



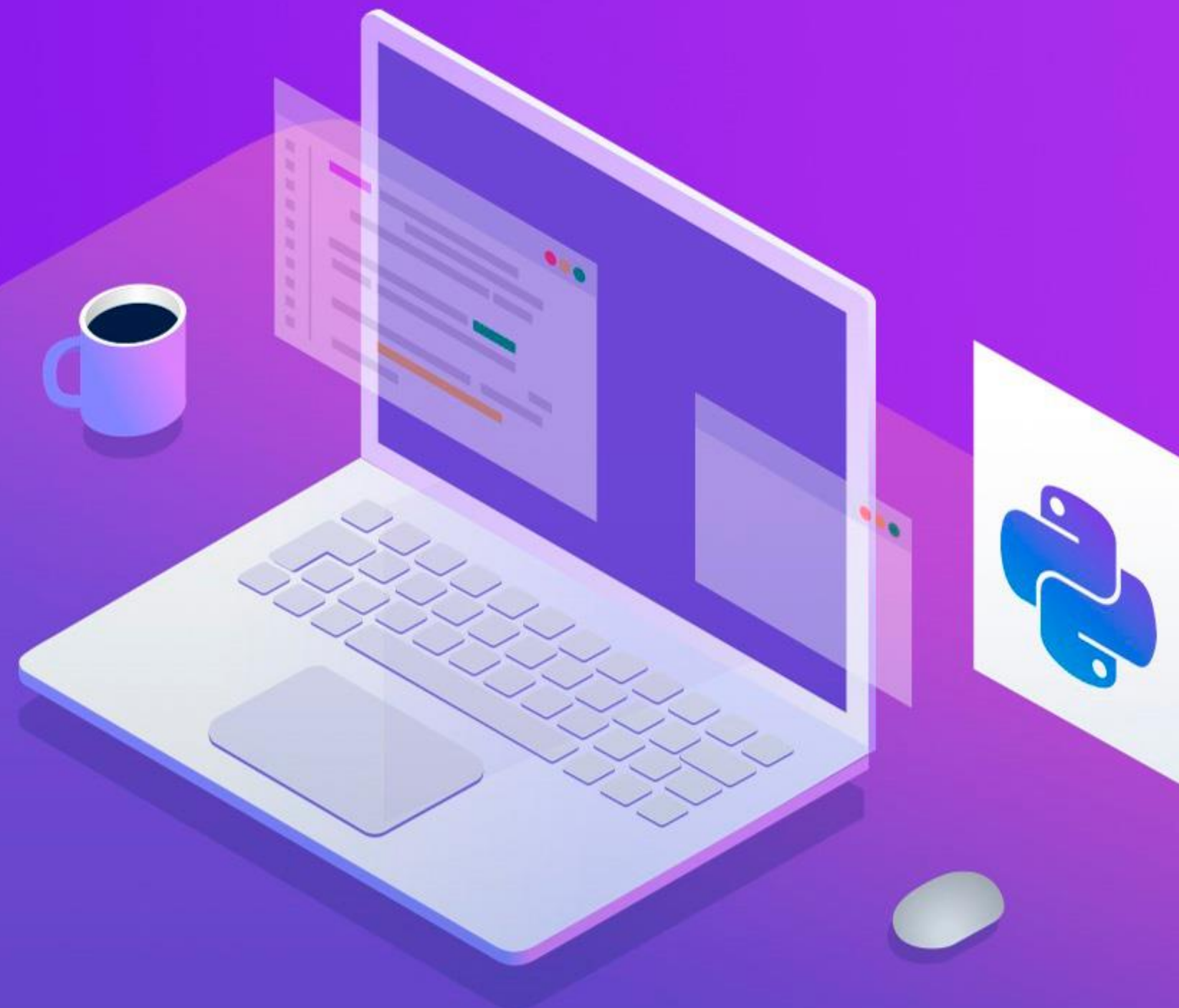
PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

ARCHIVOS Y

PROCESAMIENTO DE

DATOS

>>> Parte 1: Archivos



¿Cómo subir datos a Python?

Ya manejamos todas las herramientas necesarias para poder almacenar y trabajar con datos dentro de Python.



Ahora, surge la siguiente duda, ¿cómo podríamos cargar estos datos?



Después de modificarlos o procesarlos, ¿cómo podríamos guardar esta información?



Interacción con archivos

Python puede interactuar con varios tipos de archivos. Aquí, aprenderemos sobre uno en particular.

Archivos de texto plano

Tienen un contenido simple y fácil de interpretar.

Son el típico archivo de texto que uno crea al ocupar el "Bloc de notas" (o *Notepad* en inglés).

Python posee herramientas para poder leer estos archivos, así como también escribir sobre ellos (o crear nuevos).

¿Para qué sirve trabajar con Archivos?

Para incorporar grandes volúmenes de información a Python.

Por ello, hemos visto cómo añadir información de forma manual a Python:

Creando una o más variables de forma explícita.

EJEMPLO

```
a="texto"
```

Creando una o más listas de forma explícita.

EJEMPLO

```
lista1=["a","b","c"]
```

Creando una o más listas de listas de forma explícita.

EJEMPLO

```
matriz = [["a","b","c"],["d","e","f"],["g","h","i"]]
```

¿Para qué sirve trabajar con Archivos?

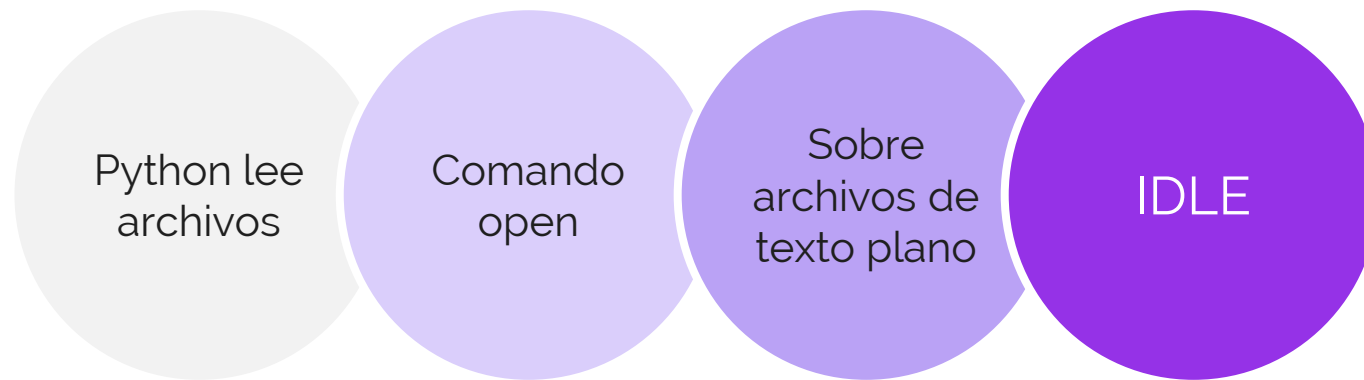
Crear o escribir sobre archivos ya existentes.

Guardar el procesamiento que hacemos sobre un archivo, que ya leímos.

Guardar lo que pasó en la ejecución de un código. Dado que, cuando la ejecución de un código se termina, todo lo que ocurrió en ésta se pierde.

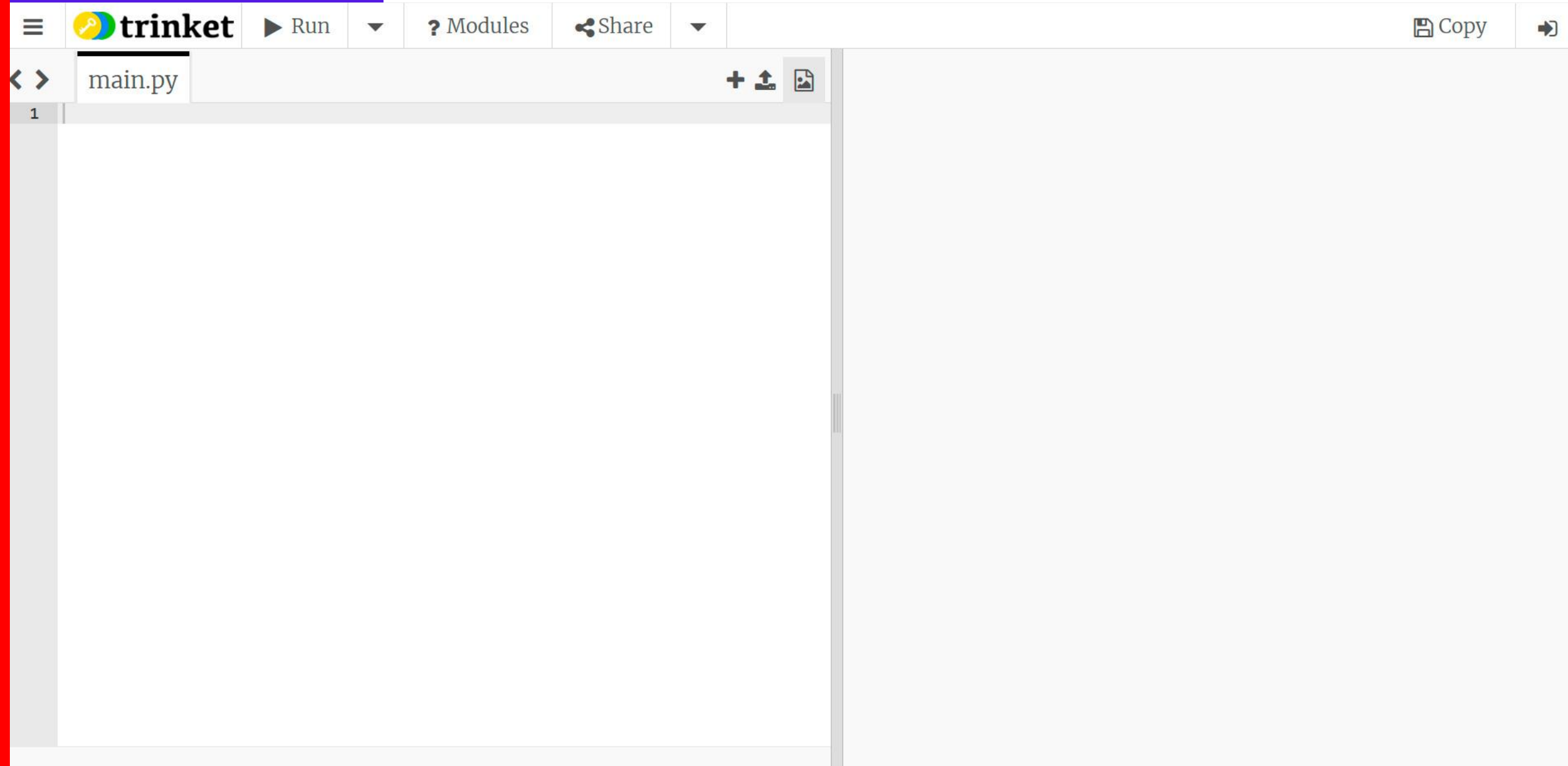
Leer Archivos

Al leer archivos, podemos ingresar mucha información de forma muy rápida y eficiente.



Esto para que ustedes lo puedan ocupar de forma autónoma y sin necesidad de una consola online.

¿Qué es IDLE?



Veamos un ejemplo de lectura de archivos

CÓDIGO	RESULTADO
<pre>informacion_archivo = open("ejemplo1.txt") print(informacion_archivo)</pre>	<pre><_io.TextIOWrapper name='ejemplo1.txt' mode='r' encoding='cp1252'></pre>

Podemos ver que Python leyó el archivo. No obstante, aún no podemos acceder a su contenido.

Trabaja con **ejemplo1.txt**

Leer Archivos

Recuerda revisar la
Ruta de ejercicios.
Ejercicio EM3-05 →



```
1 informacion_archivo = open("ejemplo1.txt")
  print(informacion_archivo.read())
```

Ocupamos la función `read()` sobre la variable que representa al archivo. Nos permite acceder a todo el contenido del archivo.

```
Hola,
bienvenido
al
curso
Herramientas de programación en Python para procesamiento de datos.
```

Notemos que la palabra con tilde no fue correctamente leída por Python.

Para acceder al contenido de un archivo, debemos hacer lo siguiente.

Leer Archivos

Para arreglar este error podemos hacer lo siguiente:

CÓDIGO	RESULTADO
<pre>informacion_archivo = open("ejemplo1.txt", encoding ="UTF-8") print(info ad())</pre>	<pre>Hola, bienvenido al curso Herramientas de</pre> <p>Si tenemos problemas para leer vocales con tildes u otros caracteres, podemos ingresar "encoding="UTF-8"" como segundo parámetro a la función open.</p>

Leer Archivos

¿Y si tuviéramos un archivo con miles y miles de líneas?

Tener una sola variable con esta información no es muy útil. Por eso, la mejor forma de leer un archivo es de la siguiente manera:

Como vimos anteriormente, la variable `informacion_archivo` contiene al archivo que abrimos. Luego, en esta variable ejecutamos la función `readlines()`.

Esta función lo que hace es retornar una lista cuyos elementos son las líneas del archivo. Esta lista la almacenamos en la variable `líneas`.

```
informacion_archivo =  
open("ejemplo1.txt",  
      "r")
```

```
líneas = informacion_archivo.readlines()
```

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

al

curso

Herramientas de programación en Python para
procesamiento de datos.

Leer Archivos

Trabajar con archivos, de la forma que mostramos anteriormente es muy útil porque:

- 1 El archivo leído se transforma en una lista cuyos elementos son las líneas del archivo, esto nos permite saber cuántas líneas tiene el archivo.
- 2 Permite hacer un `for` sobre la lista de líneas del archivo, para poder acceder a cada una de ellas y trabajar de forma independiente con cada línea del archivo.

Escribir Archivos

Para guardar el valor de una o más variables en archivos durante la ejecución de nuestro código, es fundamental entender a cabalidad el comando `open()`.

Lo que hace `open`, es abrir un archivo para poder trabajar con él, a través de 3 maneras.

r

w

a

Escribir Archivos

r

Viene de "read" (leer en inglés). Este modo sirve para leer un archivo.

w

Viene de "write" (escribir en inglés). Este modo sirve para escribir en un archivo.

IMPORTANTE: Al elegir este modo al abrir un archivo, se borra el contenido que éste tenía.

a

Viene de "append" (agregar en inglés). Este modo sirve para escribir en un archivo. Lo que se escribe, se agrega al contenido original del archivo.

Escribir Archivos

Por lo tanto, cuando ocupamos el comando `open ()` en los ejemplos anteriores, en realidad estábamos ocupando el modo "r". De hecho, este modo viene seleccionado por defecto.

Veamos el ejemplo anterior ocupando explícitamente el modo "r".

```
open (  
    r
```

Escribir Archivos

CÓDIGO	RESULTADO
<pre>informacion_archivo = open('datos.txt', 'r') linea = informacion_archivo.readline() for linea in informacion_archivo: print(linea)</pre>	<pre>Hola, bienvenido al curso Herramientas de programación en Python para procesamiento de datos.</pre>

Podemos observar que el resultado es el mismo que en el ejemplo anterior, lo que confirma que el modo por defecto del comando `open()` es "r".

Escribir Archivos

Es importante, incorporar el uso de una nueva función al ocupar el comando `open ()`.

Esta función es `close ()`

Lo que hace es terminar el uso de un archivo en la ejecución de nuestro código.



Es relevante usar este comando siempre al terminar de ocupar un archivo en nuestro código, ya que evita posibles errores.

Escribir Archivos

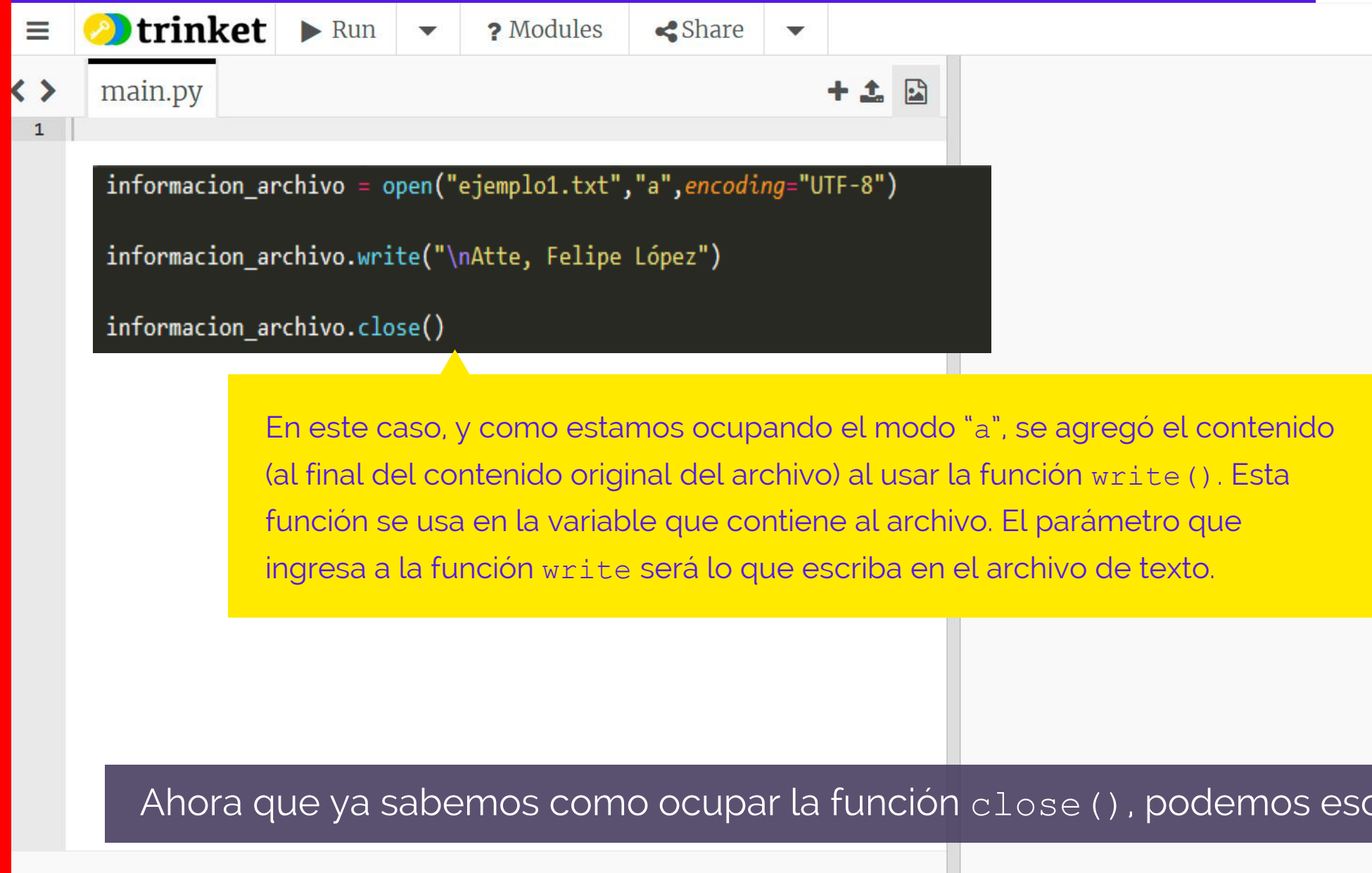
Veamos un ejemplo sobre `close()`

CÓDIGO	RESULTADO
<pre>informacion_archivo = open("ejemplo1.txt", "r", encoding ="UTF-8") lineas = informacion_archivo.readlines() informacion_archivo.close() for linea in lineas: print(linea)</pre>	<p>Después de usar la función <code>readlines()</code>, y crear la lista con las líneas del archivo, podemos ocupar la función <code>close()</code>. El uso de esta función no afecta a la ejecución del código (como vemos en el output en la consola). Lo importante es siempre recordar el uso de <code>close()</code>, especialmente cuando posteriormente queramos escribir en algún archivo.</p>

Veamos cómo ocupar el modo "a" del comando `open()`

Recuerda revisar la Ruta de ejercicios.

Ejercicio EM3-10 →



The screenshot shows the Trinket IDE interface. The top bar includes the Trinket logo, a 'Run' button, a 'Modules' dropdown, and a 'Share' button. Below the bar, the file 'main.py' is open. The code in the editor is as follows:

```
1 informacion_archivo = open("ejemplo1.txt","a",encoding="UTF-8")  
  
informacion_archivo.write("\nAtte, Felipe López")  
  
informacion_archivo.close()
```

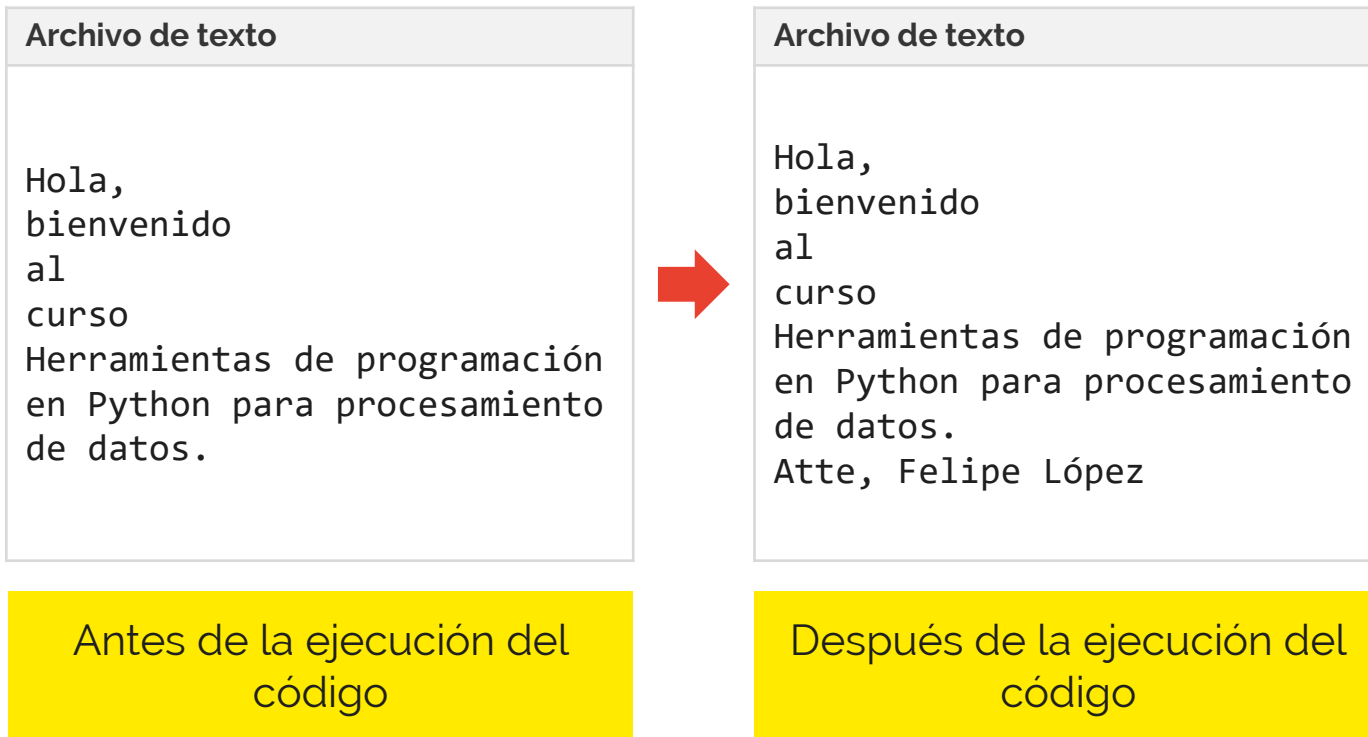
A yellow callout box points to the `write()` method in the code, containing the following text:

En este caso, y como estamos ocupando el modo "a", se agregó el contenido (al final del contenido original del archivo) al usar la función `write()`. Esta función se usa en la variable que contiene al archivo. El parámetro que ingresa a la función `write` será lo que escriba en el archivo de texto.

A dark blue footer bar at the bottom of the image contains the text:

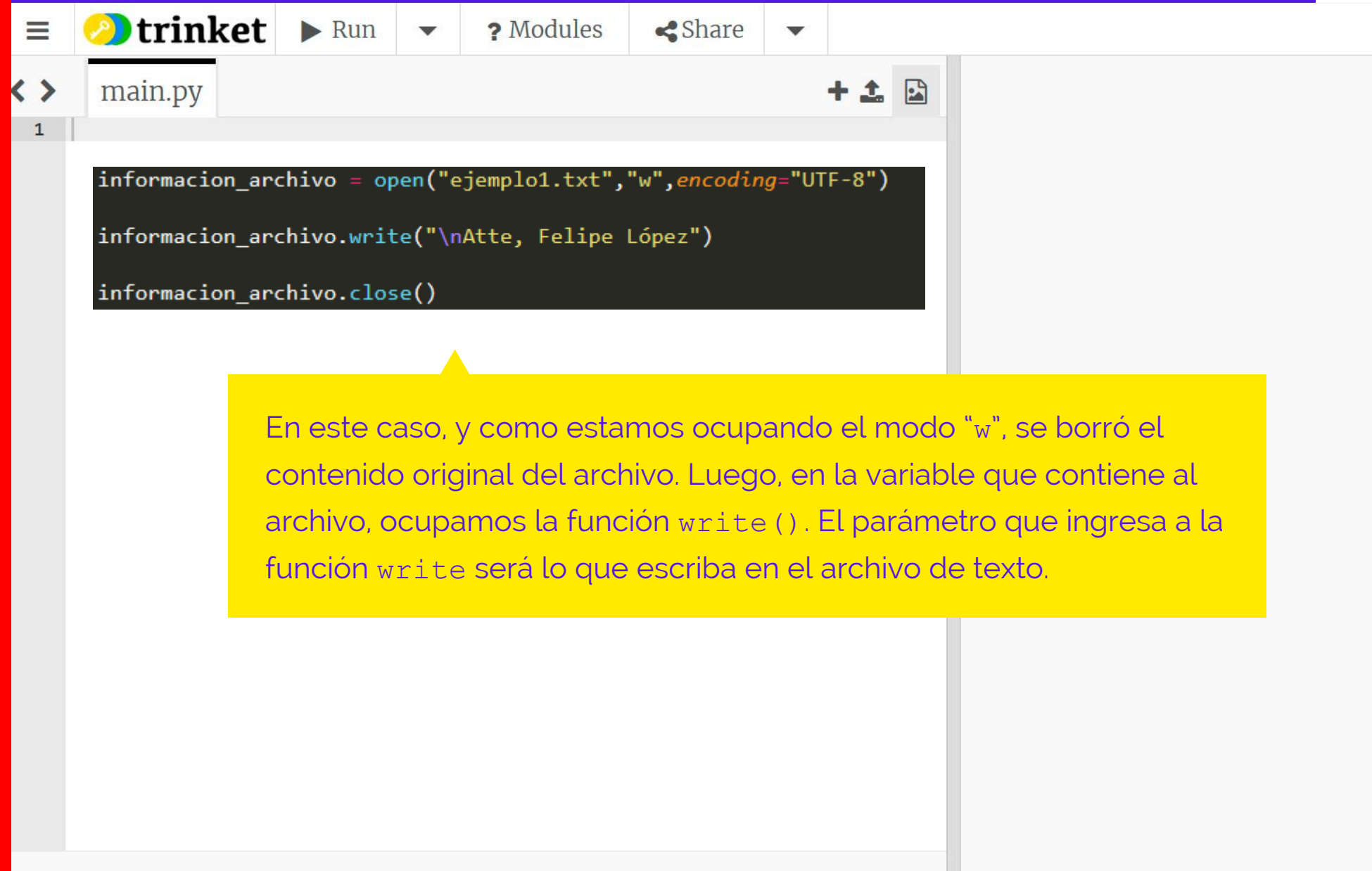
Ahora que ya sabemos como ocupar la función `close()`, podemos escribir en archivos.

Veamos cómo ocupar el modo “a” del comando `open()`



Veamos cómo ocupar el modo "w" del comando `open()`

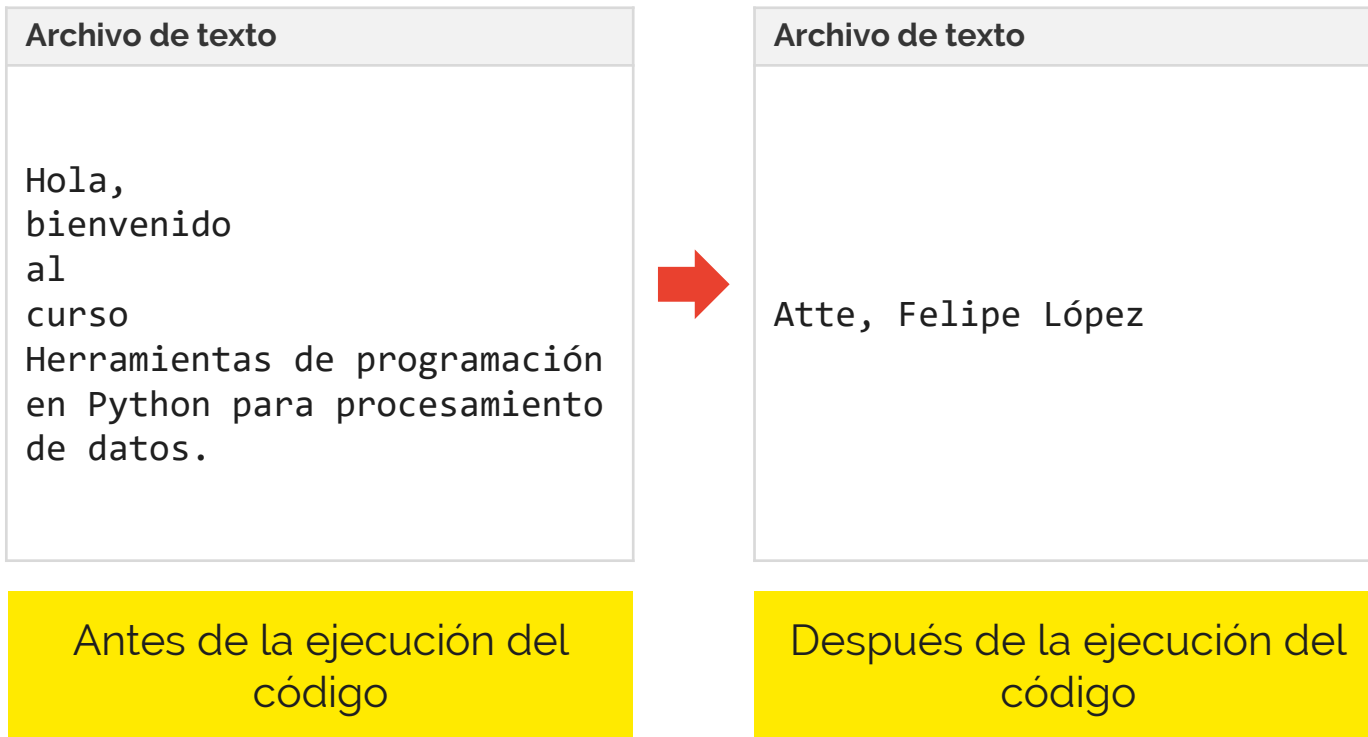
Recuerda revisar la Ruta de ejercicios.
Ejercicio EM3-11 →



```
main.py
1
informacion_archivo = open("ejemplo1.txt","w",encoding="UTF-8")
informacion_archivo.write("\nAtte, Felipe López")
informacion_archivo.close()
```

En este caso, y como estamos ocupando el modo "w", se borró el contenido original del archivo. Luego, en la variable que contiene al archivo, ocupamos la función `write()`. El parámetro que ingresa a la función `write` será lo que escriba en el archivo de texto.

Veamos cómo ocupar el modo “w” del comando `open()`



Escribir Archivos

¿Qué pasa si ocupamos el comando `open()` en un archivo que no existe o no está creado?

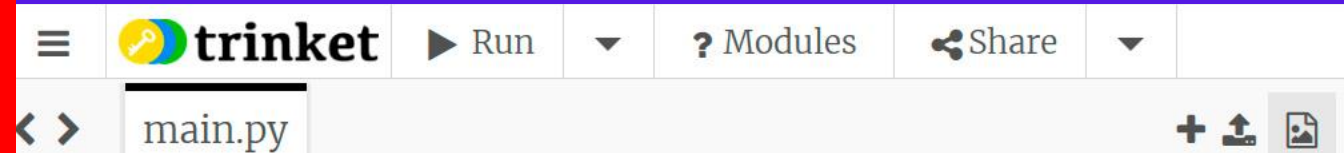
¿Cómo podríamos hacerlo?

Si estamos ocupando el modo "w" o "a", Python crea automáticamente el archivo

Si ocupamos el modo "r" (o no definimos el modo anteriormente), entonces Python buscará el archivo, y al no encontrarlo, arrojará un error.

Revisemos ¿Cómo podríamos hacerlo?

Recuerda revisar la
Ruta de ejercicios.
Ejercicio EM3-12 →



```
informacion_archivo = open("ejemplo_nuevo.txt","w",encoding="UTF-8")  
  
informacion_archivo.write("Mi nombre es:")  
  
informacion_archivo.write("Felipe López")  
  
informacion_archivo.close()
```

Se observa, que pesar de haber ocupado la función `write` dos veces, ambos textos se escribieron juntos. Si quisiéramos que fueran por separado en el archivo de texto, tendríamos que haber ocupado el carácter `"\n"`, que indica un "salto de línea", es decir, que dos textos se separen por líneas distintas.

CAPTURA DE PANTALLA

Archivo creado

```
Mi nombre es:Felipe López
```



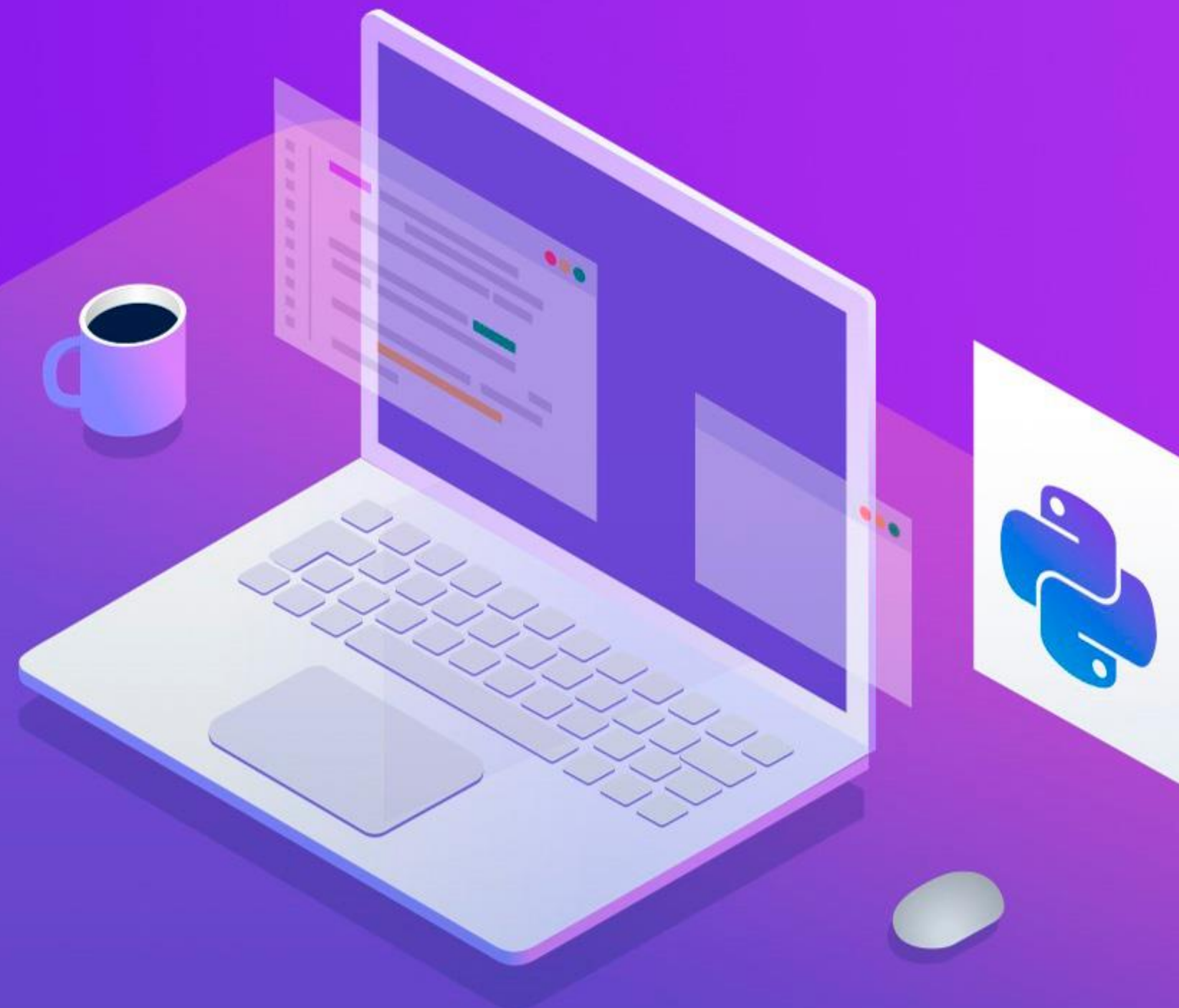

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

ARCHIVOS Y

PROCESAMIENTO DE

DATOS

>>> Parte 2: Procesamiento de Datos

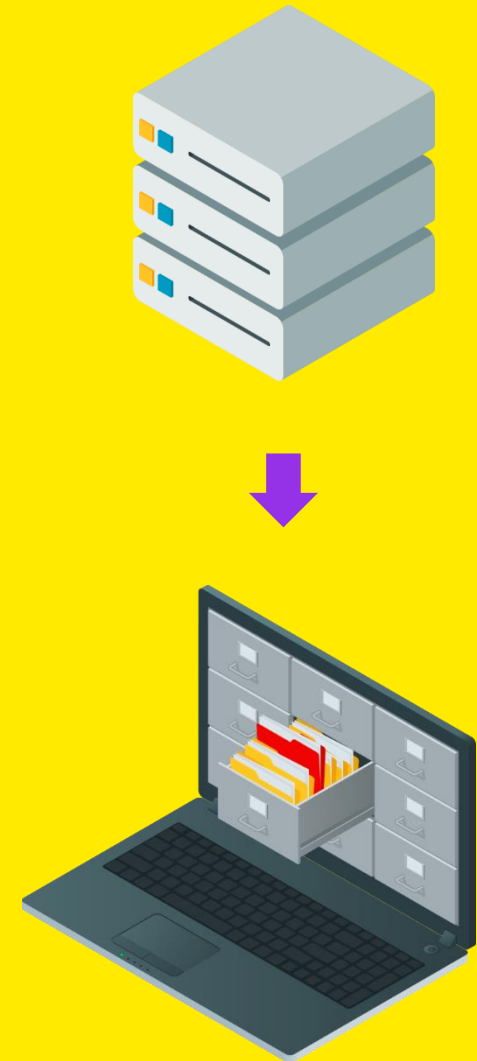


Procesamiento de datos

¿Cómo se aplica esto en la vida laboral?

Recordemos que la mayoría de los datos se almacenan en bases de datos. Lo ideal sería aprender una forma de poder traspasar la información de una base de datos a Python.

Una forma muy simple de hacerlo es mediante archivos CSV.



Archivos CSV

Los archivos CSV son archivos de texto plano que replican una matriz.

Para ejemplificar un archivo CSV, volvamos a nuestra base de datos "**Empleados**".

Ésta tenía las siguientes características:

- Nombre
- Edad
- Fecha de Nacimiento
- RUT



Archivos CSV

A continuación, un ejemplo de la base de datos:

Nombre	Edad	Fecha de Nacimiento	RUT
Juan Pérez	27	31-01-1991	17.587.451-8
María Rojas	54	04-05-1964	9.475.362-4
Pedro Rodríguez	35	18-06-1983	13.748.645-2
Soledad Ríos	21	03-03-1997	20.471.472-1

Archivos CSV

En formato CSV se vería de la siguiente manera:

CSV

```
Nombre;Edad;Fecha de Nacimiento;RUT
Juan Pérez;27;31-01-1991;17.587.451-8
María Rojas;54;04-05-1964;9.475.362-4
Pedro Rodríguez;35;18-06-1983;13.748.645-2
Soledad Ríos;21;03-03-1997;20.471.472-1
```

El CSV es una representación de una tabla (base de datos).

Archivos CSV

En formato CSV se vería de la siguiente manera:

CSV

```
Nombre;Edad;Fecha de Nacimiento;RUT
Juan Pérez;27;31-01-1991;17.587.451-8
María Rojas;54;04-05-1964;9.475.362-4
Pedro Rodríguez;35;18-06-1983;13.748.645-2
Soledad Ríos;21;03-03-1997;20.471.472-1
```

EL CSV

La primera línea del texto corresponde a la primera fila de la tabla, y en general representa a los encabezados o bien los nombres de cada columna.

IMPORTANTE: No todos los archivos CSV tienen esta línea.

Archivos CSV

En formato CSV se vería de la siguiente manera:

CSV			
Nombre;Edad;Fecha de Nacimiento;RUT			
Juan Pérez;27;31-01-1991;17.587.451-8			
María Rojas;54;04-05-1964;9.475.362-4			
Pedro Rodríguez;35;18-06-1983;13.748.645-2			
Soledad Ríos;21;03-03-1997;20.471.472-1			

El CSV

Cada línea de un archivo CSV representa a una fila de la tabla.

s).

Archivos CSV

En formato CSV se vería de la siguiente manera:

CSV

```
Nombre;Edad;Fecha de Nacimiento;RUT
Juan Pérez;27;31-01-1991;17.587.451-8
María Rojas;54;04-05-1964;9.475.362-4
Pedro Rodríguez;35;18-06-1983;13.748.645-2
Soledad Ríos;21;03-03-1997;20.471.472-1
```

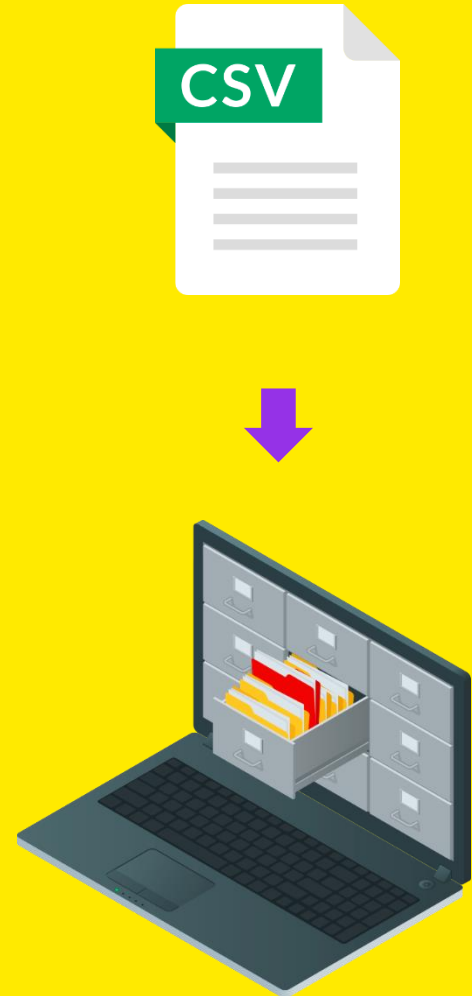
El CSV

Dentro de cada línea del archivo, se separan las columnas con el carácter ";" o ",".

Carga Masiva de Datos

¿Cómo podemos cargar un archivo CSV en Python?

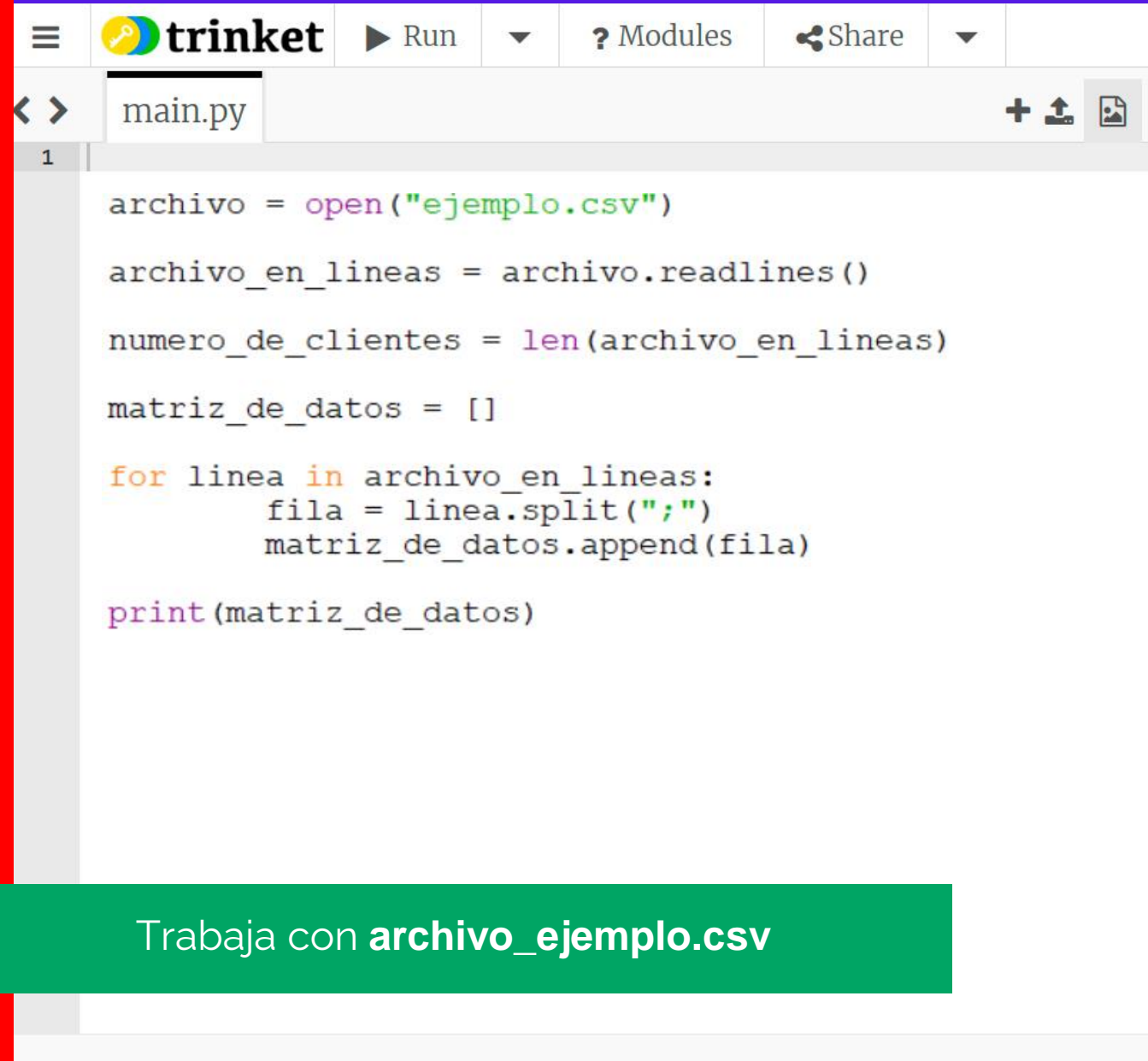
Lo que queremos, es transformar el CSV a una lista de listas, que nos permita trabajar con estos datos.



¿Cómo podemos cargar un archivo CSV en Python?

Recuerda revisar la
Ruta de ejercicios.

Ejercicio EM3-13 →



The image shows a screenshot of the Trinket Python IDE interface. At the top, there is a navigation bar with a hamburger menu, the Trinket logo, a 'Run' button, a dropdown arrow, a '? Modules' button, and a 'Share' button. Below this, the file 'main.py' is open in the editor. The code in the editor is as follows:

```
1
archivo = open("ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

At the bottom of the image, there is a green banner with the text: Trabaja con **archivo_ejemplo.csv**

¿Cómo podemos cargar un archivo CSV en Python?

Con la ejecución de este código, se hace la siguiente transformación:

Archivo de datos ;FECHA_NAC;TIPO_CLIENTE;ULTIMA_COMPRA;ULTIMA_SUCURSAL

0;7.671.557-8;Rodrigo Pablo Saavedra López;56;1962/12/6;D;1999/8/7;Sucursal3
1;16.034.848-10;Cecilia Blanca Saavedra González;50;1968/12/10;C;2004/2/13;Sucursal4
2;5.366.371-6;Isabel Daniela Valenzuela González;47;1971/11/23;B;1993/4/19;Sucursal2
3;5.269.662-6;Andrés Javier Vergara Castro;19;1999/4/29;B;1995/11/3;Sucursal6
4;18.515.856-4;Francisco Juan Robles Robles;18;2000/4/27;D;2007/12/8;Sucursal2
5;18.482.681-9;Andrea Javiera Campos Rodríguez;50;1968/7/16;B;1993/12/18;Sucursal4
6;21.678.438-10;Daniela Ignacia Quiroga Valenzuela;30;1988/5/1;E;1999/7/26;Sucursal1
7;14.727.677-10;Juan Felipe Ruiz González;27;1991/1/8;A;1992/4/27;Sucursal5
8;8.981.662-2;Blanca Daniela Quiroga Castro;56;1962/7/19;C;2016/3/26;Sucursal6
9;12.579.550-5;Pablo Javier Robles Ruiz;50;1968/4/5;C;1997/11/2;Sucursal6
10;14.504.267-8;Cecilia Daniela González Ruiz;30;1988/1/15;C;1992/9/19;Sucursal4
11;6.655.760-5;Alejandro Rodrigo Vergara Muñoz;57;1961/10/2;B;2014/6/22;Sucursal5
12;17.515.538-10;Rodrigo Rodrigo Rodríguez Castro;51;1967/9/24;C;2001/3/6;Sucursal2
13;11.903.844-0;Ignacio Pablo Muñoz Vergara;18;2000/9/14;B;2011/11/8;Sucursal5
14;7.572.421-6;María Victoria Rodríguez Campos;35;1983/3/26;E;2006/5/27;Sucursal1
15;4.285.169-3;Javiera Victoria Saavedra Rodríguez;31;1987/2/19;A;1998/5/0;Sucursal2
16;4.269.244-6;Felipe Alejandro Saavedra Rojas;32;1986/3/2;A;2008/9/15;Sucursal1
17;7.255.527-2;Victoria Ignacia Rodríguez Rodríguez;39;1979/9/12;E;2011/1/11;Sucursal3
18;4.523.302-7;Cecilia Andrea López López;38;1980/2/9;A;1990/2/22;Sucursal3
19;5.791.474-0;Felipe Vicente Quiroga Quiroga;45;1973/12/1;A;2017/5/30;Sucursal6
20;19.330.578-4;Alejandro Alejandro Marín González;31;1987/1/25;D;2017/2/23;Sucursal4
21;15.559.677-1;Isabel Valeria López Ruiz;31;1987/6/2;B;2002/2/28;Sucursal5
22;14.308.988-3;Constanza Cecilia López Rodríguez;50;1968/6/23;B;2003/5/30;Sucursal4
23;4.590.623-7;Francisco Juan Rojas Marín;40;1978/7/9;C;2017/6/1;Sucursal6
24;9.110.736-3;Constanza Paula Vergara Díaz;27;1991/7/18;B;1995/3/18;Sucursal3

Matriz de datos	NOMBRE',	'EDAD',	'FECHA_NAC',	'TIPO_CLIENTE',	'ULTIMA_COMPRA',	'ID'
	igo Pablo Saavedra López',	'56',	'1962/12/6',	'D',	'1999/8/7',	'Su
	8-10',	'Cecilia Blanca Saavedra González',	'50',	'1968/12/10',	'C',	'2004/2/13',
	71-6',	'Isabel Daniela Valenzuela González',	'47',	'1971/11/23',	'B',	'1993/4/19',
	.662-6',	'Andrés Javier Vergara Castro',	'19',	'1999/4/29',	'B',	'1995/11/3',
	-4',	'Francisco Juan Robles Robles',	'18',	'2000/4/27',	'D',	'2007/12/8',
	'Andrea Javiera Campos Rodríguez',	'50',	'1968/7/16',	'B',	'1993/12/18',	'Sucursal
	'Daniela Ignacia Quiroga Valenzuela',	'30',	'1988/5/1',	'E',	'1999/7/26',	'Sucursa
	, 'Juan Felipe Ruiz González',	'27',	'1991/1/8',	'A',	'1992/4/27',	'Sucursal5\n']
	Daniela Quiroga Castro',	'56',	'1962/7/19',	'C',	'2016/3/26',	'Sucursal6\n']
	er Robles Ruiz',	'50',	'1968/4/5',	'C',	'1997/11/2',	'Sucursal6\n']
	nzález Ruiz',	'30',	'1988/1/15',	'C',	'1992/9/19',	'Sucursal4\n']
	gara Muñoz',	'57',	'1961/10/2',	'B',	'2014/6/22',	'Sucursal5\n']
	íguez Castro',	'51',	'1967/9/24',	'C',	'2001/3/6',	'Sucursal2\n']
	Vergara',	'18',	'2000/9/14',	'B',	'2011/11/8',	'Sucursal5\n']
	Campos',	'35',	'1983/3/26',	'E',	'2006/5/27',	'Sucursal1\n']
	Rodríguez',	'31',	'1987/2/19',	'A',	'1998/5/0',	'Sucursal2\n']
	a Rojas',	'32',	'1986/3/2',	'A',	'2008/9/15',	'Sucursal1\n']
	Rodríguez',	'39',	'1979/9/12',	'E',	'2011/1/11',	'Sucursal3\n']
	pez',	'38',	'1980/2/9',	'A',	'1990/2/22',	'Sucursal3\n']
	',	'45',	'1973/12/1',	'A',	'2017/5/30',	'Sucursal6\n']
	lez',	'31',	'1987/1/25',	'D',	'2017/2/23',	'Sucursal4\n']
	'31',	'1987/6/2',	'B',	'2002/2/28',	'Sucursal5\n']	['22', '14.308.988-3', 'Consta
	'50',	'1968/6/23',	'B',	'2003/5/30',	'Sucursal4\n']	['23', '4.590.623-7', 'Franci
	'1978/7/9',	'C',	'2017/6/1',	'Sucursal6\n']	['24', '9.110.736-3', 'Constanza Paul	
	7/18',	'B',	'1995/3/18',	'Sucursal3\n']	['25', '7.335.366-0', 'Vicente Gabriel Ca	
	',	'B',	'1998/2/26',	'Sucursal5\n']	['26', '16.379.549-0', 'Andrés Andrés Rojas F	
	'2016/2/17',	'Sucursal4\n']	['27', '14.979.405-3', 'Francisco Andrés González Rod			
	'A',	'2000/8/25',	'Sucursal3\n']	['28', '9.612.917-2', 'Javiera Pamela López Rod		

¿Cómo podemos cargar un archivo CSV en Python?

Analicemos el código que ocupamos para transformar el archivo CSV en una matriz de datos.

CÓDIGO

```
archivo = open("archivo_ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

¿Cómo podemos cargar un archivo CSV en Python?

Analicemos el código que ocupamos para transformar el archivo CSV en una matriz de datos.

CÓDIGO

```
archivo = open("archivo_ejemplo.csv")
archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

Como mostramos anteriormente, leímos el archivo "ejemplo.csv" y lo cargamos a la variable `archivo`. Luego, y mediante la función `readlines()`, creamos una lista que contiene todas las líneas del archivo. Luego, creamos una lista vacía que se llama `matriz_de_datos`

¿Cómo podemos cargar un archivo CSV en Python?

Analicemos el código que ocupamos para transformar el archivo CSV en una matriz de datos.

CÓDIGO

```
archivo = open("archivo_ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

Luego, y mediante un `for` recorremos todas los elementos de la variable `archivo_en_líneas`. Cada elemento es una línea del archivo. En cada una de la líneas, hacemos un `split(";")`. Lo que hace esto es separar cada línea mediante el carácter `" ; "`.

¿Cómo podemos cargar un archivo CSV en Python?

Analicemos el código que ocupamos para transformar el archivo CSV en una matriz de datos.

CÓDIGO

```
archivo = open("archivo_ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

Recordemos que en archivos CSV las columnas se separan mediante un ";" o ",". Por lo tanto, generamos una nueva lista en esta línea donde sus elementos serán cada uno de los elementos de esa fila de la matriz original separados en columnas.

¿Cómo podemos cargar un archivo CSV en Python?

Analicemos el código que ocupamos para transformar el archivo CSV en una matriz de datos.

CÓDIGO

```
archivo = open("archivo_ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)
```

Finalmente, agregamos esta lista a nuestra matriz de datos y la variable `matriz_de_datos` será una fiel representación de la base de datos que estaba guardada originalmente en el archivo CSV.

Carga Masiva de Datos

Lo importante, es que ahora los datos están en una matriz que nos permite acceder de forma más fácil a ellos.

¿Cómo podemos saber cuántos clientes tenemos en nuestra base de datos?



Mediante `len(matriz_de_datos)`, en este caso se almacena en la variable `numero_de_clientes`.

¿Cómo podemos saber el RUT del cliente con ID (identificador) igual a 5?



Los clientes están ordenados, sabemos que el cliente con `ID=5`, está en la sexta fila (recordemos que se empieza a contar desde 0). Además, sabemos que el RUT está en la segunda columna.



Por lo tanto, `matriz_de_datos[5][1]` me dará el RUT del cliente con `ID=5`.

Edición masiva de datos

Ahora que los datos están en una matriz, es mucho más fácil editarlos.

Veamos un ejemplo de cómo poder editar datos de forma masiva:

CSV

```
[['ID', 'RUT', 'NOMBRE', 'EDAD', 'FECHA_NAC', 'TIPO_CLIENTE', 'ULTIMA_COMPRA', 'ULTIMA_SUCURSAL\n']  
[0, '7.671.537-8', 'Rodrigo Pablo Saavedra López', '56', '1962/12/6', 'D', '1999/8/7', 'Sucursal3\n']  
[1, '16.034.848-10', 'Cecilia Blanca Saavedra González', '50', '1968/12/10', 'C', '2004/2/13',  
'Sucursal4\n'],
```

1° Podemos observar en la `matriz_de_datos` que acabamos de cargar, que la última columna tiene un `'\n'`

Edición masiva de datos

Recuerda revisar la
Ruta de ejercicios.

Ejercicio EM3-14 →



The screenshot shows the Trinket IDE interface. At the top, there is a navigation bar with a menu icon, the 'trinket' logo, a 'Run' button, a dropdown arrow, a 'Modules' button with a question mark, and a 'Share' button with a dropdown arrow. Below this is a tab labeled 'main.py' with navigation arrows on the left and icons for adding files, uploading, and saving on the right. The main area is a dark-themed code editor with the following Python code:

```
1
archivo = open("ejemplo.csv")

archivo_en_lineas = archivo.readlines()

numero_de_clientes = len(archivo_en_lineas)

matriz_de_datos = []

for linea in archivo_en_lineas:
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)

for fila in matriz_de_datos:
    fila[7]=fila[7].strip()

print(matriz_de_datos)
```

Podemos editar de la siguiente manera.

Ejercicio EM3-14 →



▶ Run

? Modules

 Share

main.py

Veamos el cambio que produjo esta edición. Esta era la matriz original de datos.

```
[['ID', 'RUT', 'NOMBRE', 'EDAD', 'FECHA_NAC', 'TIPO_CLIENTE', 'ULTIMA_COMPRA', 'ULTIMA_SUCURSAL\n'], ['0', '7.671.537-8', 'Rodrigo Pablo Saavedra López', '56', '1962/12/6', 'D', '1999/8/7', 'Sucursal3\n'], ['1', '16.034.848-10', 'Cecilia Blanca Saavedra González', '50', '1968/12/10', 'C', '2004/2/13', 'Sucursal4\n'], ['2', '5.366.371-6', 'Isabel Daniela Valenzuela González', '47', '1971/11/23', 'B', '1993/4/19', 'Sucursal2\n'], ['3', '5.269
```

Y esta es la modificada.

```
[['ID', 'RUT', 'NOMBRE', 'EDAD', 'FECHA_NAC', 'TIPO_CLIENTE', 'ULTIMA_COMPRA', 'ULTIMA_SUCURSAL'], ['0', '7.671.537-8', 'Rodrigo Pablo Saavedra López', '56', '1962/12/6', 'D', '1999/8/7', 'Sucursal3'], ['1', '16.034.848-10', 'Cecilia Blanca Saavedra González', '50', '1968/12/10', 'C', '2004/2/13', 'Sucursal4'], ['2', '5.366.371-6', 'Isabel Daniela Valenzuela González', '47', '1971/11/23', 'B', '1993/4/19', 'Sucursal2'], ['3', '5.269.662-6',
```

Podemos observar que el '\n' efectivamente fue removido.

Edición masiva de datos

CÓDIGO

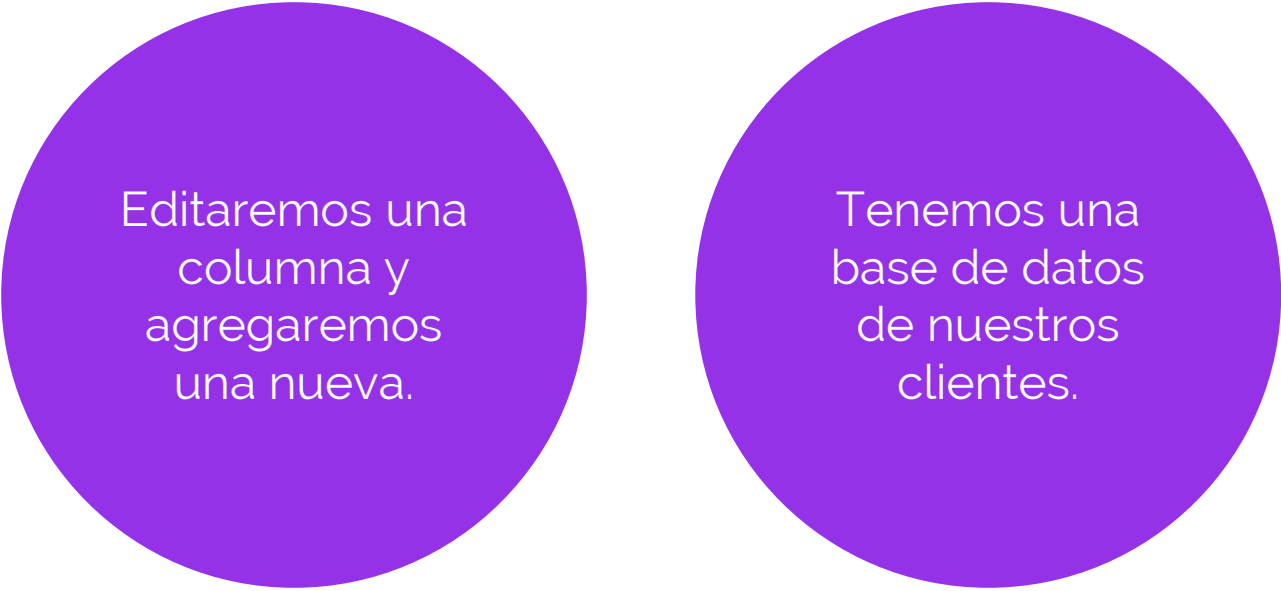
```
for fila in matriz_de_datos:
    fila[7]=fila[7].strip()

print(matriz_de_datos)
```

En realidad, lo que hicimos en el código que mostramos anteriormente es generalizable para cualquier tipo de edición masiva. Lo importante es hacer un `for` que recorra cada una de las líneas del archivo, y luego editar la columna que lo requiera. Esta edición la mayoría de las veces se hará mediante funciones de `strings`.

Ejemplos prácticos

Veamos un ejemplo práctico de carga de datos:

Two purple circles are positioned side-by-side. The left circle contains the text 'Editaremos una columna y agregaremos una nueva.' and the right circle contains the text 'Tenemos una base de datos de nuestros clientes.'

Editaremos una
columna y
agregaremos
una nueva.

Tenemos una
base de datos
de nuestros
clientes.

Ejemplos prácticos

Los datos que tenemos son:

- **ID**: identificador de cada cliente.
- **RUT**: el RUT de cada cliente.
- **Nombre**: el nombre completo (dos nombres y dos apellidos) de cada cliente.
- **Fecha de nacimiento**: la fecha de nacimiento de cada cliente.
- **Tipo de cliente**: hay 5 tipos de clientes.
- Los tipos son **A,B,C,D** y **E**.

Ejemplos prácticos



The screenshot shows the Trinket IDE interface. The top bar contains a menu icon, the Trinket logo, a 'Run' button, a dropdown arrow, a '? Modules' button, a 'Share' button, and another dropdown arrow. The right bar contains a 'Copy' button and a share icon. The main area shows a file named 'main.py' with 6 lines of CSV data. The data is as follows:

ID	Score	Name	Birth Date	Category
0	14.782.991-7	Juan Francisco González Ruiz	1980/2/26	#!#A
1	4.593.362-3	Constanza Daniela Díaz Marín	1968/6/9	#!#C
2	19.816.353-5	Pedro Rodrigo Rodríguez Muñoz	1984/7/17	#!#D
3	16.804.654-9	Ignacio Bernardo Rodríguez Muñoz	1990/8/6	#!#D
4	19.566.578-10	Alejandro Bernardo Campos Campos	1977/8/14	#!#B
5	4.208.814-9	Victoria Cecilia Valenzuela Rodríguez	1976/7/13	#!#E
6	10.903.675-7	Cecilia Andrea Rodríguez Robles	1979/2/6	#!#E

Veamos algunas filas de este archivo CSV para saber cómo es.

Ejemplos prácticos

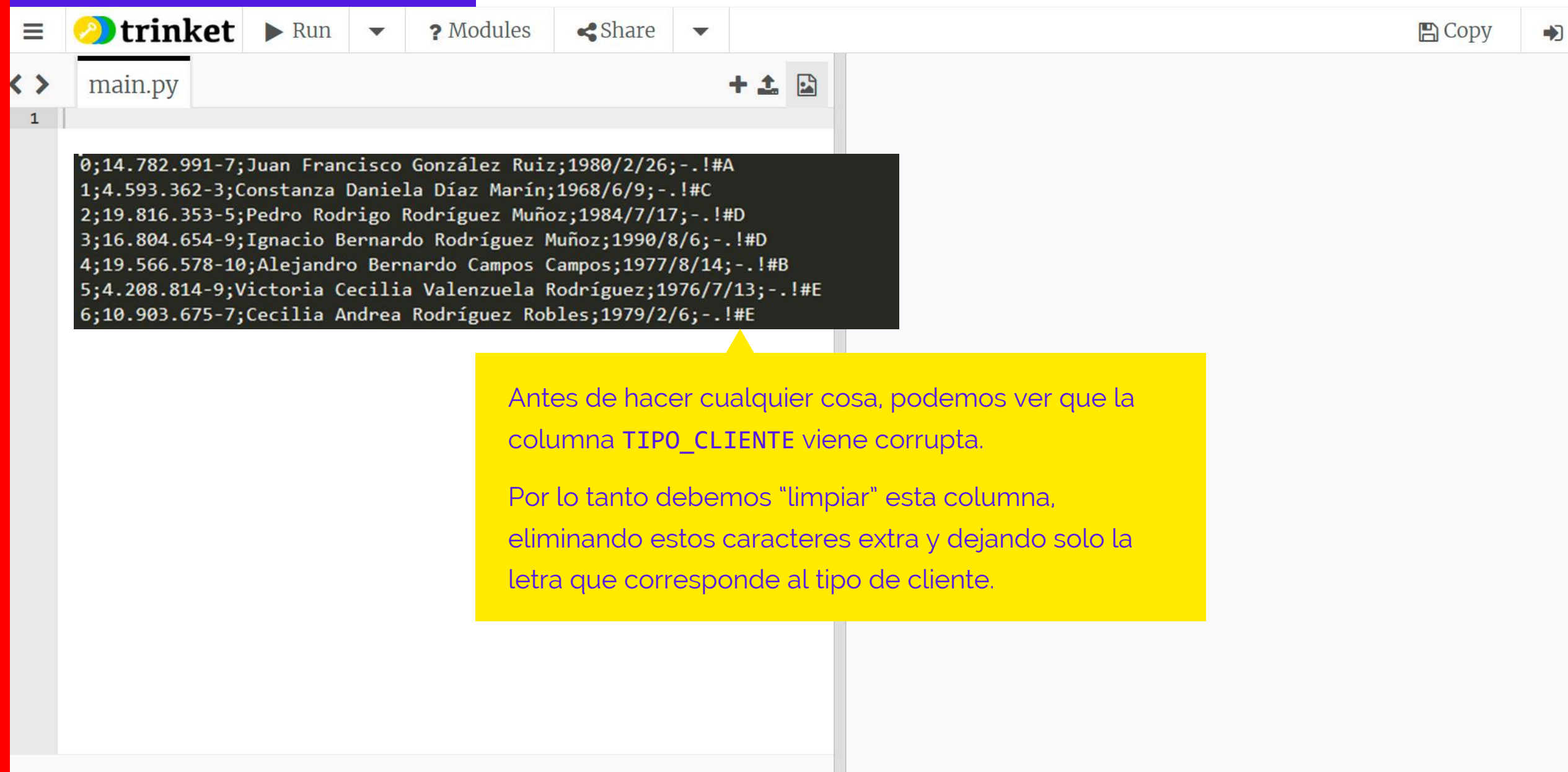


The screenshot shows the Trinket IDE interface. At the top, there is a navigation bar with a menu icon, the Trinket logo, a 'Run' button, a dropdown arrow, a '? Modules' button, a 'Share' button, another dropdown arrow, a 'Copy' button with a document icon, and a share icon. Below the navigation bar, the file name 'main.py' is displayed. The main editor area shows a single line of code (line 1) containing a CSV-like string with 6 rows of data. The data is as follows:

	0	1	2	3	4	5	6
0	14.782.991-7	Juan Francisco González Ruiz	1980/2/26	-	!	#	A
1	4.593.362-3	Constanza Daniela Díaz Marín	1968/6/9	-	!	#	C
2	19.816.353-5	Pedro Rodrigo Rodríguez Muñoz	1984/7/17	-	!	#	D
3	16.804.654-9	Ignacio Bernardo Rodríguez Muñoz	1990/8/6	-	!	#	D
4	19.566.578-10	Alejandro Bernardo Campos Campos	1977/8/14	-	!	#	B
5	4.208.814-9	Victoria Cecilia Valenzuela Rodríguez	1976/7/13	-	!	#	E
6	10.903.675-7	Cecilia Andrea Rodríguez Robles	1979/2/6	-	!	#	E

Notemos, que no viene con una primera fila con el nombre de cada columna. Esto es un caso muy normal, y es importante que sepamos cómo manejarlo.

Ejemplos prácticos



The screenshot shows the Trinket online code editor interface. The top bar includes a menu icon, the Trinket logo, and buttons for 'Run', 'Modules', 'Share', 'Copy', and a share icon. The file name 'main.py' is displayed in the editor's title bar. The code area contains a Python script with six lines of data, each representing a client record. The records are separated by semicolons and end with a client type identifier. The second record, 'Constanza Daniela Díaz Marín', has a corrupted client type field containing 'C' followed by a space and a hash symbol. A yellow callout box points to this field with a purple arrow.

```
0;14.782.991-7;Juan Francisco González Ruiz;1980/2/26;-.!#A
1;4.593.362-3;Constanza Daniela Díaz Marín;1968/6/9;-.!#C
2;19.816.353-5;Pedro Rodrigo Rodríguez Muñoz;1984/7/17;-.!#D
3;16.804.654-9;Ignacio Bernardo Rodríguez Muñoz;1990/8/6;-.!#D
4;19.566.578-10;Alejandro Bernardo Campos Campos;1977/8/14;-.!#B
5;4.208.814-9;Victoria Cecilia Valenzuela Rodríguez;1976/7/13;-.!#E
6;10.903.675-7;Cecilia Andrea Rodríguez Robles;1979/2/6;-.!#E
```

Antes de hacer cualquier cosa, podemos ver que la columna TIPO_CLIENTE viene corrupta.

Por lo tanto debemos "limpiar" esta columna, eliminando estos caracteres extra y dejando solo la letra que corresponde al tipo de cliente.

Ejemplos prácticos

Recuerda revisar la
Ruta de ejercicios.
Ejercicio EM3-15 →



The screenshot shows the Trinket IDE interface. At the top, there is a navigation bar with a menu icon, the 'trinket' logo, a 'Run' button, a dropdown arrow, a '? Modules' button, a 'Share' button, and another dropdown arrow. Below this is a tab labeled 'main.py' with navigation arrows and icons for adding files, uploading, and saving. The main area displays a Python script with 22 lines of code. The code opens a CSV file, reads its lines, and processes the data into a matrix, specifically cleaning up the last column of the data.

```
1  archivo = open("ejemplo2.csv")
2
3  archivo_en_lineas = archivo.readlines()
4
5  archivo.close()
6
7  numero_de_clientes = len(archivo_en_lineas)
8
9  matriz_de_datos = []
10
11 ▼ for linea in archivo_en_lineas:
12     linea = linea.strip()
13     fila = linea.split(";")
14     matriz_de_datos.append(fila)
15
16 print(matriz_de_datos)
17
18 ▼ for fila in matriz_de_datos:
19     valor_tipo_cliente = fila[4]
20     fila[4] = valor_tipo_cliente[len(valor_tipo_cliente)-1]
21
22 print(matriz_de_datos)
```

Trabaja con **archivo_ejemplo2.csv**

Para poder limpiar la última columna, ocuparemos el siguiente código.

Ejemplos prácticos

Recuerda revisar la
Ruta de ejercicios.

Ejercicio EM3-15 →

The screenshot shows the Trinket IDE interface. At the top, there's a navigation bar with a menu icon, the 'trinket' logo, a 'Run' button, a dropdown arrow, a '? Modules' button, a 'Share' button, and another dropdown arrow. Below this is a file browser showing 'main.py'. The main editor area contains a Python script with two data matrices, each highlighted in a yellow box. The first matrix is labeled 'Matriz de datos antes de ser limpiada.' and the second is labeled 'Matriz de datos después de ser limpiada.'.

```
1
```

```
[[ '0', '14.782.991-7', 'Juan Francisco González Ru  
iz', '1980/2/26', '-.#!#A'], [ '1', '4.593.362-3', '  
Constanza Daniela Díaz Marín', '1968/6/9', '-.#!#C'  
], [ '2', '19.816.353-5', 'Pedro Rodrigo Rodríguez  
Muñoz', '1984/7/17', '-.#!#D'], [ '3', '16.804.654-9  
, 'Ignacio Bernardo Rodríguez Muñoz', '1990/8/6',  
'-.#!#D'], [ '4', '19.566.578-10', 'Alejandro Bernar
```

```
Matriz de datos después de ser limpiada.
```

```
[[ '0', '14.782.991-7', 'Juan Francisco González Ru  
iz', '1980/2/26', 'A'], [ '1', '4.593.362-3', 'Cons  
tanza Daniela Díaz Marín', '1968/6/9', 'C'], [ '2',  
'19.816.353-5', 'Pedro Rodrigo Rodríguez Muñoz', '  
1984/7/17', 'D'], [ '3', '16.804.654-9', 'Ignacio B  
ernardo Rodríguez Muñoz', '1990/8/6', 'D'], [ '4',  
'19.566.578-10', 'Alejandro Bernardo Campos Campos  
, '1977/8/14', 'B'], [ '5', '4.208.814-9', 'Victor
```

Ejemplos prácticos

Con el código anterior efectivamente se limpió la última columna.

Analicemos en detalle un extracto del código:

CÓDIGO

```
for linea in archivo_en_lineas:
    linea = linea.strip()
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)

for fila in matriz_de_datos:
    valor_tipo_cliente = fila[4]
    fila[4] = valor_tipo_cliente[len(valor_tipo_cliente)-1]

print(matriz_de_datos)
```

Hasta esta línea es todo igual a lo que hemos mostrado antes. Esta línea hace un `strip()` sobre cada fila del archivo CSV.

Esto logra eliminar el `'\n'` al final de cada fila, que vimos era muy molesto y nos deja la información limpia.

Ejemplos prácticos

Con el código anterior efectivamente se limpió la última columna.

Analicemos en detalle un extracto del código:

CÓDIGO

```
for linea in archivo_en_lineas:
    linea = linea.strip()
    fila = linea.split(";")
    matriz_de_datos.append(fila)

print(matriz_de_datos)

for fila in matriz_de_datos:
    valor_tipo_cliente = fila[4]
    fila[4] = valor_tipo_cliente[len(valor_tipo_cliente)-1]

print(matriz_de_datos)
```

En este caso, lo que hacemos es almacenar la información que está en la última columna (4) en la variable `valor_tipo_cliente`. Sabemos que todas las filas tienen el siguiente formato: `- . ! # A`

Donde el último carácter es lo que nos importa ya que indica realmente cuál es el tipo de cliente. Por eso, lo que hacemos es extraer el último carácter de la variable `valor_tipo_cliente`. Luego, lo asignamos a la columna 4 de esta fila y hacemos esto con todas las filas de la matriz. Eso nos permite extraer la información limpia y finalmente tener en la columna 4 al tipo de cliente.

Ejemplos prácticos

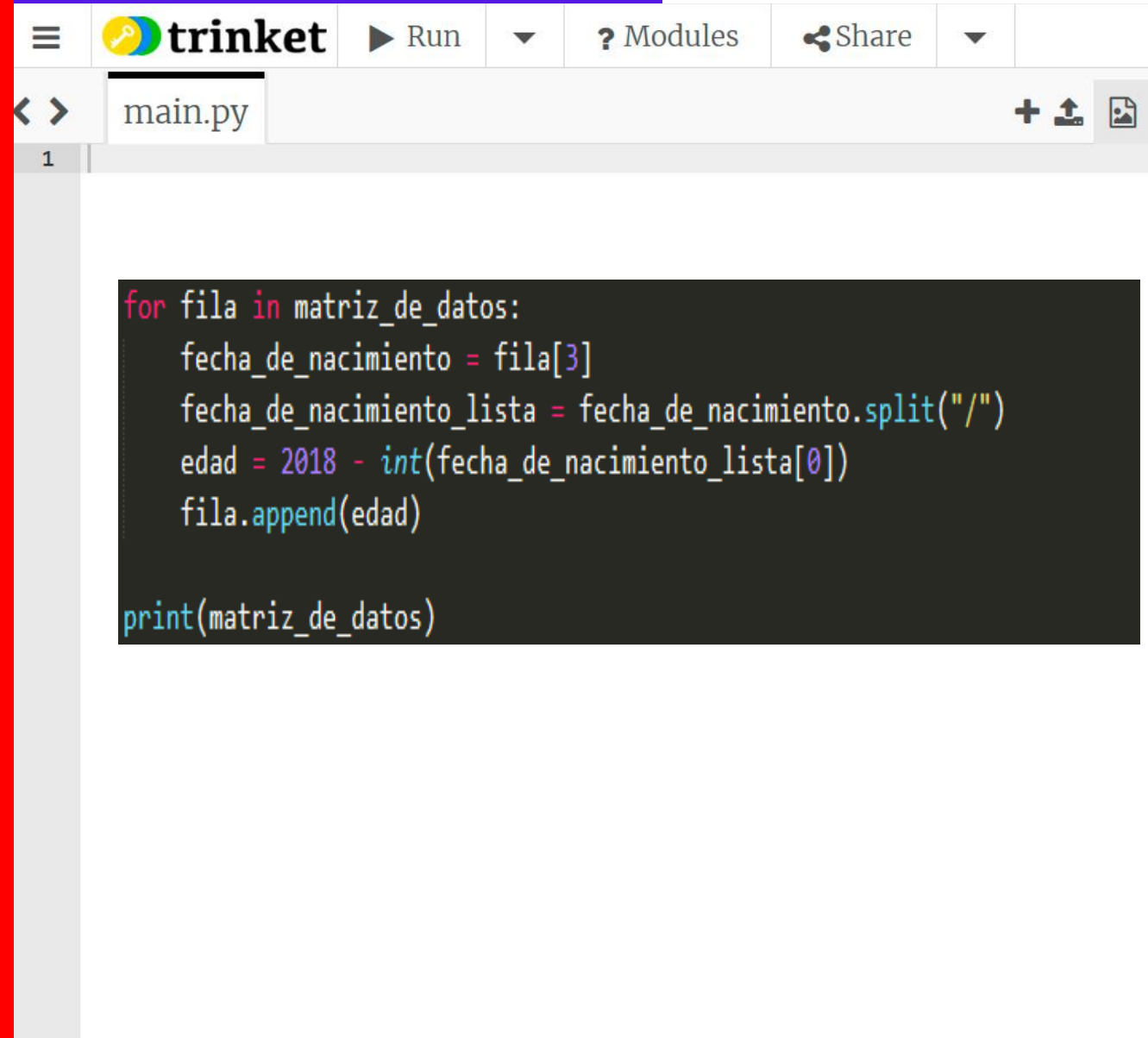
Ahora que la matriz de datos ya está limpia, podemos trabajar en ella. Específicamente, nos pidieron que agregáramos una columna "Edad", en base al año de la fecha de nacimiento de cada persona de la base de datos.

Veamos un código que lo hace:

Ejemplos prácticos

Recuerda revisar la
Ruta de ejercicios.

Ejercicio EM3-16 →



```
1
for fila in matriz_de_datos:
    fecha_de_nacimiento = fila[3]
    fecha_de_nacimiento_lista = fecha_de_nacimiento.split("/")
    edad = 2018 - int(fecha_de_nacimiento_lista[0])
    fila.append(edad)
print(matriz_de_datos)
```

Matriz de datos después de crear la
columna "Edad"

```
[['0', '14.782.991-7', 'Juan Francisco González Ru  
iz', '1980/2/26', 'A', 38], ['1', '4.593.362-3', 'C  
Constanza Daniela Díaz Marín', '1968/6/9', 'C', 50  
, ['2', '19.816.353-5', 'Pedro Rodrigo Rodríguez  
Muñoz', '1984/7/17', 'D', 34], ['3', '16.804.654-9  
, 'Ignacio Bernardo Rodríguez Muñoz', '1990/8/6',  
'D', 28], ['4', '19.566.578-10', 'Alejandro Bernar
```


Ejemplos prácticos

Analicemos el código que creó la columna edad:

CÓDIGO

```
for fila in matriz_de_datos:
    fecha_de_nacimiento = fila[3]
    fecha_de_nacimiento_lista = fecha_de_nacimiento.split("/")
    edad = 2018 - int(fecha_de_nacimiento_lista[0])
    fila.append(edad)

print(matriz_de_datos)
```

Se toma la tercera columna que corresponde a la fecha de nacimiento, y lo almacenamos en la variable `fecha_de_nacimiento`.

Ejemplos prácticos

Analicemos el código que creó la columna edad:

CÓDIGO

```
for fila in matriz_de_datos:
    fecha_de_nacimiento = fila[3]
    fecha_de_nacimiento_lista = fecha_de_nacimiento.split("/")
    edad = 2018 - int(fecha_de_nacimiento_lista[0])
    fila.append(edad)

print(matriz_de_datos)
```

Luego, hacemos la función `split("/")` sobre esta variable, aprovechando que el año, mes y día están separados por el carácter `"/"`. Esto genera una lista que tiene tres elementos (año, mes y día), la que guardamos en la variable `fecha_de_nacimiento_lista`.

Ejemplos prácticos

Analicemos el código que creó la columna edad:

CÓDIGO

```
for fila in matriz_de_datos:
    fecha_de_nacimiento = fila[3]
    fecha_de_nacimiento_lista = fecha_de_nacimiento.split("/")
    edad = 2018 - int(fecha_de_nacimiento_lista[0])
    fila.append(edad)

print(matriz_de_datos)
```

Para calcular la edad, al año actual le restamos el año de la fecha de nacimiento. El año de la fecha de nacimiento se encuentra en la posición 0 de la variable

`fecha_de_nacimiento_lista`. Además, es de tipo `string` por lo que debemos transformarla a entero (por eso es que ocupamos el comando `int()`).

Ejemplos prácticos

Analicemos el código que creó la columna edad:

CÓDIGO

```
for fila in matriz_de_datos:
    fecha_de_nacimiento = fila[3]
    fecha_de_nacimiento_lista = fecha_de_nacimiento.split("/")
    edad = 2018 - int(fecha_de_nacimiento_lista[0])
    fila.append(edad)

print(matriz_de_datos)
```

Finalmente, a la fila agregamos la edad mediante el comando `append`. Por eso podemos ver, en la matriz de datos que se imprime en consola posteriormente a la ejecución de este código, que se agregó un nuevo elemento en cada fila, que corresponde a la edad de cada persona.

Ejemplos prácticos

Recuerda revisar la
Ruta de ejercicios.
Ejercicio EM3-17 →



The screenshot shows the Trinket Python IDE interface. At the top, there's a navigation bar with a menu icon, the 'trinket' logo, a 'Run' button, a dropdown arrow, a 'Modules' button with a question mark, and a 'Share' button with a share icon. Below this, the file name 'main.py' is displayed. The main area shows a Python script that iterates through a list of data rows and writes them to a CSV file named 'ejemplo_con_edad.csv'. The script uses a loop to process each row, adding a semicolon and a newline character to separate the fields. The file is opened in write mode ('w') and closed after writing.

```
1
archivo_guardar = open("ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

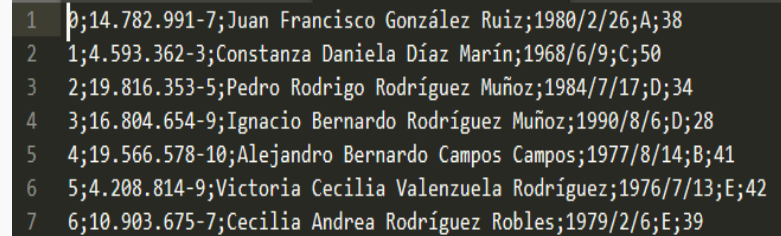
    for i in range(0, len(fila)):
        if i == len(fila) - 1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

Archivo generado



The screenshot shows the content of the generated CSV file, 'ejemplo_con_edad.csv'. It contains seven lines of data, each representing a row from the original dataset. The data is formatted as a semicolon-separated list of values, with a newline character at the end of each row.

```
1 0;14.782.991-7;Juan Francisco González Ruiz;1980/2/26;A;38
2 1;4.593.362-3;Constanza Daniela Díaz Marín;1968/6/9;C;50
3 2;19.816.353-5;Pedro Rodrigo Rodríguez Muñoz;1984/7/17;D;34
4 3;16.804.654-9;Ignacio Bernardo Rodríguez Muñoz;1990/8/6;D;28
5 4;19.566.578-10;Alejandro Bernardo Campos Campos;1977/8/14;B;41
6 5;4.208.814-9;Victoria Cecilia Valenzuela Rodríguez;1976/7/13;E;42
7 6;10.903.675-7;Cecilia Andrea Rodríguez Robles;1979/2/6;E;39
```

Finalmente, ahora que limpiamos la base de datos y agregamos la columna "edad" podemos guardar lo que hicimos en un nuevo archivo.

Para eso, ocuparemos el siguiente código en Python.

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila)-1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

Primero abrimos un archivo nuevo "ejemplo_con_edad.csv", mediante el comando `open` y el modo "w". Con esto podemos guardar nuestra matriz de datos en un archivo completamente nuevo.

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila)-1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

A continuación hacemos un `for` que recorra cada una de las filas de `matriz_de_datos`. Estas son las filas que se guardarán en el archivo `"archivo_ejemplo_con_edad.csv"`. Recordemos que esta fila es una lista, cuyos elementos son cada una de las columnas. Para poder guardarlo como un archivo CSV, debemos crear un `string` que tenga estos elementos unidos por `;`. Por eso creamos una variable auxiliar `fila_para_escribir`, que será la que finalmente escribamos en el archivo CSV.

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila) - 1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

Después, hacemos un nuevo `for` que recorra cada una de las columnas de cada fila. Para eso, este `for` tendrá una variable `i` que irá representando cada una de las posiciones de la lista `fila`. Estas posiciones representarán a cada una de las columnas.

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila)-1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

Para cada iteración de este `for`, verificamos si `i` es igual a la última columna. Si no es igual a la última columna, entonces añadimos el contenido de esta columna a nuestra variable `fila_para_escribir`, seguido del carácter `;`. Así estamos emulando el formato de un archivo CSV.

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila)-1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

No obstante, notemos que cuando `i` es igual a la última columna, esto cambia levemente. Esto se debe a que en la última columna no debemos agregar ";", sino que solo su contenido. Cabe destacar que la última fila la debemos transformar a `string` (mediante el comando `str()`), ya que era la edad y era de tipo entero (y no podemos agregar un entero a un `string`).

Ejemplos prácticos

Analicemos el código anterior:

CÓDIGO

```
archivo_guardar = open("archivo_ejemplo_con_edad.csv", "w")

for fila in matriz_de_datos:
    fila_para_escribir = ""

    for i in range(0, len(fila)):
        if i == len(fila)-1:
            fila_para_escribir += str(fila[i])
        else:
            fila_para_escribir += fila[i] + ";"

    fila_para_escribir += "\n"

    archivo_guardar.write(fila_para_escribir)

archivo_guardar.close()
```

Luego de haber agregado todas las columnas seguidas de un ";" (excepto la última), agregamos un salto de línea ("\n"). Con esto seguimos fielmente el formato CSV donde cada fila del archivo CSV corresponde a una fila de la base de datos. Finalmente, y mediante la función `write` en la variable `archivo_guardar` (que contiene al archivo donde guardaremos nuestra matriz de datos), escribimos cada fila. Al terminar el `for` general cerramos el archivo abierto mediante la función `close()`. Todo este proceso nos permitió guardar nuestro trabajo en un formato CSV.

>>> Cierre

Has finalizado la revisión de los contenidos de esta clase.

A continuación, te invitamos a realizar las actividades y a revisar los recursos del módulo que encontrarás en plataforma.