre el archivo para escribir y leer del archivo. El puntero del archivo se ubica al final del archivo.
- - x (exclusive creation): usado exclusivamente para crear un archivo. Si existe un archivo con el mismo nombre la función fallará.
- Para leer archivos binarios se agrega la letra b al final del modificador. Por ejemplo para leer `“wb”` y los otros serían `“rb”`, `“ab”`, `“r+b”` y `“a+b”`.



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos**
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Lectura de un archivo de texto (read)

Para leer el contenido de un archivo se puede usar `read()`. Sin parámetros leerá el archivo entero y almacenará el contenido la salida como un string si es un archivo de texto o como objetos en bytes si se trata de un archivo binario. Si se trata de leer un archivo más grande que la memoria disponible no se podrá leer todo el archivo de una vez.

```
with open('files/data1.txt', 'r') as f:  
    print(f.read())
```

Una posible salida sería:

```
Esta es la lÃnea 1: abcabcabc  
Esta es la lÃnea 2: abcabcabc  
Esta es la lÃnea 3: abcabcabc
```

data1.txt - Notepad

File Edit Format View Help

```
Esta es la línea 1: abcabcabc  
Esta es la línea 2: abcabcabc  
Esta es la línea 3: abcabcabc
```



Lectura de un archivos con tildes (read)

Para no tener problemas de codificación se puede agregar el siguiente parámetro:

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcabc
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc

```
with open('files/data1.txt', 'r', encoding="utf-8") as f:  
    print(f.read())
```

La salida sería como la siguiente:

Esta es la línea 1: abcabcabc
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc



Leer byte a byte (read)

Ahora si se desean leer solamente los primeros 6 bytes:

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcabc
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc

```
with open('files/data1.txt', 'r', encoding="utf-8") as f:  
    line = f.read(6)  
    line2 = f.read(6)
```

```
print(line)  
print(line2)
```

La salida del programa es:

```
Esta e  
s la l
```



Leer archivo línea por línea

Es posible leer un archivo línea por línea utilizando `readline()`. Esta es una forma eficiente en memoria de leer un archivo:

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcabc
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc

```
with open('files/data1.txt', 'r', encoding="utf-8") as f:  
    line = f.readline()  
    line2 = f.readline()
```

```
print(line, end='')
```

La salida del programa es:

Esta es la línea 1: abcabcabc



Leer archivo línea por línea

Es posible leer un archivo línea por línea utilizando `readline()`. Esta es una forma eficiente en memoria de leer un archivo:

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcabc
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc

```
with open('files/data1.txt', 'r', encoding="utf-8") as f:  
    line = f.readline()  
    line2 = f.readline()  
  
print(line, end='')
```

La salida del programa es:

Esta es la línea 1: abcabcabc



Cargar las líneas de un archivo como una lista

Es posible cargar todas las líneas del archivo como una lista si se utiliza::

```
with open("files/data1.txt", "r", encoding='utf-8') as f:  
    print("Nombre del archivo: ", f.name)  
    lista = f.readlines()  
print(lista)
```

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcab
Esta es la línea 2: abcabcab
Esta es la línea 3: abcabcab

La salida del programa es:

```
Nombre del archivo:  files/data1.txt  
['Esta es la línea 1: abcabcab',+  
'Esta es la línea 2: abcabcab',  
'Esta es la línea 3: abcabcab']
```



Procesar un archivo de forma eficiente

La forma más sencilla y eficiente de abrir y procesar un archivo de texto es:

data1.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcbc
Esta es la línea 2: abcabcbc
Esta es la línea 3: abcabcbc

```
with open("files/data1.txt", "r", encoding='utf-8') as work_data:  
    for line in work_data:  
        print(line, end='')
```

La salida del programa es:

Esta es la línea 1: abcabcbc
Esta es la línea 2: abcabcbc
Esta es la línea 3: abcabcbc



Agenda

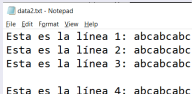
- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos**
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Escritura de archivos

A menudo se prefiere utilizar `a+` como el modo de apertura para escribir un archivo porque permite añadir datos al final del mismo. Utilizar `w+` borrará cualquier dato que existe en el archivo y pues producirá un archivo en limpio para trabajar.

El método por defecto para añadir datos en un archivo es `write`:



```
data2.txt - Notepad
File Edit Format View Help
Esta es la línea 1: abcabcbabc
Esta es la línea 2: abcabcbabc
Esta es la línea 3: abcabcbabc

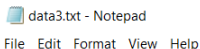
Esta es la línea 4: abcabcbabc
```

```
with open("files/data2.txt", "a+", encoding='utf-8') as work_data:
    work_data.write(" es la línea 4: abcabcbabc")
```



Escribir datos que no son de texto en un archivo

Si se desean escribir datos que no son cadenas de texto en un archivo es necesario convertir cada dato a string.



data3.txt - Notepad

File Edit Format View Help

1234
5678
9012

```
values = [1234, 5678, 9012]
```

```
with open("files/data3.txt", "w+") as work_data:  
    for value in values:  
        str_value = str(value)  
        work_data.write(str_value)  
        work_data.write("\n")
```



Seek: ubicarse en un archivo

A veces es necesario moverse al principio del archivo o a alguna posición específica en un archivo dado. La forma más fácil de hacerlo es utilizar `fileobject.seek(offset, from_what)`.

`offset` es el número de caracteres desde la posición `from_what`.

Hay algunos valores por defecto para la posición `from_what`:

- 0 - Principio del archivo
- 1 – posición actual
- 2 – al final del archivo
- Con archivos de texto 0 ó `seek(0, 2)`, lo llevarán al final del archivo.



Seek (ejemplo)

```
with open("files/data4.txt", "r", encoding='utf-8') as work_data:  
    work_data.seek(11,0)  
    for line in work_data:  
        print(line, end='')
```

La salida del programa es como la siguiente:

```
letras  
1234  
5678
```



data4.txt - Notepad

File Edit Format View Help

las quince letras

1234

5678



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés**
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Determinar tamaño de un archivo en bytes

Para obtener la longitud de un archivo o la posición en donde se encuentra en un archivo se puede usar `fileobject.tell()`. Para determinar el tamaño de un archivo en bytes se puede utilizar:

```
with open("files/data4.txt", "a+") as work_data:  
    print(work_data.tell())
```

La salida de este programa es: 31



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto**
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario



Edición de archivos

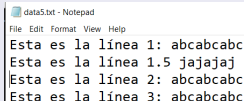
La forma más fácil de editar un archivo dado es guardar el archivo entero en una lista y utilizar `list.insert(i, x)` para insertar los datos nuevos. Una vez creada la lista se puede unir (`join`) de nuevo y escribir esto en el archivo.

```
# Open the file as read-only
with open("files/data5.txt", "r", encoding='utf-8') as f:
    list_content = f.readlines()

#print(list_content)

list_content.insert(1, "Esta es la línea 1.5 jajajaj")

# Re-open in write-only format to overwrite old file
with open("files/data5.txt", "w", encoding='utf-8') as f:
    contenido = "".join(list_content)
    f.write(contenido)
```



data5.txt - Notepad

File Edit Format View Help

Esta es la línea 1: abcabcabc
Esta es la línea 1.5 jajajaj
Esta es la línea 2: abcabcabc
Esta es la línea 3: abcabcabc



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios**
- 8 Pickle
- 9 Copiar un archivo binario



Listar archivos (os)

En las versiones modernas de python una alternativa a `os.listdir()` es utilizar `os.scandir()` (desde python 3.5) y `pathlib.Path()`. `os` trae funciones de utilidad para remover archivos `remove`, crear directorios `os.mkdir(path)`, etc.

```
import os
entries = os.scandir('files/')

for entry in entries:
    print(entry.name + ', es directorio: '
          + str(entry.is_dir()) + ', size: '
          + str(entry.stat().st_size)
          + ' bytes.')
```

```
alternancia.txt, es directorio: False, size: 12549 bytes.
data.txt, es directorio: False, size: 14 bytes.
data1.txt, es directorio: False, size: 94 bytes.
data2.txt, es directorio: False, size: 224 bytes.
data3.txt, es directorio: False, size: 18 bytes.
data4.txt, es directorio: False, size: 31 bytes.
data5.txt, es directorio: False, size: 189 bytes.
names.txt, es directorio: False, size: 220 bytes.
programming_powers.pkl, es directorio: False, size: 154 bytes.
python_es.txt, es directorio: False, size: 33754 bytes.
SalesJan2009.csv, es directorio: False, size: 130480 bytes.
testdir, es directorio: True, size: 0 bytes.
wdata.txt, es directorio: False, size: 153 bytes.
```



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 **Pickle**
- 9 Copiar un archivo binario



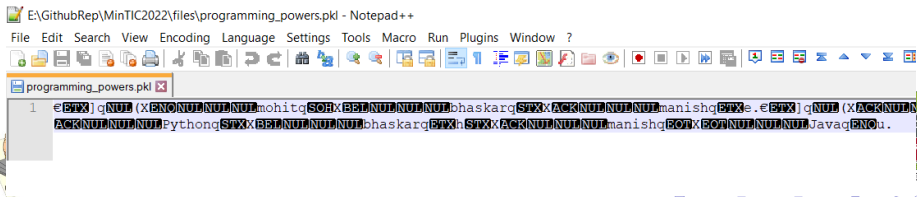
Guardar Estructuras de Datos en archivos: pickle

A veces se desea almacenar información compleja tales como la información que contiene un diccionario ó una lista. Aquí es donde podemos utilizar el módulo pickle de python para serializar objetos.

```
import pickle

name = ["mohit","bhaskar", "manish"]
skill = ["Python", "Python", "Java"]
dict1 = dict([(k,v) for k,v in zip(name, skill)])

with open("files/programming_powers.pkl", "wb") as pickle_file:
    pickle.dump(name, pickle_file)
    pickle.dump(skill,pickle_file)
    pickle.dump(dict1,pickle_file)
```



Cargar Estructuras de Datos en archivos: pickle

Para obtener los datos de vuelta es posible hacer lo siguiente:

```
import pickle

with open("files/programming_powers.pkl", "rb") as pickle_file:
    list1 = pickle.load(pickle_file)
    list2 = pickle.load(pickle_file)
    dict1 = pickle.load(pickle_file)
print(list1)
print(list2)
print(dict1)
```

La salida obtenida es:

```
['mohit', 'bhaskar', 'manish']
['Python', 'Python', 'Java']
'mohit': 'Python', 'bhaskar': 'Python', 'manish': 'Java'
```



Agenda

- 1 Introducción
- 2 Modos de apertura de archivos
- 3 Lectura de archivos
- 4 Escritura de archivos
- 5 Otras operaciones de interés
- 6 Editando un archivo de texto
- 7 Listar archivos y directorios
- 8 Pickle
- 9 Copiar un archivo binario**



Copiar un archivo binario

Una imagen de tipo jpg es un archivo de tipo binario. Con cierto procesamiento es posible crear una copia de la imagen:

```
with open("files/discurso.jpg", "rb") as imagen:  
    data = imagen.read()  
  
#print(data)  
  
with open("files/copy.jpg", "wb") as f:  
    f.write(data)
```



Sugerencia

Se sugiere consultar un manual de **Python** o de sus librerías para determinar si ya existe un método para lo que se quiera realizar con archivos.

