UTN AVELLANEDA
Tecnicatura Universitaria en Programación

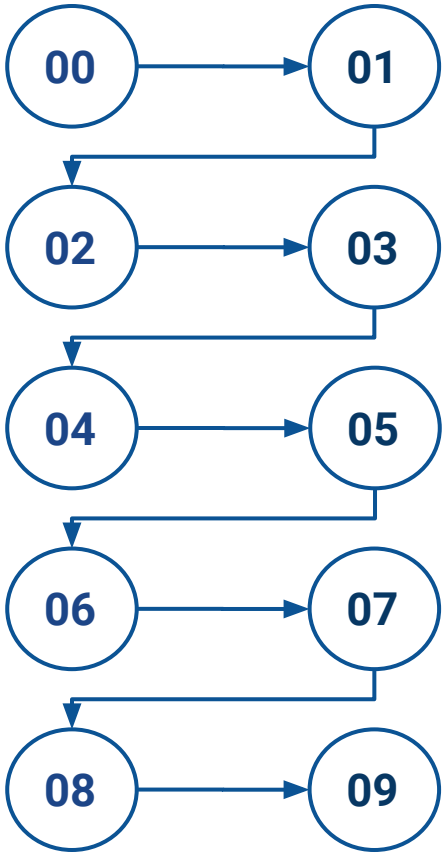
Archivos

Programación y Laboratorio I

Versión 22.2

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Archivos



Archivos de texto

Los **archivos de texto** contienen caracteres legibles, es posible no solo leer dicho contenido sino también modificarlo usando un editor de texto.

Archivos binarios

Un **archivo binario** es cualquier archivo que no se pueda interpretar en forma de texto: una imagen, un sonido o incluso un archivo comprimido.

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Modos de apertura

MODO	DESCRIPCIÓN
r	Abre un archivo de texto sólo para lectura
rb	Abre un archivo binario sólo para lectura
r+	Abre un archivo de texto para escritura y lectura
w	Abre un archivo de texto sólo para escritura (si existe lo sobreescribe)
wb	Abre un archivo sólo para escritura (si existe lo sobreescribe)
w+	Abre un archivo de texto para escritura y lectura (si existe lo sobreescribe)
a	Abre un archivo para anexar información

Modos de apertura: 'r' 'r+'

r -> abre un archivo sólo para lectura. El puntero al archivo está localizado al comienzo del archivo. Este es el modo predeterminado de la función.

r+ -> abre un archivo para escritura y lectura. El puntero del archivo está localizado al comienzo del archivo.

Modos de apertura: 'w' 'w+'

w -> abre un archivo solo para escritura. Sobreescribe el archivo si este ya existe. Si el archivo no existe lo crea.

w+ -> abre un archivo para escritura y lectura. Sobreescribe el archivo si este ya existe. Si el archivo no existe lo crea.

Modos de apertura: 'a'

a -> abre un archivo para anexo. El puntero del archivo está al final del archivo si este existe. Es decir, el archivo está en modo anexo. Si el archivo no existe, crea un nuevo archivo para escritura.

Abrir un archivo: open

Para abrir un archivo con Python, podemos usar la función **open**.

```
archivo = open(nombre_archivo, modo)
```

archivo objeto file, el cual será utilizado para llamar a otros métodos asociados con el archivo.

nombre_archivo string que contiene el nombre del archivo al que queremos acceder.

modo string que contiene el modo de apertura del archivo.

Cerrar un archivo: close

Para cerrar un archivo con Python, podemos usar la función **close**.

```
archivo.close()
```

`archivo` objeto file que fue obtenido con el método open.

El objeto: file

Una vez que un archivo está abierto y se obtiene el objeto **file** permite no sólo operar con él sino también obtener mucha información relacionada con ese archivo.

El objeto: file

```
archivo = open(nombre_archivo, modo)
```

`archivo.closed` retorna **True** si el archivo está cerrado, si no, **False**.

`archivo.mode` retorna el modo de acceso con el que el archivo ha sido abierto.

`archivo.name` retorna el nombre del archivo.

Leer un archivo: read

El método **read** permite extraer un string que contenga todos los carácteres del archivo.

```
# Abrimos el archivo en modo lectura y escritura
archivo = open('archivo.txt', 'r+')
texto = archivo.read()
print('El contenido del archivo es: ' + texto)
# Cerramos el archivo
archivo.close()
```

Leer un archivo: read

Es posible limitar al método read a que lea cierta cantidad de bytes **read**(cantidad)

```
# Abrimos el archivo en modo lectura y escritura
archivo = open('archivo.txt', 'r+')
archivo.seek(100)
texto = archivo.read(10)
print('El contenido del archivo es: ' + texto)
# Cerramos el archivo
archivo.close()
```

Leer un archivo: readlines

Permite obtener una lista con todas las líneas que contiene el archivo.

```
archivo = open('archivo.txt', 'r+')
for linea in archivo.readlines():
    print(linea, end="")
# Cerramos el archivo
archivo.close()
```

Leer un archivo: file

Si solamente requerimos recorrer el archivo línea por línea, el objeto **file** es iterable.

```
archivo = open('archivo.txt', 'r+')
for linea in archivo:
    print(linea, end="")
# Cerramos el archivo
archivo.close()
```

Leer un archivo: readline

Utilizando la función **readline**() es posible leer desde la posición actual del puntero del archivo hasta el final de la línea y luego posicionar el puntero al comienzo de la siguiente línea.

```
archivo = open('archivo.txt', 'r+')
print(archivo.tell()) #Indica en que byte esta el puntero 0
linea = archivo.readline()
print(linea,end="") # Hola mundo
print(archivo.tell()) #Indica en que byte esta el puntero 11
# Cerramos el archivo
archivo.close()
```

Escribir: write

Escribe una cadena de caracteres dentro del archivo.

```
archivo = open('archivo.txt', 'w')
archivo.write('Primer linea de texto\n')
archivo.write('segunda linea\n')
archivo.write('tercera linea\n')
archivo.close()
```

Escribir: writelines

Escribe una lista de cadena de caracteres dentro del archivo. El parámetro que recibe el método podrá ser cualquier iterable.

```
archivo = open('archivo.txt', 'w')
lineas_texto = ['Primer linea de texto\n',
                'segunda linea\n',
                'tercera linea\n']
archivo.writelines(lineas_texto)
archivo.close()
```

Mover puntero: seek

El método **seek** permite modificar la posición actual del puntero.

```
archivo = open('archivo.txt', 'r+')
archivo.seek(11)
print(archivo.tell()) #Esta en el byte 11
linea = archivo.readline()
print(linea,end="") # Hola mundo
archivo.close()
```

Administrador de contexto: with

Podemos usar la sentencia **with** para abrir archivos en Python y dejar que el intérprete se encargue de cerrar el mismo.

```
with open('archivo.txt', 'r+') as archivo:
    for linea in archivo:
    print(linea, end="")
```

CSV

El CSV es un tipo de documento que representa los datos de forma parecida a una tabla, es decir, organizando la información en filas y columnas.

Las columnas son separadas, por un signo de puntuación (, ; .) u otro carácter y las diferentes filas suelen separarse por un salto de línea.

JSON

El nombre es un acrónimo de las siglas en inglés de **J**ava**S**cript **O**bject **N**otation.

JSON es un formato para el intercambio de datos basado en texto. Por lo que es fácil de leer tanto para una persona como para una máquina.

JSON (Escritura)

El paquete **json** permite traducir un diccionario a formato JSON utilizando el método **dump**.

```
import json

data = {}
data['clientes'] = []
data['clientes'].append({ 'nombre': 'Juan', 'edad': 27})
data['clientes'].append({ 'nombre': 'Ana', 'edad': 26})

with open('data.json', 'w') as file:
    json.dump(data, file, indent=4, ensure_ascii=False )
```

JSON (Lectura)

La lectura es similar al proceso de escritura, se debe abrir un archivo y procesar esté utilizando el método **load**.

```
import json

with open('data.json') as file:
    data = json.load(file)
```