

CÓNQUER BLOCKS

PYTHON

USO DE ARCHIVOS

OBJETIVO

Aprender a trabajar con archivos y manejar grandes cantidades de datos

└─ **Necesario para analizar y modificar información**

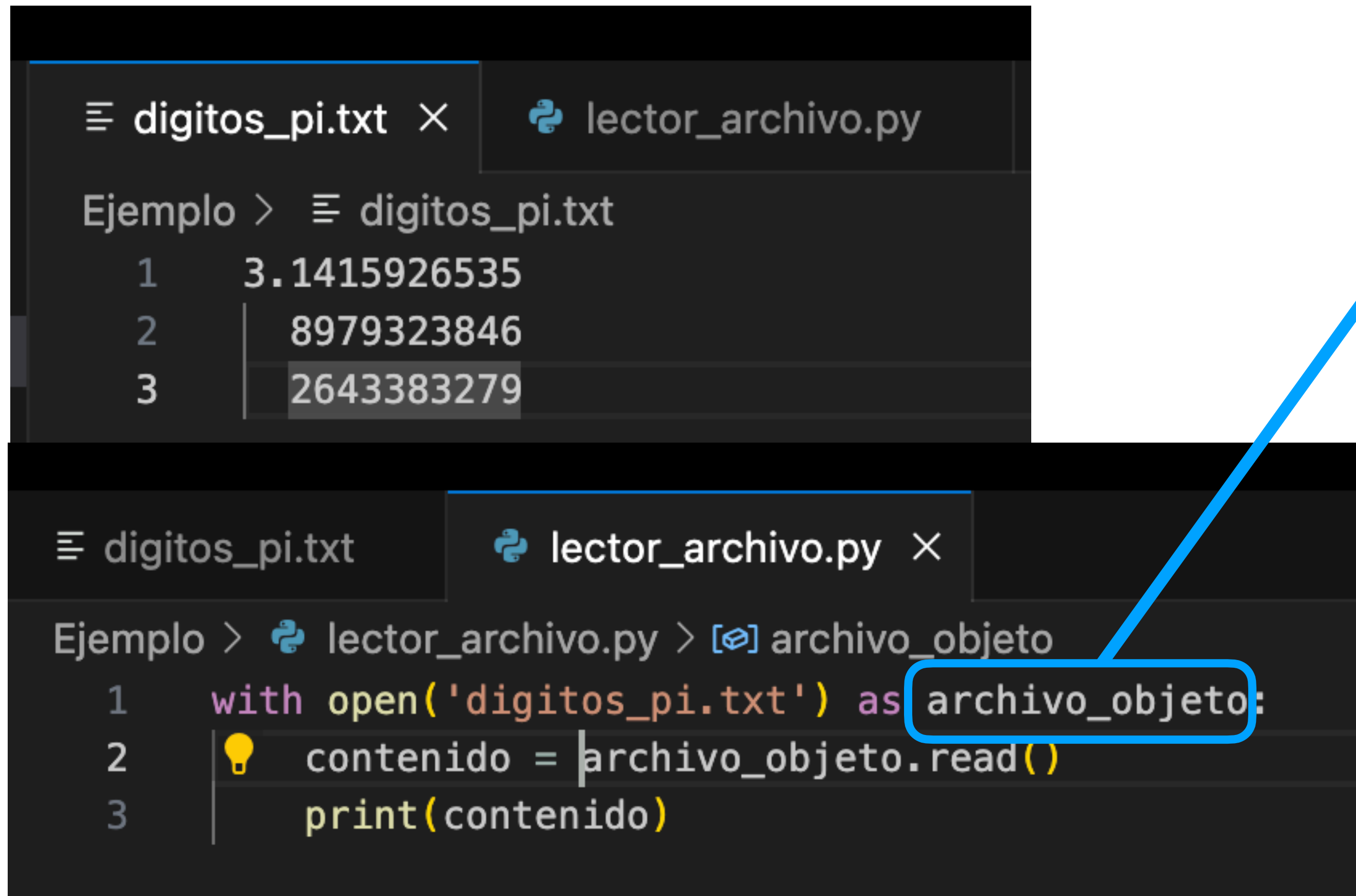
Guardar datos para hacer nuestros programas más cómodos para el usuario

└─ **Poder reanudar la ejecución de un script o guardar mensajes de error**

LEER UN ARCHIVO

LEER INFORMACIÓN DENTRO DE LA MEMORIA

`open()` devuelve un objeto que representa el archivo



```
digitos_pi.txt × lector_archivo.py

Ejemplo > digitos_pi.txt
1 3.1415926535
2 8979323846
3 2643383279

digitos_pi.txt lector_archivo.py ×

Ejemplo > lector_archivo.py > [🔗] archivo_objeto
1 with open('digitos_pi.txt') as archivo_objeto:
2     💡 contenido = archivo_objeto.read()
3     print(contenido)
```

Python se encarga de manejar el cierre del archivo

LEER UN ARCHIVO

USAR FILE PATHS - RELATIVO

Linux / OS X

```
with open('path/directorio/digitos_pi.txt') as archivo_objeto:  
    contenido = archivo_objeto.read()  
    print(contenido)
```

Windows

```
with open('path\\directorio\\digitos_pi.txt') as archivo_objeto:  
    contenido = archivo_objeto.read()  
    print(contenido)
```

LEER UN ARCHIVO

USAR FILE PATHS - ABSOLUTO

Linux / OS X

```
file_path = '/home/elena/otros_archivos/archivos_texto/filename.txt'
with open(file_path ) as archivo_objeto:
    contenido = archivo_objeto.read()
    print(contenido)
```

Windows

```
file_path = 'C:\\Users\\elena\\otros_archivos\\archivos_texto\\filename.txt'
with open(file_path ) as archivo_objeto:
    contenido = archivo_objeto.read()
    print(contenido)
```

LEER UN ARCHIVO

LEER LINEA POR LINEA

```
with open('digitos_pi.txt') as archivo_objeto:  
    for linea in archivo_objeto:  
        print(linea)
```

```
● (cblocks) MacBook-Pro-5:Ejem  
  /Ejemplo/lector_archivo.py"  
  3.1415926535  
  
    8979323846  
  
    2643383279
```


LEER UN ARCHIVO

GUARDAR INFORMACION DE UN ARCHIVO

```
filename = 'digitos_pi.txt'
with open(filename) as archivo_objeto:
    lineas = archivo_objeto.readlines()
    for linea in lineas:
        print(linea.rstrip())
```

```
(cblocks) MacBook-Pro-5:Ejemp
/Ejemplo/lector_archivo.py"
3.1415926535
8979323846
2643383279
```

LEER UN ARCHIVO

MANEJAR INFORMACION DE UN ARCHIVO

```
filename = 'digitos_pi.txt'
with open(filename) as archivo_objeto:
    lineas = archivo_objeto.readlines()
pi_string = ''
for linea in lineas:
    pi_string += linea.strip()
print(pi_string)
print(len(pi_string))
```

```
(CÓNQUERBLOCKS) MacBook-Pro-3: Ejemplo_Lector_Archivos
/Ejemplo/lector_archivo.py"
3.141592653589793238462643383279
32
```


ESCRIBIR EN UN ARCHIVO

ARCHIVOS VACIOS

```
filename = "programa.txt"
with open(filename, "w") as archivo_objeto:
    archivo_objeto.write("Estoy aprendiendo python")
```

ESCRIBIR EN UN ARCHIVO

ARCHIVOS VACIOS

```
filename = "programa.txt"  
with open(filename, "w") as archivo_objeto:  
    archivo_objeto.write("Estoy aprendiendo python")
```



MODOS OPEN

Modo	Descripción
r	Modo de lectura. Abre el archivo para lectura (predeterminado).
w	Modo de escritura. Abre el archivo para escritura, sobrescribe el archivo si existe, o crea uno nuevo si no existe.
a	Modo de anexar. Abre el archivo para escritura, pero agrega contenido al final en lugar de sobrescribirlo. Crea un nuevo archivo si no existe.
x	Modo de creación exclusiva. Abre el archivo para escritura, pero falla si el archivo ya existe.
b	Modo binario. Abre el archivo en modo binario. Se utiliza junto con otros modos, como 'rb' para lectura binaria y 'wb' para escritura binaria.
t	Modo de texto (predeterminado). Abre el archivo en modo de texto para lectura o escritura. Se utiliza junto con otros modos, como 'rt' para lectura de texto y 'wt' para escritura de texto.

MODOS OPEN

r+	Modo de actualización para lectura y escritura. El puntero se coloca al principio del archivo.
w+	Modo de actualización para lectura y escritura. El archivo se sobrescribe o crea uno nuevo.
a+	Modo de actualización para lectura y escritura. El puntero se coloca al final del archivo.

ESCRIBIR EN UN ARCHIVO

ARCHIVOS VACIOS

```
filename = "programa.txt"
with open(filename, "w") as archivo_objeto:
    archivo_objeto.write("Estoy aprendiendo python.")
    archivo_objeto.write("Estoy en el modulo avanzado.")
```

≡ *programa.txt* ×

Ejemplo > ≡ programa.txt

```
1  Estoy aprendiendo python.Estoy en el modulo avanzado.
```

ESCRIBIR EN UN ARCHIVO

ARCHIVOS VACIOS

```
1 filename = "programa.txt"
2 with open(filename, "w") as archivo_objeto:
3     ✨ archivo_objeto.write("Estoy aprendiendo python. \n")
4     archivo_objeto.write("Estoy en el modulo avanzado.")
```

≡ programa.txt ×

Ejemplo > ≡ programa.txt

```
1 Estoy aprendiendo python.
2 Estoy en el modulo avanzado.
```


AÑADIR ELEMENTOS A UN ARCHIVO

ARCHIVOS VACIOS

```
filename = "programa.txt"
with open(filename, "w") as archivo_objeto:
    archivo_objeto.write("Estoy aprendiendo python. \n")
    archivo_objeto.write("Estoy en el modulo avanzado.\n")

with open(filename, 'a') as file_object:
    file_object.write("Estoy creando un nuevo set de datos.\n")
    file_object.write("Y separo las lineas correctamente.\n")
```

programa.txt ×

```
ejemplo > ≡ programa.txt
1  Estoy aprendiendo python.
2  Estoy en el modulo avanzado.
3  Estoy creando un nuevo set de datos.
4  Y separo las lineas correctamente.
5
```

AÑADIR ELEMENTOS A UN ARCHIVO

ARCHIVOS VACIOS

```
filename = "programa.txt"
with open(filename, "w") as archivo_objeto:
    archivo_objeto.write("Estoy aprendiendo python. \n")
    archivo_objeto.write("Estoy en el modulo avanzado.\n")

with open(filename, 'a') as file_object:
    file_object.write("Estoy creando un nuevo set de datos.\n")
    file_object.write("Y separo las lineas correctamente.\n")

f = open(nombre_archivo, "w+")
f.write(contenido en forma de string)
```

programa.txt ×

```
ejemplo > ≡ programa.txt
1  Estoy aprendiendo python.
2  Estoy en el modulo avanzado.
3  Estoy creando un nuevo set de datos.
4  Y separo las lineas correctamente.
5
```

MANEJAR ARCHIVOS CON NUMPY

LECTURA:

```
import numpy as np

# Leer datos de un archivo CSV
data = np.loadtxt('datos.csv', delimiter=',')

# Leer datos de un archivo de texto
data = np.loadtxt('datos.txt')

# Leer datos de un archivo con encabezados
data = np.genfromtxt('datos.csv', delimiter=',', skip_header=1)
```

MANEJAR ARCHIVOS CON NUMPY

ESCRITURA:

```
import numpy as np

# Crear datos de ejemplo
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Guardar datos en un archivo CSV
np.savetxt('datos.csv', data, delimiter=',')

# Guardar datos en un archivo de texto
np.savetxt('datos.txt', data)

# Guardar datos en un archivo con encabezados
header = 'Columna 1, Columna 2, Columna 3'
np.savetxt('datos.csv', data, delimiter=',', header=header, comments='')
```


MANEJAR ARCHIVOS CON NUMPY

ESCRITURA:

```
import numpy as np

# Crear datos de ejemplo
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Guardar datos en un archivo CSV
np.savetxt('datos.csv', data, delimiter=',')

# Guardar datos en un archivo de texto
np.savetxt('datos.txt', data)

# Guardar datos en un archivo con encabezados
header = 'Columna 1, Columna 2, Columna 3'
np.savetxt('datos.csv', data, delimiter=',', header=header, comments='')
```

TRABAJAR CON ARCHIVOS JSON

**En muchos programas queremos guardar el input de los usuarios.
(p.e. preferencias en un juego o datos de visualización)**

Guardaremos la información en estructuras de datos como listas o diccionarios.

Cuando se cierre el programa queremos que la información de los settings no se pierda. Para ello podremos guardar la información en archivos (por ejemplo de tipo json)

TRABAJAR CON ARCHIVOS JSON

JSON = JavaScript Object Notation

Desarrollado originalmente para JavaScript.

Es uno de los formatos más usados en muchos lenguajes de programación, también python.

TRABAJAR CON ARCHIVOS JSON

```
import json

numeros = [2, 3, 5, 7, 11, 13]
filename = 'numeros.json'
with open(filename, 'w') as f_obj:
    json.dump(numeros, f_obj)
```

TRABAJAR CON ARCHIVOS JSON

```
import json

numeros = [2, 3, 5, 7, 11, 13]
filename = 'numeros.json'
with open(filename, 'w') as f_obj:
    json.dump(numeros, f_obj)
```

Datos

Archivo

Convierte un objeto de python en un json string

```
{ } numeros.json × ≡ programa.tx
```

```
Ejemplo > { } numeros.json > ...
```

```
1 [2, 3, 5, 7, 11, 13]
```

TRABAJAR CON ARCHIVOS JSON

```
import json
filename = 'numeros.json'
with open(filename) as f_obj:
    numeros = json.load(f_obj)
    print(numeros)
```

TRABAJAR CON ARCHIVOS JSON

```
import json
filename = 'numeros.json'
with open(filename) as f_obj:
    numeros = json.load(f_obj)
    print(numeros)
```

```
(cblocks) MacBook-Pro-5:Ej  
/Ejemplo/json_files.py"  
[2, 3, 5, 7, 11, 13]  
<class 'list'>
```

→ Abrimos el archivo en reading mode

↓
Convierte json string en un objeto de python

CÔNQUER BLOCKS