Exercicis Resolts de Fitxers

Author: Sebas Vila-Marta
Date: 30 nov 2010

1 Comarques

1.1 Enunciat

El fitxer *comarques.txt* conté un llistat de les comarques de Catalunya. És un fitxer de text i cada comarca ocupa una línia. Es vol un programa que consulti aquest fitxer i escrigui per la pantalla el nom de les comarques que comencen per 'B'.

1.2 Solució

En aquest cas la soluió és immediata: obrim el fitxer en mode lectura, recorrem les seves línies (Noms de comarca) i escrivim per la pantalla aquelles que comencen per 'B'.

```
# -*- encoding: utf-8 -*-

if __name__ == "__main__":
    fitxer = open('comarques.txt', 'r')
    comarca = fitxer.readline()
    while comarca != "":
        if comarca[0] == 'B':
            print comarca,
        comarca = fitxer.readline()
    fitxer.close()
```

Hi ha alguns aspectes interessants a comentar:

- Fixeu-vos que la funció *readline* retorna un string buit quan arriba al final del fitxer. Per tant, aquesta és la condició que fem servir per acabar el *while*. Considerant que, com sabeu, un string buit interpretat coma booleà equival a *False*, també és podria haver escrit:

  ```
  ...
  comarca = fitxer.readline()
  while comarca:
     if comarca[0] == 'B':
        ...
  ```

- En el moment d'escriure per pantalla el nom de les comarques la sentència *print* acaba en una coma. Aquesta com indica que no s'ha de saltar línia. Malgrat això, si proveu el programa veureu que el resultat s'escriu en columna i, per tant, es salten línies!

  No hi ha cap cosa estranya en aquest comportament. El que succeeix simplement és que les línies llegides del fitxer **ja incorporen el salt de línia**. Efectivament, en el fitxer *comarques.txt* la línia "Bages", per exemple, en realitat conté els caràcters "Bages\n". La funció *readline* llegeix **tots** els caràcters d'una línia i, per tant, també el salt de línia ("\n").

2 Participació

2.1 Enunciat

El fitxer de text *municipals.txt*, obtingut de l'IDESCAT, conté dades electorals de les eleccions municipals del 27 de maig del 2007 comarca per comarca.

Cada línia està composada per les següents dades (en aquest ordre):

1. Nom de la comarca (string)
2. Nombre d'electors (enter)
3. Nombre de votants (enter)
4. Vots al PSC (enter)
5. Vots a CiU (enter)
6. Vots a ERC (enter)
7. Vots a PP (enter)
8. Vots a IC-V (enter)
9. Vots a altres partits (enter)

En una línia, cada dada se separa de l'anterior amb una coma.

Es demana que dissenyeu i implementeu un programa que escriu per pantalla una taula amb el nom de cada comarca i el percentatge de participació.

A continuació teniu una taula amb els continguts del fitxer *municipals.txt* per tal que en tingueu referència.

Resultat eleccions municipals 2007. Fitxer municipals.txt

Comarca	Electors	Votants	PSC	CIU	ERC	PP	ICV	ALT
Alt Camp	31942	20903	6637	8344	2566	668	573	1365
Alt Emporda	87322	54987	14105	19366	10824	3246	3161	2638
Alt Penedes	73002	47184	16729	15302	5096	2769	2221	3351
Alt Urgell	19572	10849	3430	4030	1773	528	473	213
Alta Ribagorca	3112	2179	824	820	149	34	92	199
Anoia	84723	49213	17702	13847	9122	2848	1661	2276
Bages	137295	79291	20272	25708	14847	3201	4637	7264
Baix Camp	126805	70127	21564	19692	8287	6863	3454	7865
Baix Ebre	57273	38435	10354	13107	7908	2741	1186	1925
Baix Emporda	88518	53640	16592	13549	7750	2023	4893	7066
Baix Llobregat	576430	292119	113335	40776	26280	31185	48390	22985
Baix Penedes	65787	38380	11474	12704	3104	2697	1538	5906
Barcelones	1680772	818371	274041	178803	63784	129640	80830	57211
Bergueda	33031	21402	6018	8136	2694	613	1277	1264
Cerdanya	13072	8469	1050	2966	3201	212	346	486
Conca de Barbera	16021	11018	2084	3732	2409	104	115	1995

Garraf	98165	50574	17252	11665	3132	4631	4809	7386
Garrigues	16184	11726	2733	4717	3190	373	390	3
Garrotxa	40150	25748	7911	10578	3412	775	1222	800
Girones	116285	66553	19145	19825	11563	3895	4669	4756
Maresme	307835	164719	45177	44873	19455	16550	13237	19179
Montsia	48599	33774	9504	10258	6380	2788	2267	1408
Noguera	30140	21467	5770	8025	4103	870	1122	714
Osona	107742	69651	14446	21968	14284	1021	4455	10037
Pallars Jussa	11072	7360	2817	2516	1546	145	167	102
Pallars Sobira	5851	4520	1457	1819	625	91	220	142
Pla d'Urgell	25587	19182	4176	8210	4573	477	261	807
Pla de l'Estany	20965	14172	1126	5714	5280	283	425	673
Priorat	8074	6438	1338	1864	2524	58	0	337
Ribera d'Ebre	18321	14059	4039	5094	3025	288	519	627
Ripolles	21868	15241	3671	5860	3518	213	824	454
Segarra	14751	10647	1175	3265	2959	348	389	2209
Segria	141560	82239	32438	21785	9119	7248	4126	4675
Selva	109106	63994	13116	21629	9928	2336	5399	9080
Solsones	9981	7055	1010	3159	1620	181	0	750
Tarragones	163047	89723	29593	24425	7626	10408	4887	10487
Terra Alta	10332	8284	2530	3301	878	435	0	898
Urgell	25876	17418	3651	6388	3822	775	652	1392
Val d'Aran	6959	4896	2276	2009	0	298	115	37
Valles Occidental	630276	308755	110082	60137	24529	28410	39039	36730
Valles Oriental	278382	152854	52129	35368	18038	10926	15058	15421

2.2 Solució

Procedim fent un recorregut per les línies del fitxer. Per cada línia (comarca) obtenim les dades del nombre de votants i d'electors i calculem la participació. El resultat és el que segueix:

```
# -*- encoding: utf-8 -*-

if __name__ == "__main__":
    f = open('municipals.txt', 'r')
    comarca = f.readline()
    while comarca:
        d = comarca.split(",")
        nomc  = d[0]
        elect = float(d[1])
        vots  = float(d[2])
        print "%-20s %4.2f%%" % (nomc, vots*100/elect)
        comarca = f.readline()
    f.close()
```

Noteu els següents detalls:

- Donada una línia, els diferents camps de la línia es troben separats per comes. D'això se'n diu format CSV (Coma Separated Value) i és un format en que tots els fulls de càlcul saben escriure. Per trencar la línia en els seus camps, usem la funció *split* dels strings. El resultat és una llista de strings que corresponen a cadascún dels camps.
- Les dades numèriques, com ara el nombre de votants, les tenim després de l'split en forma de string. Per tant cal convertir-les a *float* en el nostre cas. Això ho fem usant la funció *float()*.
- per últim la línia següent:

```
print "%-20s %4.2f%%" % (nomc, vots*100/elect)
```

  s'encarrega d'escriure ben tabulat el resultat. Fixeu-vos que usa un format en que hi ha dos "forats" a omplir. El primer, "%-20s", indica que hi ha d'anar un string i que ha d'ocupar sempre 20 columnes alineant per l'esquerra. El segon, "%4.2f", indica que hi ha d'anar un real que ha de tenir 4 xifres, 2 de les quals han d'estar a darrera la coma decimal.

3 Comarca més militant

3.1 Enunciat

Considerant el mateix fitxer de l'exercici anterior, dissenyeu i implementeu un programa que obtingui un acronim de partit de la línia de comanda i escrigui el nom de la comarca en que, percentualment, aquest partit ha estat més votat i el percentatge de vots sobre el nombre de votants. Useu com acrònims de partit els següents: PSC, CIU, ERC, PP, ICV, ALT.

3.2 Solució

Construïrem el programa que es demana a base d'aproximacions successives. Primer començarem per un programa que, donat un acrònim de partit, ens llista el percentatge de votants que ha tingut aquest partit en cada comarca. La idea essencial és fer un recorregut de totes les comarques i escriure els percentatges corresponents al partit indicat:

```
# -*- encoding: utf-8 -*-
import sys

if __name__ == "__main__":

    # Acronims dels partits en l'ordre del fitxer
    acronims = ['PSC', 'CIU', 'ERC', 'PP', 'ICV', 'ALT']

    # Obtenim el partit i el número de columna
    partit = sys.argv[1]
    i_partit = acronims.index(partit) + 3

    # Escrivim el resultat
    f = open('municipals.txt', 'r')
    comarca = f.readline()
    while comarca:
        d = comarca.split(",")
        nomc  = d[0]
        votants  = float(d[2])
        votspar  = float(d[i_partit])
```

```
        print "%-20s %4.2f%%" % (nomc, votspar*100/votants)
        comarca = f.readline()
    f.close()
```

Fixeu-vos en com es tracta l'acrònim del partit: L'acrònim, en realitat, només en serveix per localitzar la columna escaient de les dades. Atès que sabem com estan disposades les dades en cada línia del fitxer, sabem quina columna ocupen les dades de cada partit. Per exemple, les dades de 'CIU' ocupen la columna 4. Les línies següents fan la feina:

```
...
# Acronims dels partits en l'ordre del fitxer
acronims = ['PSC', 'CIU', 'ERC', 'PP', 'ICV', 'ALT']

# Obtenim el partit i el número de columna
partit = sys.argv[1]
i_partit = acronims.index(partit) + 3
...
```

fixeu-vos que primer definim una llista que conté els acrònims en el mateix ordre que estan les dades en el fitxer. Això ens permet, donat un acronim *partit* cercar-lo en la llista i obtenir la seva posició dins la llista amb la funció *index*. Com que el primer partit, 'PSC', es troba en la columna 3, cal sumar aquest desplaçament per acabar obtenint *i_partit*, la posició de les dades del partit que ens interessa en cada línia del fitxer. Amb aquesta dada podem accedir al nombre de vots del partit en qüestió fent:

```
...
votspar  = float(d[i_partit])
...
```

Ara ens queda modificar el programa anterior per tal que, en comptes de tabular totes les comarques, determini aquella amb el major percentatge de vots i escrigui les dades per aquesta comarca. La solució consisteix a trobar un màxim. Per tant procedim de la manera habitual per calcular màxims i obtenim la següent versió:

```
# -*- encoding: utf-8 -*-
import sys


def obte_percentatge(l, p):
    """
    Donat un string que correspon a una línia del
    fitxer de dades i un acrònim de partit, en retorna una
    tupla formada pel nom de la comarca i el percentatge
    de vots corresponents al partit.
    """
    acronims = ['PSC', 'CIU', 'ERC', 'PP', 'ICV', 'ALT']
    i_partit = acronims.index(p) + 3

    d = l.split(",")
    nomc  = d[0]
    votants  = float(d[2])
    votspar  = float(d[i_partit])
    return (nomc, votspar*100/votants)
```

```
if __name__ == "__main__":
    # Obtenim el partit i el número de columna
    partit = sys.argv[1]

    # Escrivim el resultat
    f = open('municipals.txt', 'r')
    maxim_vot = ('', 0)
    comarca = f.readline()
    while comarca:
        c,p = obte_percentatge(comarca, partit)
        if p > maxim_vot[1]:
            maxim_vot = (c,p)
        comarca = f.readline()
    print "%-20s %4.2f%%" % maxim_vot
    f.close()
```

Fixeu-vos que:

- En la nova versió hem separat part del càlcul del programa principal en una funció anomenada *obte_percentatge*. Aquesta funció extreu d'una línia del fitxer de dades el percentatge de vots d'un partit determinat així com el nom de la comarca.

4 Matriu de percentatges

4.1 Enunciat

Seguint amb el mateix fitxer de dades del problema anterior, ara es vol un programa que llegeixi les dades del fitxer *municipals.txt* i generi un nou fitxer de text anomenat *percentatges.csv* que contingui una taula on cada fila contingui:

1. El nom de la comarca
2. El percentatge de vot de la comarca (sobre nombre de votants) del PSC
3. El percentatge de vot de la comarca (sobre nombre de votants) del CIU
4. El percentatge de vot de la comarca (sobre nombre de votants) del ERC
5. El percentatge de vot de la comarca (sobre nombre de votants) del PP
6. El percentatge de vot de la comarca (sobre nombre de votants) del ICV
7. El percentatge de vot de la comarca (sobre nombre de votants) d'altres partits.

En el fitxer resultat escriviu les dades de cada fila separades per comes i sense espais entre elles: és a dir en format CSV.

4.2 Solució

La solució és força similar a la de l'anterior exercici:

```
# -*- encoding: utf-8 -*-
import sys


def obte_percentatge(l):
    """
    Donat un string que correspon a una línia del fitxer de dades,
```

```
    retorna una tupla formada pel nom de la comarca i el percentatge
    de vots corresponents a cada partit en l'ordre original.
    """
    d = l.split(",")
    votants  = float(d[2])
    perc = [d[0]]
    for p in range(3,9):
        perc.append(float(d[p])*100/votants)
    return tuple(perc)


if __name__ == "__main__":
    f = open('municipals.txt', 'r')
    g = open('percentatges.csv', 'w')
    maxim_vot = ('', 0)
    comarca = f.readline()
    while comarca:
        p = obte_percentatge(comarca)
        g.write("%s,%f,%f,%f,%f,%f,%f\n" % p)
        comarca = f.readline()
    f.close()
    g.close()
```

Note com en aquesta solució s'obre dos fitxers simultàniament: *f* i *g*. Un en mode lectura i l'altre en mode escriptura. El procés consisteix a recórrer el primer i, simultàniament, anar escrivint els resultats en el segon.

El fitxer obtingut, *percentatges.csv*, és un fitxer que pot ser llegit per un full de càlcul directament. Proveu de llegir-lo amb Openoffice.org a veure si us en sortiu!

5 Generador d'esquelets

5.1 Enunciat

Cada vegada que un programador comença l'edició d'un nou mòdul Python fa una sèrie de feines repetitives fins que comença a escriure codi. Particularment, escriu en el fitxer una sèrie de comentaris i elements que són comuns a tots els mòduls de Python.

En aquest context , volem dissenyar i implementar una aplicació que faciliti la generació d'esquelets de mòduls Python. Aquesta apicació ha de ser tal que, si executem des del terminal la comanda:

```
$ python cm.py matriu
```

Es generi un fitxer Python de nom *matriu.py* amb el següent contingut:

```
# -*- encoding: utf-8 -*-
"""
Mòdul matriu
============

:Author:  Lisbeth Salander
:Author:  Mikael Blomkvist

"""
```

```
if __name == "__main__":
```

Els noms dels autors s'han d'obtenir d'un fitxer de configuració. Aquest fitxer és un fitxer de text anomenat *autors.dat* que conté un nom d'autor per línia. En l'exemple anterior, el fitxer contenia:

```
Lisbeth Salander
Mikael Blomkvist
```

i per això en el mòdul creat apareixen com a autores aquestes dues persones.

5.2 Solució

En aquest cas us deixem una solució íntegra sense detallar com s'hi ha arribat. Fixeu-vos que a la funció *obte_autors*, allí on diu:

```
for a in fa:
   llista_a.append(a.strip())
```

cal observar un parell de detalls. El primer és l'ús de l'iterador *for* per recórrer les línies d'un fitxer. En efecte, podem aplicar l'iterador *for* a un fitxer com *fa* obert en mode lectura. L'efecte és que l'iterador recorre totes i cadascuna de les línies del fitxer i les va assignant a la variable lliure, en aquest cas *a*. El segon detall es refereix a l'ús de l'operació *strip* dels strings per eliminar els blancs del principi i final d'una línia. Cal adonar-se de que les línies llegides del fitxer inclouen el caràcter salt de línia al final. La funció *strip* fa desapareixer aquest caràcter.

```
# -*- encoding: utf-8 -*-
"""
python cm.py [-m] <nom_modul>
"""
import sys


def escriure_titol(g,t,nivell):
    """
    Escriu en el fitxer `g` el títol `t` de nivell `nivell`.  En
    nivell indica el tipus de subratllat que s'usarà. El nivell 1 usa
    '=' i el nivell 2 usa '-'.

    g: fitxer obert en mode escriptura
    t: el titol nivell:
    nivell: (1->titol, 2->subtitol)
    """
    if nivell == 1:
        g.write(t + '\n')
        g.write("="*len(t)+'\n')
    elif nivell == 2:
        g.write(t + '\n')
        g.write("-"*len(t)+'\n')



def escriure_autors(g, la):
    """
```

```
    Escriu en el fitxer `g` una lista d'autors.

    g: fitxer obert en mode escriptura
    la: llista d'strings que representen noms d'autor
    """
    for autor in la:
        f.write(":Author: %s\n" % autor)


def error():
    """
    Escriu el missatge d'error i acaba l'execució
    del programa.
    """
    print "error: sintaxi incorrecta"
    print "usage: python cm.py [-m] <nom_modul>"
    sys.exit()


def obte_autors():
    """
    Retorna la llista de noms d'autors que conté el fitxer `autors.dat`
    """
    fa = open("autors.dat", "r")
    llista_a = []
    for a in fa:
        llista_a.append(a.strip())
    fa.close()
    return llista_a



if __name__ == "__main__":
    # parsing comanda
    if len(sys.argv) != 2:
        error()
    es_modul = False
    nom_modul = sys.argv[1]


    # fa la feina
    nom_f = nom_modul + ".py"
    f = open(nom_f, "w")
    f.write("# -*- encoding: utf-8 -*-\n")

    # Escrivim el comentari del mòdul
    f.write('"""\n')
    escriure_titol(f,'Mòdul '+ nom_modul, 1)
    f.write('\n')

    # Escrivim llista d'autors
    escriure_autors(f, obte_autors())
    f.write('"""\n')

    # escrivim condicional del main
    f.write('\n\n')
```

```
    if not es_modul:
        f.write('if __name__ == "__main__":\n')
    f.close()
```