

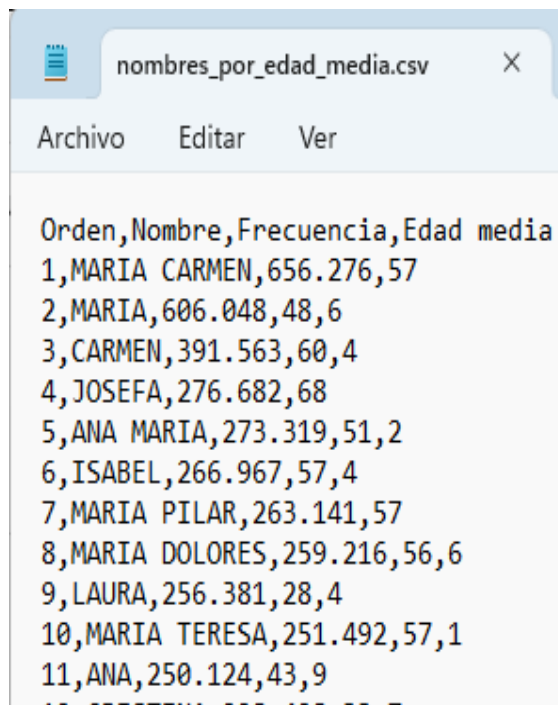
Archivos CSV y el formato JSON ¶

En Python y en cualquier otro lenguaje, los datos de entrada pueden proceder de distintas fuentes: lo más sencillo es introducir datos desde la consola, también es posible hacer que el programa los descargue de una página web de Internet, o que extraiga la información contenida en un archivo almacenado en el disco.

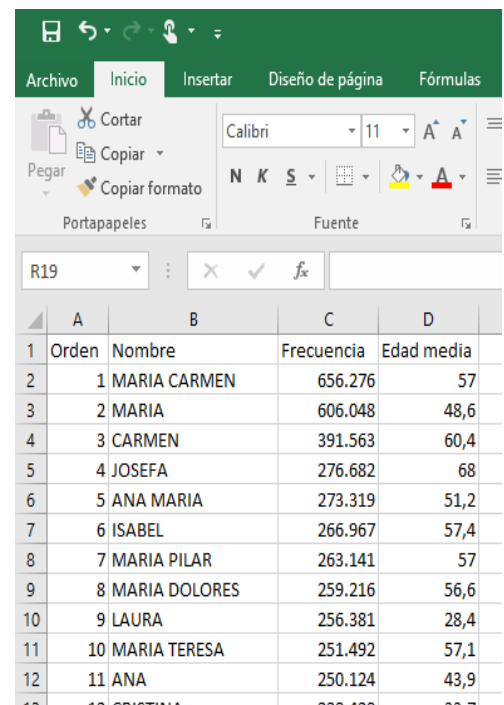
Este último caso es muy frecuente. Un archivo puede venir en texto plano, o codificado en un formato especial, como es el caso de los archivos `csv` o los archivos `json`. En esta pequeña sesión atendemos a estos dos casos.

Archivos CSV

He aquí un fragmento de un archivo `csv`, abierto con el portapapeles y con excell, respectivamente:



```
Orden,Nombre,Frecuencia,Edad media
1,MARIA CARMEN,656.276,57
2,MARIA,606.048,48,6
3,CARMEN,391.563,60,4
4,JOSEFA,276.682,68
5,ANA MARIA,273.319,51,2
6,ISABEL,266.967,57,4
7,MARIA PILAR,263.141,57
8,MARIA DOLORES,259.216,56,6
9,LAURA,256.381,28,4
10,MARIA TERESA,251.492,57,1
11,ANA,250.124,43,9
12,CRISTINA,220.420,22,7
```



	A	B	C	D
1	Orden	Nombre	Frecuencia	Edad media
2	1	MARIA CARMEN	656.276	57
3	2	MARIA	606.048	48,6
4	3	CARMEN	391.563	60,4
5	4	JOSEFA	276.682	68
6	5	ANA MARIA	273.319	51,2
7	6	ISABEL	266.967	57,4
8	7	MARIA PILAR	263.141	57
9	8	MARIA DOLORES	259.216	56,6
10	9	LAURA	256.381	28,4
11	10	MARIA TERESA	251.492	57,1
12	11	ANA	250.124	43,9
13	12	CRISTINA	220.420	22,7

El manejo de archivos `csv` es sencillo en Python. Como referencia básica, puede verse la siguiente, entre otras muchas:

<https://docs.python.org/3/library/csv.html>

El formato `csv` (*Comma Separated Values*) es el más usado para almacenar tablas de datos en archivos de disco.

In [1]: `import csv`

```
csvFile = csv.reader(open("nombres_por_edad_media.csv", "r"))
for row in csvFile:
    print(row)
```

```
['Orden', 'Nombre', 'Frecuencia', 'Edad media']
['1', 'MARIA CARMEN', '656.276', '57']
['2', 'MARIA', '606.048', '48', '6']
['3', 'CARMEN', '391.563', '60', '4']
['4', 'JOSEFA', '276.682', '68']
['5', 'ANA MARIA', '273.319', '51', '2']
['6', 'ISABEL', '266.967', '57', '4']
['7', 'MARIA PILAR', '263.141', '57']
['8', 'MARIA DOLORES', '259.216', '56', '6']
['9', 'LAURA', '256.381', '28', '4']
['10', 'MARIA TERESA', '251.492', '57', '1']
['11', 'ANA', '250.124', '43', '9']
['12', 'CRISTINA', '228.428', '33', '7']
['13', 'MARIA ANGELES', '226.047', '55', '4']
['14', 'MARTA', '225.323', '29', '3']
['15', 'FRANCISCA', '213.820', '64', '9']
['16', 'ANTONIA', '207.597', '64', '7']
['17', 'MARIA ISABEL', '204.354', '52', '8']
['18', 'MARIA JOSE', '203.283', '46', '1']
```

In [2]: `# De otro modo:`

```
with open('nombres_por_edad_media.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for row in reader:
        print(row)
```

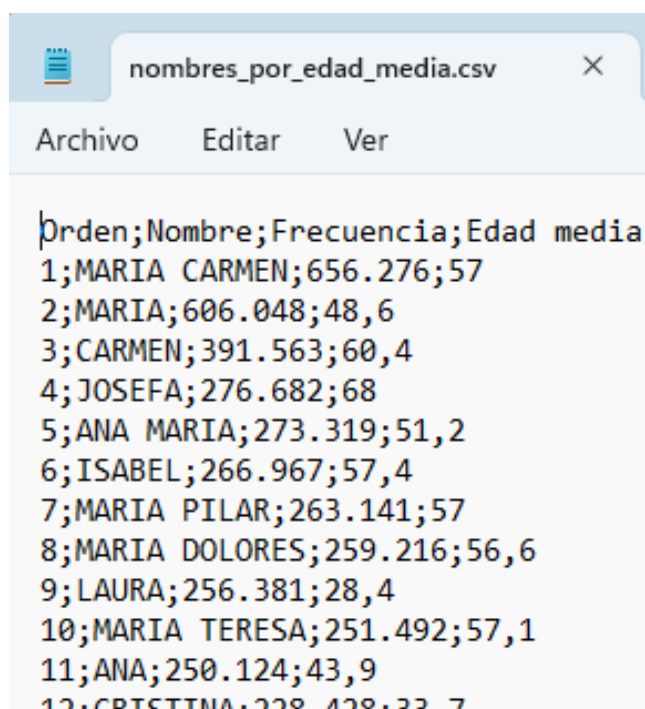
```
['Orden', 'Nombre', 'Frecuencia', 'Edad media']
['1', 'MARIA CARMEN', '656.276', '57']
['2', 'MARIA', '606.048', '48', '6']
['3', 'CARMEN', '391.563', '60', '4']
['4', 'JOSEFA', '276.682', '68']
['5', 'ANA MARIA', '273.319', '51', '2']
['6', 'ISABEL', '266.967', '57', '4']
['7', 'MARIA PILAR', '263.141', '57']
['8', 'MARIA DOLORES', '259.216', '56', '6']
['9', 'LAURA', '256.381', '28', '4']
['10', 'MARIA TERESA', '251.492', '57', '1']
['11', 'ANA', '250.124', '43', '9']
['12', 'CRISTINA', '228.428', '33', '7']
['13', 'MARIA ANGELES', '226.047', '55', '4']
['14', 'MARTA', '225.323', '29', '3']
['15', 'FRANCISCA', '213.820', '64', '9']
['16', 'ANTONIA', '207.597', '64', '7']
['17', 'MARIA ISABEL', '204.354', '52', '8']
['18', 'MARIA JOSE', '203.283', '46', '1']
```

```
In [3]: # Y también, Leyendo la cabecera por separado,
# seleccionando algunas columnas y realizando una conversión, ya de paso

with open('nombres_por_edad_media.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    cab = next(reader)
    print(cab[1], cab[3])
    for row in reader:
        nombre, edad_media = row[1], row[3]
        print(nombre, int(edad_media))
```

Nombre Edad media
 MARIA CARMEN 57
 MARIA 48
 CARMEN 60
 JOSEFA 68
 ANA MARIA 51
 ISABEL 57
 MARIA PILAR 57
 MARIA DOLORES 56
 LAURA 28
 MARIA TERESA 57
 ANA 43
 CRISTINA 33
 MARIA ANGELES 55
 MARTA 29
 FRANCISCA 64
 ANTONIA 64
 MARIA ISABEL 52
 MARIA JOSE 46
 CRISTINA 33

Aunque el separador natural en csv es la coma, es posible usar otro, como puede ser el punto y coma:



Archivo csv abierto con excel

Las instrucciones anteriores deberían especificar el parámetro opcional así:
 delimiter=';' , cuando el separador sea distinto de la coma:

```
csv_file = csv.reader(open("nombres_por_edad_media_pc.csv", "r", de
limiter=';'))
reader = csv.reader(csv_file, delimiter=';')
```

In [4]: **▶** *# Escritura:*

```
ids_columnas = ['Nombre', 'Matemáticas', 'Lengua', 'Historia']

filas = [ ['Juan', '5.7', '2.5', '9.0'],
          ['Sara', '9.5', '6.7', '9.3'],
          ['Alberto', '7.5', '7.5', '7.5'],
          ['Sara', '4.9', '5.2', '8.0']]

with open("calificaciones_2019.csv", 'w', newline='') as csv_archivo:
    csvwriter = csv.writer(csv_archivo, delimiter=";")
    csvwriter.writerow(ids_columnas)
    csvwriter.writerows(filas)

print("Hecho")
```

Hecho

Transformación de datos

Los datos leídos de un csv son siempre cadenas de caracteres, pero se pueden convertir en los formatos necesarios con las funciones (y librerías) adecuadas:

In [5]: **▶** *# Enteros:*

```
print(int("7"),int("123.000".replace('.', '')))

# Reales:

print(float("4.5"), float("4,5".replace(",",".")))
print(float("123.000,75".replace('.', '').replace(',','.')))

# Fechas:

from datetime import datetime
fecha_str = '10-24-2019'

fecha_objeto = datetime.strptime(fecha_str, '%m-%d-%Y').date()
print(type(fecha_objeto))
print(fecha_objeto)
```

```
7 123000
4.5 4.5
123000.75
<class 'datetime.date'>
2019-10-24
```

(Las transformaciones de datos deben realizarse únicamente con los datos sin la cabecera.)

El formato JSON

El formato JSON es una notación sencilla para especificar datos y facilitar su intercambio. En la wikipedia puede leerse que se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, actualmente se considera un formato independiente del lenguaje.

La idea subyacente a este formato es explotar la codificación mediante el emparejamiento de clave-valor, y la utilización de listas. Los siguientes ejemplos se han tomado de la dirección siguiente:

<https://support.oneskyapp.com/hc/en-us/articles/208047697-JSON-sample-files>

Ejemplo 1:

```
{  
  "fruit": "Apple",  
  "size": "Large",  
  "color": "Red"  
}
```

Ejemplo 2:

```
{
    "quiz": {
        "sport": {
            "q1": {
                "question": "Which one is correct team name in NB
A?",
                "options": [
                    "New York Bulls",
                    "Los Angeles Kings",
                    "Golden State Warriros",
                    "Huston Rocket"
                ],
                "answer": "Huston Rocket"
            }
        },
        "maths": {
            "q1": {
```

In [6]: ▶ `import json`

```
with open("example_1.json") as archivo:
    datos = json.loads(archivo.read())

print(datos)
print(type(datos))
```

```
{'fruit': 'Apple', 'size': 'Large', 'color': 'Red'}
<class 'dict'>
```

In [7]: ▶ *# Impresión con sangrado:*

```
print(json.dumps(datos, indent=4))
```

```
{
    "fruit": "Apple",
    "size": "Large",
    "color": "Red"
}
```

In [8]: ▶ *# Obsérvese que el archivo `example_1.json`
no contiene un diccionario,
sino únicamente cadenas de caracteres:*

```
with open("example_1.json") as archivo:
    for row in archivo:
        print(row, end="")
```

```
{
    "fruit": "Apple",
    "size": "Large",
    "color": "Red"
}
```

```
In [9]:  with open("example_2.json") as archivo:
          datos = json.loads(archivo.read())
          datos
```

```
Out[9]: {'quiz': {'sport': {'q1': {'question': 'Which one is correct team name
in NBA?',
  'options': ['New York Bulls',
    'Los Angeles Kings',
    'Golden State Warriros',
    'Huston Rocket'],
  'answer': 'Huston Rocket'}}},
  'maths': {'q1': {'question': '5 + 7 = ?',
    'options': ['10', '11', '12', '13'],
    'answer': '12'},
  'q2': {'question': '12 - 8 = ?',
    'options': ['1', '2', '3', '4'],
    'answer': '4'}}}}
```

```
In [10]:  # Escritura:

          with open("example_3.json", "w") as archivo:
              archivo.write(json.dumps(datos))

          # Obviamente, Los archivos example_2.json y example_3.json son iguales
```

Se puede cargar un archivo json en un diccionario...

```
In [11]:  import json

          f = open("estaciones.json")
          estaciones_dicc = json.load(f)
          estaciones_dicc
```

```
Out[11]: [{'latitud': '431825N',
  'provincia': 'A CORUÑA',
  'altitud': '98',
  'indicativo': '1387E',
  'nombre': 'A CORUÑA AEROPUERTO',
  'indsinop': '08002',
  'longitud': '082219W'},
  {'latitud': '432157N',
  'provincia': 'A CORUÑA',
  'altitud': '58',
  'indicativo': '1387',
  'nombre': 'A CORUÑA',
  'indsinop': '08001',
  'longitud': '082517W'},
  {'latitud': '430938N',
  'provincia': 'A CORUÑA',
  'altitud': '50',
  'indicativo': '1393',
  'nombre': 'CABO VILAN',
  'indsinop': '08000'}
```

Y luego pasar el diccionario a una tabla de pandas:

```
In [12]: import pandas

estaciones = pandas.DataFrame(estaciones_dicc)
estaciones
```

Out[12]:

	latitud	provincia	altitud	indicativo	nombre	indsinop	longitud
0	431825N	A CORUÑA	98	1387E	A CORUÑA AEROPUERTO	08002	082219W
1	432157N	A CORUÑA	58	1387	A CORUÑA	08001	082517W
2	430938N	A CORUÑA	50	1393	CABO VILAN	08006	091239W
3	434710N	A CORUÑA	80	1351	ESTACA DE BARES	08004	074105W
4	425529N	A CORUÑA	230	1400	FISTERRA	08040	091729W
...
286	411952N	ZARAGOZA	600	9394X	CALATAYUD	08156	013843W
287	410652N	ZARAGOZA	779	9390	DAROCA	08157	012436W
288	422927N	ZARAGOZA	626	9244X	SOS DEL REY CATÓLICO	08090	011249W
289	413938N	ZARAGOZA	249	9434	ZARAGOZA AEROPUERTO	08160	010015W
290	413715N	ZARAGOZA	254	9434P	ZARAGOZA, VALDESPARTERA	08159	005606W

291 rows × 7 columns

```
In [13]: estaciones.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 291 entries, 0 to 290
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   latitud     291 non-null   object
1   provincia   291 non-null   object
2   altitud     291 non-null   object
3   indicativo  291 non-null   object
4   nombre      291 non-null   object
5   indsinop    291 non-null   object
6   longitud    291 non-null   object
dtypes: object(7)
memory usage: 16.0+ KB
```

Pero también se puede leer directamente un archivo json en una tabla de pandas:


```
In [14]: ▶ datos_pandas = pandas.read_json("estaciones.json", encoding="latin1")
datos_pandas
```

Out[14]:

	latitud	provincia	altitud	indicativo	nombre	indsinop	longitud
0	431825N	A CORUÑA	98	1387E	A CORUÑA AEROPUERTO	08002	082219W
1	432157N	A CORUÑA	58	1387	A CORUÑA	08001	082517W
2	430938N	A CORUÑA	50	1393	CABO VILAN	08006	091239W
3	434710N	A CORUÑA	80	1351	ESTACA DE BARES	08004	074105W
4	425529N	A CORUÑA	230	1400	FISTERRA	08040	091729W
...
286	411952N	ZARAGOZA	600	9394X	CALATAYUD	08156	013843W
287	410652N	ZARAGOZA	779	9390	DAROCA	08157	012436W
288	422927N	ZARAGOZA	626	9244X	SOS DEL REY CATÓLICO	08090	011249W
289	413938N	ZARAGOZA	249	9434	ZARAGOZA AEROPUERTO	08160	010015W
290	413715N	ZARAGOZA	254	9434P	ZARAGOZA, VALDESPARTERA	08159	005606W

291 rows × 7 columns