

Unidad 3

“Manejo de archivos en Python”

Informática III – ISM – UNL
2018



Archivos

- Hasta ahora nuestros programas se ejecutaban con datos almacenados en memoria del sistema, es decir, una vez que se terminaba el programa, esos datos dejaban de existir.
- Esto hace muy poco útiles los programas que tienen bases de datos, como por ejemplo el ejercicio de la Lista de Contactos Telefónicos.
- El archivo (fichero) es una forma de guardar información en unidades de almacenamiento (discos rígidos, pendrives, DVDs) que no se pierde cuando el programa se cierra o la PC se apaga.

Las “rutas” de los archivos

- Todo archivo está almacenado en algún lugar de nuestro sistema de archivos.
- Este lugar está identificado unívocamente por la ruta (“path” en inglés) que tiene ese archivo.
- La forma de esta ruta depende del sistema donde estemos, los sistemas Windows tienen una estructura a partir de las unidades de almacenamiento (C:\, D:\, E:\ etc).
 - Ejemplo:
`C:\Usuarios\Juan\Documentos\archivo.txt`
- Mientras que los sistemas Unix (como Linux) tienen una estructura de árbol, donde todo parte a partir del directorio raíz: /
 - Ejemplo:
`/home/Juan/Documentos/archivo.txt`

Las “rutas” de los archivos

- Veamos la ruta en Windows:

`C:\Usuarios\Juan\Documentos\archivo.txt`

- El nombre del archivo es `archivo.txt`, se encuentra en la carpeta `Documentos`, que a su vez está dentro de la carpeta `Juan`, que a su vez está dentro de la carpeta `Usuarios`, en la unidad `C:\`
- Podríamos tener otro archivo llamado `archivo.txt` en nuestro disco, pero nunca en la misma ruta.
- Ahora veamos la ruta en Linux:

`/home/Juan/Documentos/archivo.txt`

- Nuevamente, el archivo se llama `archivo.txt`, que se encuentra en el directorio `Documentos`, en el directorio del usuario `Juan` en el punto de montaje para espacio de usuarios `/home`.

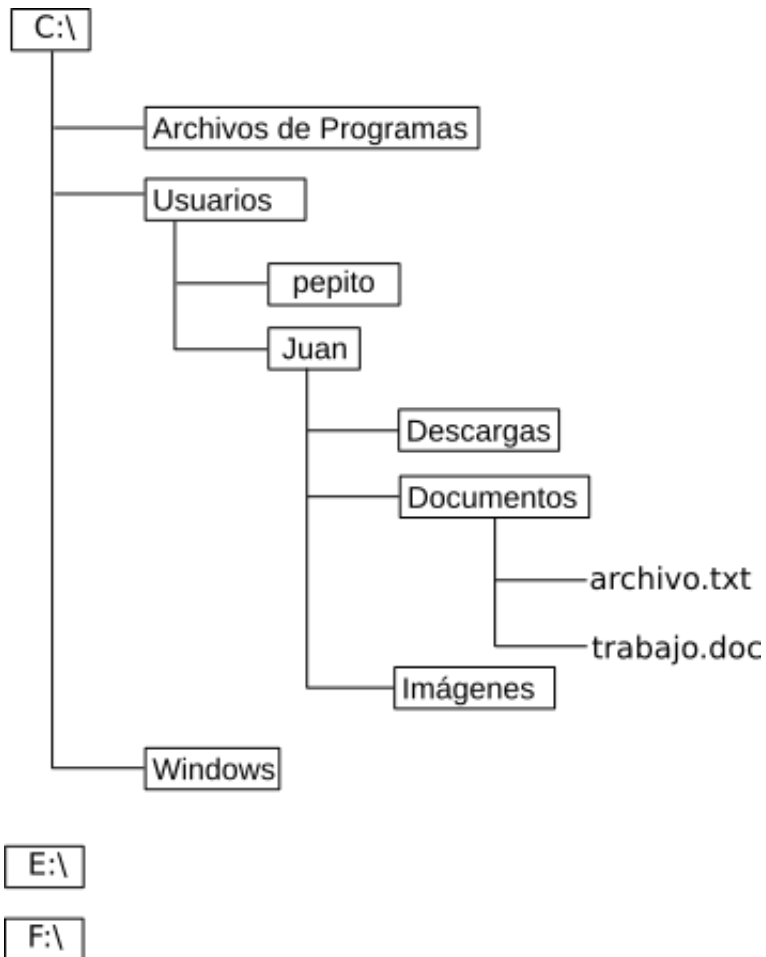


Las “rutas” de los archivos

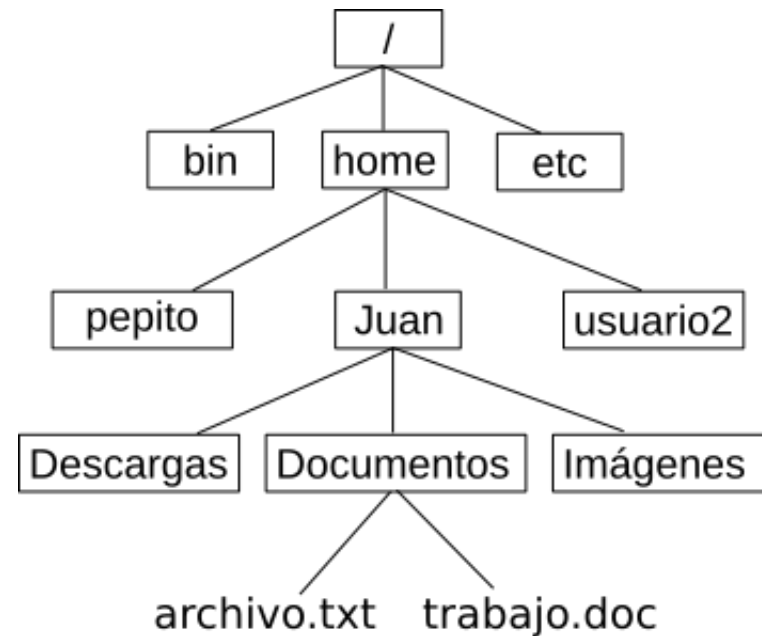
- Algunas cuestiones para diferenciar entre las rutas Windows y las rutas Linux:
- En Windows la estructura de archivos parte de una unidad de disco (puede ser un disco rígido, pendrive, disco externo, DVD, diskette, o unidad de red). En Linux la estructura está centralizada en la raíz, integrando las unidades físicas en ese raíz.
- En Windows se denominan “carpetas” mientras que en Linux se denominan “directorios”.
- En Windows las carpetas se separan con la contrabarra “\”, mientras que en Linux con la barra simple “/”.

Las “rutas” de los archivos

- Estructura en Windows



- Estructura en Linux



Archivos de texto plano

- Los archivos de texto plano son aquellos que poseen una codificación estándar que cualquier programa puede abrir.
- Son aquellos que podemos crear con `gedit` en linux o el Bloc de Notas en Windows, por ejemplo.
- No debemos confundirlos con los archivos `.doc` que generan softwares como *Microsoft Word* o *Libre Office*, ese es un formato con texto enriquecido (permite formatos como negritas, cursiva, líneas, imágenes, etc), en general ocupan más espacio, y principalmente el formato no es abierto.

Archivos en programación

- Para la programación los archivos son objetos que pueden ser creados, escritos o leídos.
- Siempre debemos seguir estos 3 pasos al usar archivos:
 - 1) Abrir el archivo indicando su ruta y el modo de trabajo:
 - Lectura: lee información del archivo pero no puede modificarla ni agregar.
 - Escritura: solo permite escribir en el archivo. Si el archivo existiese, borra todo su contenido.
 - Adición: Permite agregar información al archivo al final del mismo.
 - 2) Leer o escribir en el archivo.
 - 3) Cerrar el archivo.

Leer un archivo en Python

- Vamos a ver los 3 pasos del uso de archivo para el ejemplo de lectura:

```
# Paso 1: abrir un archivo en modo lectura ("r")
arch = open("archivo1.txt", "r")

# Paso 2: leer el archivo
for linea in arch:
    print(linea)

# Paso 3: cerrar el archivo
arch.close()
```

Leer un archivo en Python

- Vamos a ver los 3 pasos del uso de archivo para el ejemplo de lectura:

```
# Paso 1: abrir un archivo en modo lectura ("r")
```

```
arch = open("archivo1.txt","r")
```

```
# Paso 2: leer el archivo
```

```
for linea in arch:  
    print(linea)
```

```
# Paso 3: cerrar el archivo
```

```
arch.close()
```

- En el paso 1) usamos la función open para crear un objeto del tipo Archivo.
- La función permite 2 parámetros:
 - La ruta del archivo: si no se especifica completa, entonces será relativa a la ruta donde está guardado el programa.
 - El modo de apertura, "r" significa lectura ("read")

Leer un archivo en Python

- Vamos a ver los 3 pasos del uso de archivo para el ejemplo de lectura:

```
# Paso 1: abrir un archivo en modo lectura ("r")
```

```
arch = open("archivo1.txt","r")
```

```
# Paso 2: leer el archivo
```

```
for linea in arch:  
    print(linea)
```

```
# Paso 3: cerrar el archivo
```

```
arch.close()
```

- En el paso 2) mostramos por pantalla línea por línea el contenido del archivo.
- Los objetos de tipo Archivo pueden recorrerse con un `for-in`.

Leer un archivo en Python

- Vamos a ver los 3 pasos del uso de archivo para el ejemplo de lectura:

```
# Paso 1: abrir un archivo en modo lectura ("r")
```

```
arch = open("archivo1.txt","r")
```

```
# Paso 2: leer el archivo
```

```
for linea in arch:  
    print(linea)
```

```
# Paso 3: cerrar el archivo
```

```
arch.close()
```

- En el paso 3) cerramos el objeto de tipo Archivo con el método `close()`.

Leer un archivo en Python

- Probemos ejecutar el programa:

```
Shell

>>> %Run lectura.py

Traceback (most recent call last):
  File "/home/scientist/Documents/ISM/Teorias/11-Archivos/lectura.py", line 3, in <module>
    arch = open("archivo.txt","r")
FileNotFoundError: [Errno 2] No such file or directory: 'archivo.txt'
```

- Qué sucedió? Si el archivo a leer no está en la ruta señalada, entonces tenemos un error.
- Podemos solucionar eso usando un bloque `try - except` que prueba un código y captura cualquier error que surja.

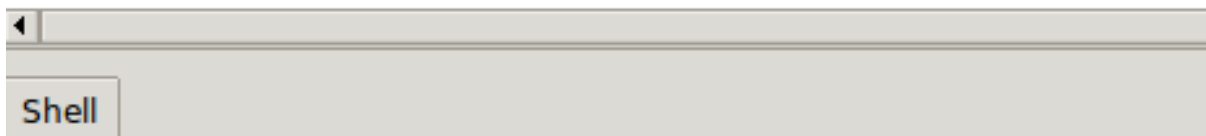
Leer un archivo en Python

- Todo lo que se escriba dentro del bloque try está “a prueba”, si algo falla se dirige al bloque except (excepción):

```
try:
    # Paso 1: abrir un archivo en modo lectura ("r")
    arch = open("archivo.txt", "r")

    # Paso 2: leer el archivo
    for linea in arch:
        print(linea)

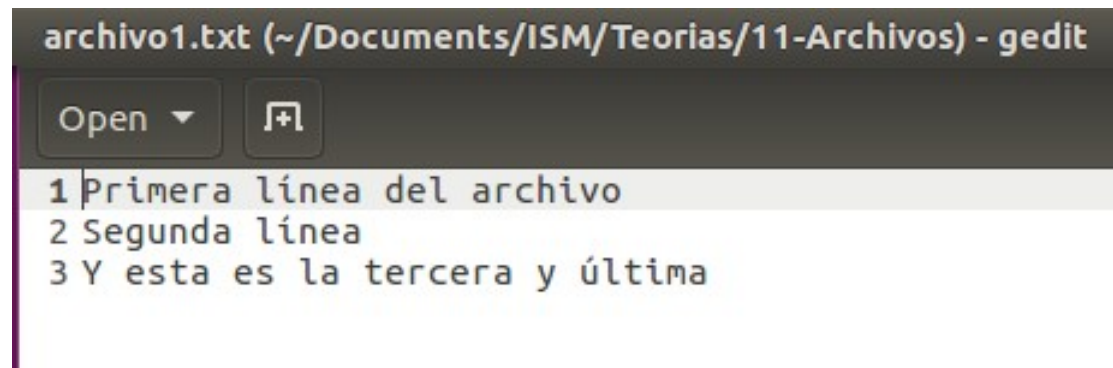
    # Paso 3: cerrar el archivo
    arch.close()
except IOError:
    print("Ha ocurrido un error al intentar abrir el archivo")
```



```
>>> %Run lectura.py
Ha ocurrido un error al intentar abrir el archivo
```

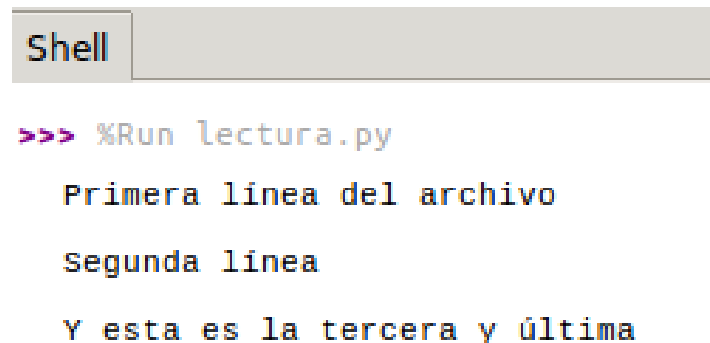
Leer un archivo en Python

- Ahora si, probemos con un archivo que exista:.
- En el disco se creó este archivo:



```
archivo1.txt (~/Documents/ISM/Teorias/11-Archivos) - gedit
Open ▾ [icon]
1 Primera línea del archivo
2 Segunda línea
3 Y esta es la tercera y última
```

- La salida del programa es:



```
Shell
>>> %Run lectura.py
Primera línea del archivo
Segunda línea
Y esta es la tercera y última
```

Leer un archivo en Python

- Ahora si, probemos con un archivo que exista:.
- En el disco se creó este archivo:

```
archivo1.txt (~/.Documents/ISM/Teorias/11-Archivos) - gedit
Open ▾ [icon]
1 Primera línea del archivo
2 Segunda línea
3 Y esta es la tercera y última
```

- La salida del programa es:

```
Shell
>>> %Run lectura.py
Primera línea del archivo
Segunda línea
Y esta es la tercera y última
```

Cada línea salió con un salto entre medio
¿Qué pasó?

Leer un archivo en Python

- Es simple, cada línea termina con un carácter oculto: el “\n” o sea, salto de línea
- La función `print()` escribe una línea nueva cada vez que se ejecuta, pero al leer el “\n” realiza un salto extra.
- La solución es preguntar si al final de la línea hay un carácter de salto, y en ese caso extraerlo de la cadena a mostrar.

Leer un archivo en Python

```
lectura.py x
1  try:
2
3      # Paso 1: abrir un archivo en modo lectura ("r")
4
5      arch = open("archivo1.txt","r")
6
7      # Paso 2: leer el archivo
8
9      for linea in arch:
10         # es el último caracter un salto?
11         if linea[-1] == "\n":
12             # entonces solo muestro del principio
13             # hasta el anteúltimo caracter
14             linea = linea[:-1]
15             print(linea)
16
17         # Paso 3: cerrar el archivo
18
19         arch.close()
20
21 except IOError:
22     print("Ha ocurrido un error al intentar abrir el archivo")
```

Shell

```
>>> %Run lectura.py
Primera línea del archivo
Segunda línea
Y esta es la tercera y última
```

Leer un archivo en Python

- Una cosa para notar es que modificamos la variable línea:

```
línea = línea[:-1]
```

- Pero esto no significa que hayamos modificado el archivo en el disco, ya que lo que hacemos al leer es trabajar con una copia en memoria del contenido de la línea.

Leer un archivo en Python

- El ejemplo anterior leía un archivo de texto línea por línea. Ahora veamos un ejemplo de lectura “caracter por caracter”:

```
lectura.py x
1  try:
2
3      arch = open("archivo1.txt","r")
4
5      caracter = arch.read(1)
6      contador = 0
7
8      while caracter != '':
9
10         contador += 1
11         caracter = arch.read(1)
12
13     arch.close()
14
15     print ("Cantidad de caracteres del archivo: %d"%contador)
16
17 except IOError:
18     print("Ha ocurrido un error al intentar abrir el archivo")

Shell

>>> %Run lectura.py
Cantidad de caracteres del archivo: 69
```

Leer un archivo en Python

- En el ejemplo se usa el método `read()` que lee la cantidad de caracteres que se pasen como parámetro.
- En el caso del ejemplo se utiliza `read(1)` es decir, muestra cual es el siguiente carácter a leer.
- Cuando ya no hay nada para leer, `read()` retorna una cadena vacía. Es ahí donde detenemos el bucle.

Leer un archivo en Python

- Veamos ahora otro método para leer línea por línea, esta vez usando el método `readline()` (“leer una línea”).

```
1  try:
2
3      arch = open("archivo1.txt","r")
4
5      linea = arch.readline()
6
7      while linea != '':
8
9          if linea[-1] == "\n":
10             linea = linea[:-1]
11
12             print(linea)
13
14             linea = arch.readline()
15
16         arch.close()
17
18 except IOError:
19     print("Ha ocurrido un error al intentar abrir el archivo")
```

Shell

```
>>> %Run lectura.py
Primera línea del archivo
Segunda línea
Y esta es la tercera y última
```

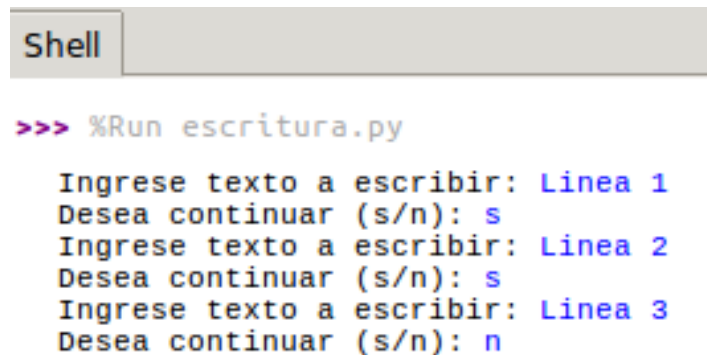
Escribir un archivo en Python

- Por ejemplo, veamos un código en el cual se ingresa al archivo una línea escrita por teclado.

```
1  try:
2
3      arch = open("archivo2.txt","w")
4      seguir = "s"
5
6      while seguir == "s":
7
8          entrada = input("Ingrese texto a escribir: ")
9
10         arch.write("%s"%(entrada))
11
12         seguir = input("Desea continuar (s/n): ")
13
14     arch.close()
15
16 except IOError:
17     print("Error al escribir el archivo")
18
19
```

Escribir un archivo en Python

- Debemos abrir el archivo con el parámetro “w” (write = escritura)
- El método `write()` escribe la cadena que se pasa como parámetro.



```
Shell

>>> %Run escritura.py

Ingrese texto a escribir: Linea 1
Desea continuar (s/n): s
Ingrese texto a escribir: Linea 2
Desea continuar (s/n): s
Ingrese texto a escribir: Linea 3
Desea continuar (s/n): n
```

- El archivo queda así:



```
escritura.py x archivo2.txt x

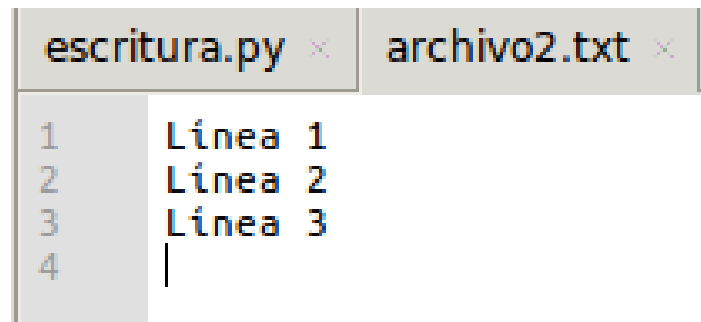
1 Linea 1Linea 2Linea 3
```

- Cada vez que se ejecuta `write()` se escribe en el archivo.

Escribir un archivo en Python

- Si agregamos el salto de línea cada vez que escribimos en el archivo.

```
arch.write("%s\n"%(entrada))
```



The screenshot shows a code editor with two tabs: 'escritura.py' and 'archivo2.txt'. The 'escritura.py' tab is active, displaying a Python script with four lines of code. The first three lines are 'Línea 1', 'Línea 2', and 'Línea 3', each followed by a newline character. The fourth line is a single vertical bar '|'. The 'archivo2.txt' tab is also visible, showing the output of the script, which is the same three lines of text followed by a newline character.

```
escritura.py x archivo2.txt x
1 Línea 1
2 Línea 2
3 Línea 3
4 |
```

Escribir un archivo en Python

- Tratememos de programar un algoritmo para escribir la tabla de multiplicar de un número ingresado por teclado.
- Es decir, si ingresamos 3, deberíamos poder escribir un archivo de la siguiente manera:

```
0  
3  
6  
9  
12  
15  
18  
21  
24  
27  
30|
```

Escribir un archivo en Python

```
1  try:
2
3      arch = open("multiplicar.txt","w")
4
5      num = int(input("Ingrese un número: "))
6
7      for i in range(11):
8
9          arch.write(num*i)
10
11
12      arch.close()
13
14  except IOError:
15      print("Error al escribir el archivo")
16
```

Shell

```
>>> %Run escritura_formato.py
Ingrese un número: 3
Traceback (most recent call last):
  File "/home/scientist/Documents/ISM/Teorias/11-A
    arch.write(num*i)
TypeError: write() argument must be str, not int
```

El error aparece en el método `write()` ya que el parámetro debe ser del tipo cadena.

Escribir un archivo en Python

```
1  try:
2
3      arch = open("multiplicar.txt","w")
4
5      num = int(input("Ingrese un número: "))
6
7      for i in range(11):
8
9          arch.write("%s\n" % (num*i))
10
11
12      arch.close()
13
14  except IOError:
15      print("Error al escribir el archivo")
16
```

Shell

```
>>> %Run escritura_formato.py
```

```
    Ingrese un número: 3
```

Agregar texto a un archivo

- Hay momentos en los que necesitamos agregar información a un archivo en lugar de sobrescribir todo.
- Para eso usamos el método de apertura con el parámetro “a” (“append” = agregar)

```
try:
    arch = open("multiplicar.txt","a")
    num = int(input("Ingrese un número: "))
    arch.write("%d\n"%(num*11))
    arch.close()
except IOError:
    print("Error al escribir el archivo")
```

Agregar texto a un archivo

- De esa manera agregamos al final del archivo la cadena pasada en el método `write()`

```
0  
3  
6  
9  
12  
15  
18  
21  
24  
27  
30  
33
```