file-entrada-salida

January 28, 2020

1 Archivo Entrada y Salida

Los programas requieren datos, y a veces muchos datos. Hay diferentes maneras de guardar y acceder a ello, pero una de las mas comunes es a traves del sistema de archivos de una computadora. Por tanto, trabajar con archivos es una herramienta sumamente util para cualquier programador.

Operacions comunes involucran: guardando el dato de salida de un programa a un archivo de texto, limpiar un documento tabulado a fin que cada columna este en el formato correcto, o eliminar archivos grandes de un directorio en el disco duro.

En esta seccion aprenderemos a: 1. Leer y escribir archivos 2. Trabajar con rutas de archivos 3. Trabajar con archivos CSV

2 Leer y escribir archivos

Hasta ahora hemos visto programas que toman dato de entrada del mismo programa o del usuario. Si han trabajado con muchos datos, estos metodos son problematicos. En muchas aplicaciones, el dato de entrada es leido de algun(os) archivo(s). Python tiene herramientas que nos permiten realizar estas operaciones.

2.1 Escribir a un archivo

Para escribir un archivo de texto sin formato, podemos utilizar la funcion general incorporada `open()`. Cuando abrimos un archivo con `open()`, lo primero que debemos determinar es si realizamos una operacion de lectura o de escritura.

[ ]:
```
# abrir y escribir a un archivo
archivo_salida = open('hola.txt', 'w')  # abrimos en modo "w" de escritura
```

[ ]:
```
# escribimos una linea de texto con writelines()
archivo_salida.writelines('Este es mi primer archivo')
archivo_salida.close()
```

Cuando solo proporcionamos el nombre del archivo, este se creará en el mismo directorio que tiene el script, toda vez que no proporcionamos la ruta del archivo.

Cuando abrimos un archivo con `open()`, siempre debemos cerrar el archivo con `close()`. Python cierra automaticamente los archivos que uno abre, pero si no los cerramos, puede ocasionar problemas no esperados.

Despues de ejecutar el script, observamos un archivo nuevo en el directorio denominado `hola.txt`, con la linea escrita `Este es mi primer archivo`.

[ ]:
```
# writelines() tambien toma una lista de lineas
archivo_salida = open('hola.txt', 'w')  # si abrimos un archivo existente, el␣
 ↪contenido viejo se borra
lineas = [
    'Este archivo es nuevo',
    'Contiene esta linea',
    'Esta otra linea tambien'
]
archivo_salida.writelines(lineas)  # las lineas son escritas seguidamente sin␣
 ↪espacio
archivo_salida.close()
```

[1]:
```
# para escribir en una linea nueva, insertemos el caracter de nueva linea "\n"
archivo_salida = open('hola.txt', 'w')
lineas = [
    'Este archivo es nuevo',
    '\nContiene esta linea',     # \n en linea nueva
    '\nEsta otra linea tambien'  # \n en linea nueva
]
archivo_salida.writelines(lineas)  # las lineas son escritas seguidamente sin␣
 ↪espacio
archivo_salida.close()
```

Si abrimos el archivo, observamos cada linea en una linea nueva:

```
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
```

[2]:
```
# podemos abrir el archivo en modo "a" que permita "adjuntar"
archivo_salida = open('hola.txt', 'a')  # abrimos el archivo pero no borra el␣
 ↪contenido
archivo_salida.writelines('\nEsta linea se adjunta')  # linea nueva
archivo_salida.close()
```

Si abrimos el archivo, observamos cada linea en una linea nueva:

```
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
Esta linea se adjunta
```

2.2 Leer un archivo

[3]:
```
# abrimos el archivo en modo lectura "r"
archivo_entrada = open('hola.txt', 'r')
print(archivo_entrada.readlines())  # utilizamos el metodo readlines() que␣
↪retorna una lista de lineas
archivo_entrada.close()
```

```
['Este archivo es nuevo\n', 'Contiene esta linea\n', 'Esta otra linea
tambien\n', 'Esta linea se adjunta']
```

[4]:
```
# ya que el archivo retorna una lista, podemos ciclar sobre la misma
archivo_entrada = open('hola.txt', 'r')
for linea in archivo_entrada.readlines():
    print(linea)
archivo_entrada.close()
```

```
Este archivo es nuevo

Contiene esta linea

Esta otra linea tambien

Esta linea se adjunta
```

[5]:
```
# observamos que hay una nueva linea vacia entre cada linea, podemos anular␣
↪este comportamiento
archivo_entrada = open('hola.txt', 'r')
for linea in archivo_entrada.readlines():
    print(linea, end='')  # print incluye nueva linea automaticamente,␣
↪especificamos comportamiento deseado end=''
archivo_entrada.close()
```

```
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
Esta linea se adjunta
```

[6]:
```
# podemos leer linea por linea con readline() en singular
archivo_entrada = open('hola.txt', 'r')
linea = archivo_entrada.readline()  # lee la primera linea
while linea != '':  # si no hemos llegao al fin
    print(linea, end='')  # imprime la linea
    linea = archivo_entrada.readline()  # lee la proxima linea
archivo_entrada.close()
```

```
Este archivo es nuevo
Contiene esta linea
```

```
Esta otra linea tambien
Esta linea se adjunta
```

Cuando Python lee un archivo, gestiona un tipo de marcador que recuerda su ubicacion en el documento. Es asi como el metodo `readline()` funciona, ya que lee la primera linea, y cuando el metodo se ejecuta nuevamente, Python lee el documento a partir de donde dejó el marcador la última vez, por tanto lee la próxima linea del documento, y asi sucesivamente hasta llegar al final del archivo. Cuando se cierra el archivo con `close()`, el marcador vuelve a reiniciarse, a fin de que la proxima vez pueda empezar desde el principio del archivo. El metodo `readlines()` tambien se comporta de la misma manera.

Esto lo podemos comprobar si leemos el archivo, ejecutamos readlines, y sin cerrar el archivo, ejecutamos nuevamente el metodo `readlines()`. Ya que el marcador sigue al final del documento, `readlines()` no retorna linea alguna, porque no hay nada que leer.

[7]:
```
archivo_entrada = open('hola.txt', 'r')
print('Primera vez:')
for linea in archivo_entrada.readlines():
    print(linea, end='')

print('\n\nSegunda vez:')
for linea in archivo_entrada.readlines():
    print(linea, end='')
archivo_entrada.close()
```

```
Primera vez:
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
Esta linea se adjunta

Segunda vez:
```

[8]:
```
# si queremos acceder al contenido del archivo nuevamente, es mejor guardarlo␣
 ↪en una lista
archivo_entrada = open('hola.txt', 'r')

lineas = archivo_entrada.readlines()

print('Primera vez:')
for linea in lineas:
    print(linea, end='')

print('\n\nSegunda vez:')
for linea in lineas:
    print(linea, end='')
archivo_entrada.close()
```

```
Primera vez:
```

```
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
Esta linea se adjunta

Segunda vez:
Este archivo es nuevo
Contiene esta linea
Esta otra linea tambien
Esta linea se adjunta
```

[9]:
```python
# para no tener que cerrar el archivo, Python lo hace automaticamente con la␣
 ↪palabra with
# with establece un contexto
with open('hola.txt', 'r') as archivo:
    for line in archivo.readlines():
        print(line)
```

```
Este archivo es nuevo

Contiene esta linea

Esta otra linea tambien

Esta linea se adjunta
```

[10]:
```python
# podemos abrir mutiples archivos en la misma operacion
with open('hola.txt', 'r') as fuente, open('salida.txt', 'w') as salida:
    for line in fuente.readlines():
        salida.write(line)

# todo el contenido del primer archivo se copio a este
with open('salida.txt', 'r') as archivo:
    for line in archivo.readlines():
        print(line)
```

```
Este archivo es nuevo

Contiene esta linea

Esta otra linea tambien

Esta linea se adjunta
```

2.3 Ejercicios

1. Abra y escriba un archivo que contenga varias lineas utilizando writelines()
2. Abra el archivo que escribio y lea sus contenidos con readlines() y readline()

3. Abra y lea el archivo que escribio utilizando with()
4. Abra y lear al archivo que escribio utilizando with() y copie los contenidos del mismo a otro archivo

3 Trabajando con Rutas en Python

Lo mas probable es que vamos a necesitar abrir archivos ubicados en otros directorios, y no solamente los archivos ubicados en el directorio actual donde reside el script. Para acceder a distintos directorions, podemos ingresar la ruta completa directamente como argumento a la funcion incorporada `open(ruta_absoluta_del_archivo)`.

```
archivo = open('C:/home/adriaanbd/documentos/hola.txt', 'r')
```

Observemos el uso del `/`. Las rutas de Windows contienen un `\` en vez de un `/`, pero en Python podemos substituir el `\` por el `/`, toda vez que el `\` tiene un significado especial en Python por ser utilizado como un caracter de escape, lo que quiere decir que el caracter que le sigue inmediatamente, e.g. `\n` es tratado como caracter especial. Python entiende que el uso del `\` con el caracter a continuacion es un caracter especial. Por ejemplo, `\n` significa una nueva linea, `\t` significa un caracter `tab` que representa 2 o 4 caracteres de espacios en la misma linea.

```
# podemos utilizar el \ de la siguiente manera
path = r'C:\home\adriaanbd\documentos\hola.txt'
```

3.1 El modulo `os`

Si deseamos hacer algo mas avanzado con estructuras de archivos, vamos a tener que hacer uso del modulo `os`, que expone varias funciones del sistema operativo. Lo primero que tenemos que hacer es importar el modulo.

[11]:
```
# esto importa el modulo al programa
import os
```

[16]:
```
# para crear un directorio nuevo en el directorio donde reside este programa
os.mkdir('mi-directorio')
```

```
␣
↪---------------------------------------------------------------------------

FileExistsError                           Traceback (most recent call␣
↪last)

<ipython-input-16-ddac2c5353aa> in <module>
  1 # para crear un directorio nuevo en el directorio donde reside este␣
↪programa
----> 2 os.mkdir('mi-directorio')
```

```
FileExistsError: [Errno 17] File exists: 'mi-directorio'
```

[17]:
```
# para crear el directorio en una ruta especifica
ruta = 'mi-directorio'
os.mkdir(os.path.join(ruta, 'subdirectorio'))  # utilizemos os.path.join para␣
↪concatenar dos strings
```

[12]:
```
# pudimos haber concatenado asi tambien:
ruta = 'C:/home/adriaanbd/documentos'
directorio = ruta + '/' + 'mi-directorio'
print(directorio)
```

```
C:/home/adriaanbd/documentos/mi-directorio
```

[21]:
```
ruta = 'mi-directorio'
directorio = os.path.join(ruta, 'subdirectorio')
directorio
```

[21]:
```
'mi-directorio/subdirectorio'
```

[22]:
```
# para eliminar un directorio usemos rmdir()
os.rmdir(directorio)
```

[24]:
```
ruta = 'mi-directorio'
os.rmdir(ruta)
```

[26]:
```
# para obtener una lista de los archivos en un directorio usemos os.listdir()
os.listdir()  # su dato de salida podrá ser distinto
```

[26]:
```
['contenido.ipynb',
 '.vscode',
 'intro.ipynb',
 'errores.ipynb',
 'contenido.md',
 'funciones-y-ciclos.ipynb',
 'salida',
 'file-entrada-salida.ipynb',
 '.ipynb_checkpoints',
 'otros-temas',
 'oop.ipynb',
 '.gitignore',
 'numeros-y-matematica.ipynb',
 'tips.ipynb',
 'encontrando-resolviendo-errores.ipynb',
 '.git',
 'datos',
```

```
 'logica-condicional-control-de-flujo.ipynb',
 'variables.ipynb',
 'strings.ipynb',
 'textos-llamadas.ipynb',
 'hola.txt',
 'juego-de-aventura.ipynb',
 'tuplas-listas-diccionarios.ipynb',
 '.python-version',
 'salida.txt']
```

[27]:
```
# una lista de los archivos con terminacion txt usando endswith()
for archivo in os.listdir():
    if archivo.lower().endswith('txt'):
        print(archivo)
```

```
hola.txt
salida.txt
```

3.2 el modulo glob

[28]:
```
# este modulo nos ayuda a encontrar patrones con caracteres comodin
import glob
glob.glob('*.txt')  # el asterisco * es un comodin que representa todo, por␣
↪tanto todo archivo con extension .txt
```

[28]:
```
['hola.txt', 'salida.txt']
```

3.3 Verificando la existencia de archivos y directorios

[42]:
```
for archivo in os.listdir():
    print(f'Archivo: "{archivo}", \nes un directorio: {os.path.
↪isdir(archivo)}\n\n')  # es un directorio?
```

```
Archivo: "contenido.ipynb",
es un directorio: False


Archivo: ".vscode",
es un directorio: True


Archivo: "intro.ipynb",
es un directorio: False


Archivo: "errores.ipynb",
es un directorio: False
```

```
Archivo: "contenido.md",
es un directorio: False


Archivo: "funciones-y-ciclos.ipynb",
es un directorio: False


Archivo: "salida",
es un directorio: True


Archivo: "file-entrada-salida.ipynb",
es un directorio: False


Archivo: ".ipynb_checkpoints",
es un directorio: True


Archivo: "otros-temas",
es un directorio: False


Archivo: "oop.ipynb",
es un directorio: False


Archivo: ".gitignore",
es un directorio: False


Archivo: "numeros-y-matematica.ipynb",
es un directorio: False


Archivo: "tips.ipynb",
es un directorio: False


Archivo: "encontrando-resolviendo-errores.ipynb",
es un directorio: False


Archivo: ".git",
es un directorio: True
```

```
Archivo: "datos",
es un directorio: True


Archivo: "logica-condicional-control-de-flujo.ipynb",
es un directorio: False


Archivo: "variables.ipynb",
es un directorio: False


Archivo: "strings.ipynb",
es un directorio: False


Archivo: "textos-llamadas.ipynb",
es un directorio: False


Archivo: "hola.txt",
es un directorio: False


Archivo: "juego-de-aventura.ipynb",
es un directorio: False


Archivo: "tuplas-listas-diccionarios.ipynb",
es un directorio: False


Archivo: ".python-version",
es un directorio: False


Archivo: "salida.txt",
es un directorio: False
```

[41]:
```
for archivo in os.listdir():
    print(f'Archivo: "{archivo}" \nes un archivo: {os.path.
↪isfile(archivo)}\n\n')  # es un archivo?
```

```
Archivo: "contenido.ipynb"
```

```
es un archivo: True


Archivo: ".vscode"
es un archivo: False


Archivo: "intro.ipynb"
es un archivo: True


Archivo: "errores.ipynb"
es un archivo: True


Archivo: "contenido.md"
es un archivo: True


Archivo: "funciones-y-ciclos.ipynb"
es un archivo: True


Archivo: "salida"
es un archivo: False


Archivo: "file-entrada-salida.ipynb"
es un archivo: True


Archivo: ".ipynb_checkpoints"
es un archivo: False


Archivo: "otros-temas"
es un archivo: True


Archivo: "oop.ipynb"
es un archivo: True


Archivo: ".gitignore"
es un archivo: True


Archivo: "numeros-y-matematica.ipynb"
```

```
es un archivo: True


Archivo: "tips.ipynb"
es un archivo: True


Archivo: "encontrando-resolviendo-errores.ipynb"
es un archivo: True


Archivo: ".git"
es un archivo: False


Archivo: "datos"
es un archivo: False


Archivo: "logica-condicional-control-de-flujo.ipynb"
es un archivo: True


Archivo: "variables.ipynb"
es un archivo: True


Archivo: "strings.ipynb"
es un archivo: True


Archivo: "textos-llamadas.ipynb"
es un archivo: True


Archivo: "hola.txt"
es un archivo: True


Archivo: "juego-de-aventura.ipynb"
es un archivo: True


Archivo: "tuplas-listas-diccionarios.ipynb"
es un archivo: True


Archivo: ".python-version"
```

```
es un archivo: True


Archivo: "salida.txt"
es un archivo: True
```

[40]:
```python
for archivo in os.listdir():
    print(f'Archivo: "{archivo}" \nexiste: {os.path.exists(archivo)}\n\n')  # el archivo existe?
```

```
Archivo: "contenido.ipynb"
existe: True


Archivo: ".vscode"
existe: True


Archivo: "intro.ipynb"
existe: True


Archivo: "errores.ipynb"
existe: True


Archivo: "contenido.md"
existe: True


Archivo: "funciones-y-ciclos.ipynb"
existe: True


Archivo: "salida"
existe: True


Archivo: "file-entrada-salida.ipynb"
existe: True


Archivo: ".ipynb_checkpoints"
existe: True
```

```
Archivo: "otros-temas"
existe: True


Archivo: "oop.ipynb"
existe: True


Archivo: ".gitignore"
existe: True


Archivo: "numeros-y-matematica.ipynb"
existe: True


Archivo: "tips.ipynb"
existe: True


Archivo: "encontrando-resolviendo-errores.ipynb"
existe: True


Archivo: ".git"
existe: True


Archivo: "datos"
existe: True


Archivo: "logica-condicional-control-de-flujo.ipynb"
existe: True


Archivo: "variables.ipynb"
existe: True


Archivo: "strings.ipynb"
existe: True


Archivo: "textos-llamadas.ipynb"
existe: True
```

```
Archivo: "hola.txt"
existe: True


Archivo: "juego-de-aventura.ipynb"
existe: True


Archivo: "tuplas-listas-diccionarios.ipynb"
existe: True


Archivo: ".python-version"
existe: True


Archivo: "salida.txt"
existe: True
```

3.4 Ejercicios

1. Imprime la ruta absoluta de todos los archivos y directorios en el directorio de `Documentos/` en su computador
2. Imprima la ruta absoluta de todos los archivos .txt en el directorio actual

4 Lea y Escriba Data CSV

Los archivos del dia a dia son un poco mas complicados que archivos simples de texto. Para modificar el contenido de estos archivo, necesitamos un poco mas de herramientas.

Una manera comun para guardar datos de texto es en archivos CSV, que por sus siglas en ingles significa Valores Separados por Comma, toda vez que cada entrada en una fila de datos es usualmente separada de otras entrada con una coma. Por ejemplo:

```
Nombre, Apellido, Edad
Juan, Perez, 40
Juana, Perez, 45
```

Cada linea representa una fila de datos, incluyendo la primera fila que representa el encabezamiento de la informacion. Cada entrada aparece en el mismo orden para cada fila, con cada entrada separada de otras con comas.

Python tiene un modulo `csv` que nos permite realizar operacions necesarias para la gestion de archivos CSV.

4.1 Leer un archivo CSV

[44]:
```
# leemos un archivo con csv.reader(archivo)
import csv
import os

archivo = 'datos/llamadas.csv'
with open(archivo, 'r') as datos:
    lector = csv.reader(datos)
    for registro in lector:
        print(registro)
```

```
['numero_saliente', 'numero_entrante', 'fecha_tiempo', 'tiempo_segundos']
['(473) 5373591', '(221) 3829872', '2019-08-23 07:13:28', '779']
['(712) 6079829', '(521) 9979466', '2019-06-02 19:01:04', '150']
['(170) 7207064', '(667) 9707152', '2019-02-18 11:22:21', '1425']
['(267) 6838416', '(704) 6053438', '2019-12-26 22:29:30', '3278']
['(202) 7159564', '(848) 5356715', '2019-11-11 14:06:47', '2823']
['(971) 4270187', '(312) 3476941', '2019-07-21 22:30:45', '2824']
['(688) 1872860', '(580) 6692170', '2019-01-05 20:40:15', '363']
['(527) 3643293', '(700) 6013130', '2019-03-21 10:25:15', '1090']
['(824) 3120489', '(736) 5219693', '2019-06-15 19:31:29', '2383']
['(135) 5879807', '(210) 4726824', '2019-12-11 06:37:28', '3289']
['(946) 9885969', '(967) 6260487', '2019-04-30 18:12:15', '266']
['(822) 1999029', '(394) 2159591', '2019-07-20 13:24:02', '3171']
['(214) 1831354', '(407) 4594421', '2019-10-22 18:27:53', '2987']
['(301) 9038508', '(117) 3599538', '2019-08-11 14:34:08', '472']
['(975) 2050968', '(225) 7340340', '2019-05-24 17:07:56', '1297']
['(532) 4461437', '(159) 6755397', '2019-07-27 09:02:02', '2548']
['(854) 2632368', '(865) 1092554', '2019-11-12 18:27:12', '256']
['(302) 7956136', '(427) 4230223', '2019-04-17 13:49:54', '360']
['(694) 4605593', '(423) 9644633', '2019-10-12 10:25:53', '3476']
['(361) 6243068', '(817) 9801242', '2019-01-13 16:55:39', '740']
['(832) 2674004', '(134) 4315303', '2019-10-07 07:16:17', '3135']
['(833) 8445033', '(191) 5366913', '2019-03-18 09:42:11', '2112']
['(823) 4146625', '(263) 8920846', '2019-03-17 16:10:28', '1635']
['(901) 2728567', '(997) 8431267', '2019-06-05 11:45:06', '2793']
['(695) 8465544', '(486) 6125527', '2019-08-19 14:22:47', '1563']
['(715) 6420894', '(828) 6640394', '2019-03-16 00:36:28', '1891']
['(600) 8596964', '(762) 7724562', '2019-12-19 15:44:12', '3157']
['(454) 4219619', '(432) 1223026', '2019-02-05 02:43:10', '1050']
['(699) 8211331', '(123) 8577076', '2019-01-26 02:35:52', '2547']
['(502) 2393708', '(748) 6208057', '2019-04-28 12:23:38', '3047']
['(319) 7353522', '(588) 9583209', '2019-01-02 05:17:31', '3584']
['(519) 2780596', '(359) 2449867', '2019-02-05 23:35:17', '1436']
['(439) 1787485', '(802) 8632114', '2019-01-03 02:05:09', '2878']
['(611) 2732835', '(605) 7128788', '2019-05-06 22:24:59', '636']
['(481) 4216326', '(288) 7103116', '2019-05-06 01:35:05', '2339']
```

['(819) 2841562', '(651) 9421311', '2019-12-05 16:05:32', '3449']
['(561) 7890310', '(487) 1704598', '2019-08-09 16:54:44', '1187']
['(205) 8012873', '(348) 6088588', '2019-04-09 18:36:15', '3194']
['(656) 2596247', '(645) 3744183', '2019-12-18 21:38:31', '2428']
['(784) 1502772', '(732) 5122798', '2019-08-06 05:30:25', '983']
['(187) 7805812', '(831) 1984447', '2019-10-09 02:03:01', '3577']
['(404) 9897959', '(810) 9464280', '2019-03-31 01:10:37', '1188']
['(320) 9017964', '(105) 7031191', '2019-03-19 16:47:31', '2009']
['(441) 9421898', '(352) 2239520', '2019-10-09 07:19:24', '3024']
['(289) 3939816', '(897) 5873250', '2019-06-24 08:18:43', '2230']
['(268) 8129614', '(109) 5020811', '2019-10-18 20:45:40', '1559']
['(530) 2679399', '(929) 7641354', '2019-05-03 08:45:23', '128']
['(655) 5001076', '(216) 9767752', '2019-11-13 19:03:39', '171']
['(883) 8587195', '(449) 7773819', '2019-06-29 14:01:03', '1818']
['(815) 4795720', '(312) 1327386', '2019-02-11 06:32:37', '3119']
['(197) 3603866', '(412) 8148714', '2019-06-13 18:10:46', '595']
['(911) 1379852', '(804) 6251709', '2019-03-06 21:18:13', '2234']
['(795) 2762776', '(661) 8174095', '2019-02-18 06:09:36', '2300']
['(501) 1466641', '(602) 3090356', '2019-08-01 05:07:26', '734']
['(154) 9559400', '(632) 8185869', '2019-05-16 08:59:38', '3506']
['(639) 8951743', '(742) 1588632', '2019-09-25 00:11:59', '73']
['(792) 8079631', '(598) 3917497', '2019-06-14 15:26:01', '1908']
['(755) 2227215', '(235) 5321774', '2019-06-19 23:02:17', '1065']
['(712) 6065475', '(794) 3858022', '2019-10-22 20:49:53', '3485']
['(680) 3236045', '(804) 9903489', '2019-09-10 16:34:14', '2922']
['(148) 6443267', '(169) 8934639', '2019-05-22 10:47:23', '129']
['(790) 6530469', '(814) 3215137', '2019-10-25 13:33:34', '2664']
['(202) 1610658', '(607) 3944087', '2019-04-28 23:54:28', '1569']
['(262) 4164407', '(399) 5230169', '2019-02-19 03:10:29', '450']
['(262) 6287235', '(522) 8488463', '2019-06-01 19:05:29', '3383']
['(304) 7491008', '(244) 5322157', '2019-08-10 13:52:18', '3064']
['(615) 3509514', '(708) 4135633', '2019-05-25 05:54:32', '147']
['(459) 3930189', '(149) 4330839', '2019-01-08 10:47:42', '3140']
['(855) 2282632', '(666) 2793624', '2019-11-24 23:08:45', '1022']
['(476) 5233902', '(820) 5595528', '2019-01-31 21:41:20', '1837']
['(537) 6546615', '(612) 2202646', '2019-04-04 19:51:31', '2036']
['(541) 4800549', '(138) 3724141', '2019-02-04 06:45:02', '2469']
['(384) 8739072', '(941) 6726850', '2019-04-19 02:27:44', '3105']
['(147) 7291940', '(326) 5393948', '2019-03-05 17:08:14', '2586']
['(270) 2354861', '(273) 7690535', '2019-09-02 16:24:08', '3249']
['(636) 1133234', '(462) 1957853', '2019-09-13 12:22:37', '1939']
['(570) 6207945', '(581) 2812391', '2019-04-04 03:03:56', '1225']
['(291) 5185327', '(531) 9281928', '2019-10-22 11:18:33', '2989']
['(243) 3167219', '(570) 5034926', '2019-04-13 11:54:36', '789']
['(542) 4546760', '(567) 2828533', '2019-05-03 20:25:18', '1865']
['(249) 7029277', '(295) 8985580', '2019-09-23 06:24:47', '319']
['(916) 5096404', '(376) 2884045', '2019-04-24 08:30:37', '246']
['(996) 9736898', '(969) 2964664', '2019-09-02 15:31:42', '2342']

```
['(207) 4248725', '(456) 8645080', '2019-11-15 20:30:18', '630']
['(729) 4815293', '(763) 9893406', '2019-08-22 07:42:54', '2279']
['(188) 3501714', '(464) 3997111', '2019-01-14 04:57:34', '121']
['(534) 4568556', '(792) 9326352', '2019-06-09 05:23:38', '1046']
['(892) 1686376', '(249) 7615536', '2019-03-02 22:11:17', '2444']
['(310) 6945801', '(164) 9416529', '2019-04-20 07:44:28', '1683']
['(741) 8134173', '(712) 6154466', '2019-01-12 02:12:28', '210']
['(780) 2506688', '(246) 9160852', '2019-08-10 12:18:32', '512']
['(677) 3634048', '(650) 1143542', '2019-03-09 14:08:49', '2166']
['(770) 5974145', '(270) 5953021', '2019-01-20 09:19:34', '608']
['(251) 5430038', '(570) 1985179', '2019-03-20 13:05:08', '3447']
['(823) 1821952', '(835) 1658609', '2019-08-10 07:35:24', '2504']
['(302) 9131957', '(738) 3350982', '2019-09-30 18:05:45', '1176']
['(787) 2744903', '(435) 1451178', '2019-07-27 09:08:10', '989']
['(723) 1155427', '(810) 5853913', '2019-04-09 19:24:35', '2467']
['(368) 3851791', '(631) 8084245', '2019-07-31 18:03:27', '556']
['(285) 1009067', '(219) 1745803', '2019-11-25 18:22:30', '3054']
```

4.2 Escribir a un archivo

[11]:
```python
# escribimos un archivo con csv.writer(archivo)
import csv
import os

archivo = 'salida/ejemplo.csv'

# os.mkdir('salida')

nombres = [
    ['Nombre', 'Apellido'],
    ['Juan', 'Perez'],
    ['Juana', 'Perez']
]

with open(archivo, 'w') as salida:
    escritor = csv.writer(salida)
    escritor.writerows(nombres)

with open(archivo, 'r') as entrada:
    lector = csv.reader(entrada)
    print(list(lector))
```

```
[['Nombre', 'Apellido'], ['Juan', 'Perez'], ['Juana', 'Perez']]
```

4.3 Ejercicios

1. Escriba un script que escriba un archivo csv con informacion inventada, puede ser cualquier cosa, pero que tenga columnas, e.g. Nombre, Apellido, Edad, etc.

2. Escriba un script que lea el archivo escrito e imprima cada fila del archivo

5 Reto: Puntos de Acceso de la Red Nacional de Internet por Provincia

Escriba un script que lea el archivo csv `datos/rni-puntos-de-acceso.csv` (se le entregará) que contiene informacion de todos los puntos de acceso de la red nacional de internet de Panama, y escriba un archivo csv nuevo `datos/rni-pda-por-provincia.csv`, que contiene lo siguiente: 1. Provincia y Puntos de Acceso como encabezado 2. El nombre de la provincia, y el numero total de Puntos de Acceso como fila

[45]:
```
# un ejemplo practico

import csv
import os

archivo = 'datos/rni-puntos-de-acceso.csv'
with open(archivo, 'r', encoding='latin-1') as entrada, open('salida/
 ↪rni-pda-chiriqui.csv', 'w') as salida:
    lector = csv.reader(entrada)  # para leer
    escritor = csv.writer(salida)  # para escribir
    for registro in lector:
        provincia = registro[3]
        if provincia.startswith('Chi'):
            escritor.writerow(registro)

with open(archivo, 'r', encoding='latin-1') as entrada:
    lector = csv.reader(entrada)
    for registro in list(lector)[:5]:
        print(registro)
```

```
['Regi¢n', 'PA', 'Nombre', 'Provincia', 'Distrito', 'Corregimiento', 'Tipo UM',
'Latitude', 'Longitude', 'Fecha de Activaci¢n']
['1', '1', 'Colegio Rogelio Josu\x82 Ibarra', 'Bocas del Toro', 'Bocas del
Toro', 'Bocas del Toro', 'FO', '9.340654', '-82.242499', '12/07/17']
['1', '3', 'Escuela Rep£blica de Nicaragua', 'Bocas del Toro', 'Bocas del Toro',
'Bocas del Toro', 'FO', '9.338938', '-82.242668', '12/07/17']
['1', '4', 'Gobernaci¢n', 'Bocas del Toro', 'Bocas del Toro', 'Bocas del Toro',
'FO', '9.297858', '-82.41136', '23/11/17']
['1', '5', 'Parque Sim¢n Bol¡var', 'Bocas del Toro', 'Bocas del Toro', 'Bocas
del Toro', 'FO', '9.340183', '-82.240631', '12/07/17']
```

6 Resumen

Hemos aprendido como trabajar con archivos en Python. Ahora podemos abrir un archivo para leer y escribir con la funcion `open()`, que toma dos argumentos: un string que contiene la ruta al archivo, y un string que contiene el modo de abrir el archivo, e.g. `r` para leer y `w` para escribir.

Tambien vimos como trabajar con archivos utilizando el modulo `os`, como crear y borrar directorios con `os.mkdir()` y `os.rmdir()`.

Finalmente, aprendimos como trabajar con archivos CSV, que es un tipo de archivo estandar para guardar datos por categoria. Vimos como leer data en un CSV con `csv.reader()`, que retorna cada fila del CSV como una lista, y vimos como escribir filas de datos a un CSV utilizando `csv.writer()`.