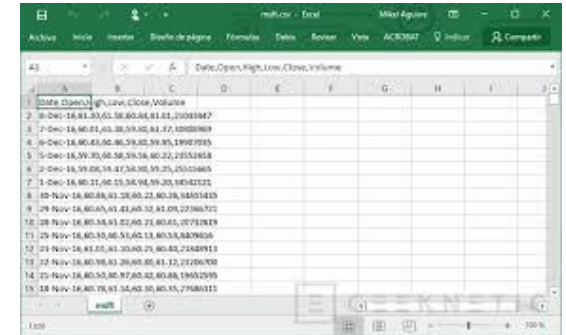


# Manejo Ficheros

- análisis de texto con ER
- cvsreader



# Practica P04

Accede a esta web ( <https://www.tutorialpython.com/>) escoge y copia 4 párrafos de texto y pégalos a un fichero que se llame : [texto2.txt](#).

Crea un programa que lea dicho archivo y calcule:

- Número total de líneas:
- Número de palabras por línea :

Mira la siguiente diapositiva Info-extra y aprende como seleccionar palabras de un texto usando expresiones regulares.

# Info extra : palabras que ...

*Una **expresión regular** es un cadena de caracteres especial que define una búsqueda de patrones.*

*Puedes pensar en las expresiones regulares como un tipo de comodín con asteriscos.*

**/hola/** : Buscar el texto explícito hola.

**/a[nr]t/** : Busca un patrón el cual su primera letra sea la a y su última letra sea la t, y entre dichas letras contenga la letra n o la r.

Por lo tanto, casos de éxito para este patrón, serían las palabras inglesas ant y art.

**/ca[tbr]/** : Busca palabras que empiecen con ca y terminen con alguno de estos caracteres tbr

El módulo `re` de Python incluye funciones para trabajar con expresiones regulares.

`re.search`: busca un patrón en otra cadena:

`re.match`: busca un patrón al principio de otra cadena

`re.findall`: busca todas las coincidencias en una cadena



La `findall()` función devuelve una lista que contiene todas las coincidencias.

Imprime una lista de todos los partidos:

```
import re

str = "The rain in Spain"
x = re.findall("ai", str)
print(x)
```

Ejemplo

Identifiers	Modifiers	White space characters	Escape required
\d= any number (a digit)	\d represents a digit.Ex: \d{1,5} it will declare digit between 1,5 like 424,444,545 etc.	\n = new line	. + * ? [] \$ ^ () {}   \ 
\s = space (tab,space,newline etc.)	? = matches 0 or 1	\t =tab	
\S= anything but a space	* = 0 or more	\e = escape	
\w = letters ( Match alphanumeric character, including "_")	\$ match end of a string	\r = carriage return	
\W =anything but letters ( Matches a non-alphanumeric character excluding "_")	^ match start of a string	\f= form feed	
. = anything but letters (periods)	matches either or x/y	----- ---	
\b = any character except for new line	[] = range or "variance"		
\.	{x} = this amount of preceding code		

```
fiche = open('text.txt', 'r')      # abre para leer
texto = fiche.read().lower()        # lee contenido y lo pasa a minúsculas
print("texto base", texto)          # imprime

# Busca todas las palabras de 4 a 7 letras.
lista = re.findall(r'[a-z]{4,7}', texto, re.MULTILINE)
print ("total palabras de 4 a 7 letras: ", len(lista))

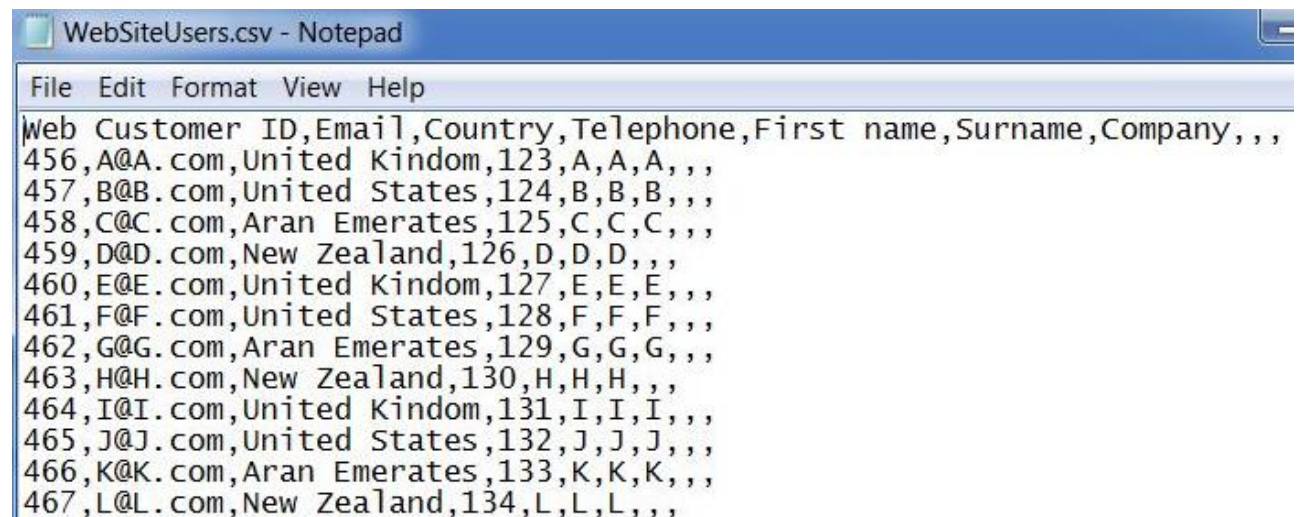
print ( "long.\t palabra")
for item in lista:
    print (len(item), '\t', item)
```

## **Usando re.findall para el buscar en todo el texto multilínea**

El módulo re.findall () se usa cuando se desea recorrer en iteración las líneas del archivo, devolverá una lista de todas las coincidencias en un solo paso

# ¿ Qué es un fichero csv ?

- Es un fichero de texto en el cual la información está estructurada en campos separados por un delimitador : coma ( ; tab, etc).
- Todos los registros siguen la misma separación de campos
- Puede haber una cabecera opcional
- Algunos campos pueden separarse entre comillas (simples o dobles)



```
WebSiteUsers.csv - Notepad
File Edit Format View Help
Web Customer ID,Email,Country,Telephone,First name,Surname,Company,,
456,A@A.com,United Kindom,123,A,A,A,,
457,B@B.com,United States,124,B,B,B,,
458,C@C.com,Aran Emerates,125,C,C,C,,
459,D@D.com,New Zealand,126,D,D,D,,
460,E@E.com,United Kindom,127,E,E,E,,
461,F@F.com,United States,128,F,F,F,,
462,G@G.com,Aran Emerates,129,G,G,G,,
463,H@H.com,New Zealand,130,H,H,H,,
464,I@I.com,United Kindom,131,I,I,I,,
465,J@J.com,United States,132,J,J,J,,
466,K@K.com,Aran Emerates,133,K,K,K,,
467,L@L.com,New Zealand,134,L,L,L,,
```

# Python : como manejar csv

```
In [2]: import csv

with open("exampleSet.csv") as fProductos :

    readPC = csv.reader(fProductos, delimiter = ",")
    for row in readPC :
        print (row)
```

Podemos leer ficheros en los distintos formatos **csv**, ya que el separador se especifica.

Puedes introducir y probar este programa ?

exampleSet.csv

'1990',64.326',65.897',64.67'
'1991',74.326',75.897',74.67'
'1992',84.326',75.897',74.67'
'1993',94.326',75.897',74.67'

Para ello crea un **csv** o salva estos datos en un fichero



```
import csv

with open("exampleSet.csv") as fProductos :
    readPC = csv.reader(fProductos, delimiter = ",", quotechar="'")
    for row in readPC :
        print (row)
```

```
['1990', '64.326', '65.897', '64.67']
['1991', '74.326', '75.897', '74.67']
['1992', '84.326', '75.897', '74.67']
['1993', '94.326', '75.897', '74.67']
```

```
from io import open
```

```
fichero = open("exampleSet.csv", "r")
lineas = fichero.readlines()
```

```
for linea in lineas :
    limpia = linea.replace("'", " ")
    print (limpia)
```

```
1990 , 64.326 , 65.897 , 64.67
1991 , 74.326 , 75.897 , 74.67
1992 , 84.326 , 75.897 , 74.67
1993 , 94.326 , 75.897 , 74.67
```

Graba un fichero de empleados con el nombre el departamento y el mes de nacimiento.

Crea una lista con los nombres de los campos (**fieldnames**) y crea un diccionario de escritura (**DictWriter**) para pasar los datos en dicho formato.

```
import csv

with open('employee_file2.csv', mode='w') as csv_file:
    fieldnames = ['emp_name', 'dept', 'birth_month']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerow({'emp_name': 'John Smith', \
                     'dept': 'Accounting', \
                     'birth_month': 'November'})
    writer.writerow({'emp_name': 'Erica Meyers', \
                     'dept': 'IT', \
                     'birth_month': 'March'})
```

Lee el mismo fichero, usa la cabecera para crear un Diccionario de Lectura (DictReader) y entonces lo que lee es un diccionario donde las claves son los nombres de los campos.

```
import csv

with open('employee_file2.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    line_count = 0

    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            print("\temp_name \tdept \t\tbirth_month")
            line_count += 1

print(f'\t{row["emp_name"]}\t{row["dept"]}\t\t{row["birth_month"]}.')
    line_count += 1
print(f'Processed {line_count} lines.')
```

# CSV más a fondo

<https://code.tutsplus.com/es/tutorials/how-to-read-and-write-csv-files-in-python--cms-29907>

He aquí la primera dificultad. Trabajar con CSV se realiza, exclusivamente, en Unicode, que es el formato de las cadenas de caracteres por defecto de la rama 3 de Python. Puede que los archivos que queramos leer no sean archivos Unicode.

Su lectura produce un error:

```
>>> with open('comaut.txt', 'r') as f:
...     datas = csv.reader(f)
...     next(datas)
...
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
  File "/usr/lib/python3.2/codecs.py", line 300, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf8' codec can't decode byte 0xe9 in
position 153: invalid continuation byte
```

La solución no consiste en abrir el archivo en binario y trabajar con sus bytes, incluso aunque se piense en hacer una conversión. Los archivos CSV son archivos de texto y deben cargarse como tales:

De modo que la solución consiste en abrir el archivo en modo texto, indicando su codificación (la codificación utilizada por Python es, obligatoriamente, Unicode):

```
>>> with open('comaut.txt', 'r', encoding='latin1') as f:
...     datas = csv.reader(f)
...     next(datas)
...
['Cod.Comaut\tComAut']
```

A partir de ahora, seremos capaces de leer cualquier archivo CSV.

La representación pythónica de un archivo CSV no contiene encabezados y es, simplemente, una lista de listas (lista de filas de la tabla, que se presentan en forma de lista), pues la lista de Python dispone de una relación de orden y permite conservar el orden de las filas y, dentro de cada fila, el orden de sus columnas.

Eliminaremos el encabezado del archivo simplemente no volviendo al inicio del archivo una vez determinado el dialecto, y construiremos dicha representación:

```
>>> with open('comaut.txt', 'r', encoding='latin1') as f:
...     dialecto = csv.Sniffer().sniff(f.readline())
...     datos = list(csv.reader(f, dialect=dialecto))
... 
```

La operación se lleva a cabo en tan solo tres líneas de código.

# Práctica P05

- Encuentra en internet una tabla de datos (por ejemplo (<http://www.bcn.cat/estadistica/catala/index.htm> )).
- Pégallo en Excell o LibreOffice, arregla un poco el formato y sávalo en formato csv, con separador “;”.
- Realiza un programa para leer dichos datos y mostrarlos por pantalla con print.

Dte.	Barri	;Població	;Domicilis	;"Ocupació mitjana per domicili "
BARCELONA		;1666530	;664476	;2,51
1	1. el Raval	;48263	;17384	;2,78
1	2. el Barri Gòtic	;21715	;6617	;3,28
1	3. la Barceloneta	;15112	;6994	;2,16
1	4. Sant Pere, Santa Caterina i la Ribera	;23241	;10208	;2,28
2	5. el Fort Pienc	;33369	;12839	;2,6
2	6. la Sagrada Família	;52245	;21968	;2,38
2	7. la Dreta de l'Eixample	;44325	;18516	;2,39
2	8. l'Antiga Esquerra de l'Eixample	;43228	;18183	;2,38
2	9. la Nova Esquerra de l'Eixample	;58621	;24719	;2,37
2	10. Sant Antoni	;38906	;16301	;2,39
3	11. el Poble Sec - AEI Parc Montjuïc	;40157	;15961	;2,52
3	12. la Marina del Prat Vermell - AEI Zona Franca	;1227	;492	;2,49
3	13. la Marina de Port	;31352	;12035	;2,61

```
import csv
```

```
with open("dat\\inev_barris.csv") as fbarris :
    readPC = csv.reader(fbarris, delimiter = ";")
    for row in readPC :
        if row[1].isdigit():
            print (f" {row[0]:30.30}      {int(row[1]):7d}
                    {int(row[2]):7d}    {row[3]:7.5}")
        else :
            print (f" {row[0]:30.30}      {row[1]:7.7}
                    {row[2]:7.7}    {row[3]:7}")
```

**Avanzado :**

**Carga con pandas**