11 - Fitxers

November 20, 2015

*Authors* : *Sonia Estrade*´

*Jose M. G* ´ *omez* ´

*Ricardo Graciani*

*F rank Guell* ¨

*M anuel Lopez* ´

*Xavier Luri*

*Josep Sabater*

Moltes vegades ´es necessari enmagatzemnar i demanar dades per no perdre-les. Python permet guardar la informaci´o en un fitxer i llegir-la quan sigui necessari.

1 Funcions m`agiques al Notebook

iPython Notebook t´e un conjunt de ‘magic functions’ que les podeu cridar des del notebook cell of code. Hi ha dos tipus: line-oriented (%) and cell-oriented (%%).

La funci´o %lsmagic s’utilitza per llistar totes les funcions m`agiques, i mostrar`a els dos tipus correctament definides:

In [1]: %lsmagic

Out[1]: Available line magics:

%alias %alias magic %autocall %automagic %autosave %bookmark %cat %cd %clear %colors

Available cell magics:

%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%latex %%perl Automagic is ON, % prefix IS NOT needed for line magics.

Per exemple, podem utilitzar en Unix/Linux el comand pwd per mostrar el directori de treball actual: In [2]: %pwd

Out[2]: ’/Users/chema/Documents/Clases/Informatica/Repositori/Python/Apunts 2015-2016/Catala/11 - Fitxe

A les properes seccions, utilitzarem m´es funcions m`agiques, especialment les relacionades amb fitxers. Podeu trobar m´es exemples aqu´ı.

1

2 Treballant amb dades de text b`asiques

El primer pas ´es llegir o escriure dades desestructurades en un fitxer.

2.1 Llegint un fitxer de text simple

Primer creem un fitxer simple, utilitzant la funci´o m`agica cell-oriented %%file:

In [3]: %%file test.txt

This is a text file created to check

Python file read and write.

Writing test.txt

Per llegir el fitxer, en primer lloc cal obrir-lo:

In [4]: inputFile = open(’test.txt’, ’r’)

print(inputFile)

< io.TextIOWrapper name=’test.txt’ mode=’r’ encoding=’UTF-8’>

Els par`ametres s´on:

*•* name of the file.

*•* read mode (r), we can also use binary mode (b) to avoid problems with text conversion. *•* the optional universal line-end mode (U) to be able to interchange documents between operating sys tems.

A continuaci´o llegim les l´ınies del fitxer:

In [5]: file\_in = inputFile.readlines()

print(file\_in)

for line in file\_in:

print(line, end=’’)

[’This is a text file created to check*\*n’, ’Python file read and write.’] This is a text file created to check

Python file read and write.

A continuaci´o el fitxer s’ha de tancar:

In [6]: inputFile.close()

2.2 Escribint un fitxer de text simple

Ara seguint un procediment similar escriurem un fitxer desestructurat.

In [7]: outputFile = open(’secondTest.txt’, ’w’)

In [8]: outputFile.write(’Another file\n’)

outputFile.write(’A second line\n’)

Out[8]: 14

In [9]: outputFile.close()

In [10]: %less secondTest.txt

2

2.3 La sent`encia with

D’avegades els fitxers no es poden escriure o llegir apropiadament i apareixen errors. I si apareix un error, el fitxer s’ha de tancar sempre. Per evitar problemes, python t´e la sent`encia with .

In [11]: with open(’thirdTest.txt’, ’w’) as outputFile:

for i in range(10):

outputFile.write(’New test. ’ +

str(i) + ’\n’)

In [12]: %less thirdTest.txt

In [13]: outputFile.closed

Out[13]: True

El fitxer s’ha escrit i tancat.

In [14]: with open(’thirdTest.txt’, ’r’) as inputFile:

print(inputFile.read(), end = ’’)

New test. 0

New test. 1

New test. 2

New test. 3

New test. 4

New test. 5

New test. 6

New test. 7

New test. 8

New test. 9

3 Treballant amb moduls d’ajuda

Ara intentarem escriure dades. Per exemple, podem crear dues fileres de dades:

In [15]: import math

x = [point/10. for point in range(0, 100)]

y = [math.sin(x\_point) for x\_point in x]

In [16]: %pylab inline

%config InlineBackend.figure\_format = ’svg’

import matplotlib.pyplot as plt

str\_title = ’f(x) = sin(x)’

fig1 = plt.figure()

fig1.suptitle(str\_title, fontsize=14)

fig1\_ax = fig1.add\_subplot(1, 1, 1)

fig1\_ax.set\_xlabel(’x’)

fig1\_ax.set\_ylabel(’sin(x)’)

fig1\_ax.grid(True, which=’both’)

fig1\_ax.plot(x, y)

3

Populating the interactive namespace from numpy and matplotlib

Out[16]: [<matplotlib.lines.Line2D at 0x11354e400>]

3.1 Escribint un fitxer amb json

La manera m´es simple d’enmagatzemar dades des de Python ´es utilitzant el m`odul json. En aquest cas, nom´es cal utilitzar la funci´o dump amb la variable a enmagatzemar com a primer argument i el fitxer com a segon:

In [17]: import json

with open(’sin.json’, ’w’) as outputFile:

json.dump({’x’: x, ’y’: y}, outputFile, sort\_keys=True)

Amb aix`o, s’ha creat un fitxer que cont´e la informaci´o des de x a y.

In [18]: %less sin.json

Per`o no ´es f`acil entendre el fitxer.

3.2 Llegint el fitxer json

Ara ´es pot demanar amb la funci´o load, la qual retorna els continguts de la variable. Es important tenir en ´ compte que l’ordre de llegir les dades ha de ser el mateix en el que s’han enmagatzemat:

4

In [19]: with open(’sin.json’, ’r’) as inputFile:

data = json.load(inputFile)

xf = data[’x’]

yf = data[’y’]

In [20]: import matplotlib.pyplot as plt

plt.plot(xf, yf, x, y)

Out[20]: [<matplotlib.lines.Line2D at 0x113632b00>,

<matplotlib.lines.Line2D at 0x113632cc0>]

La dificultat m´es gran d’aquest format ´es que nom´es Python ho enten. Per aquesta ra´o, si desitjem intercanviar les dades, ser`a necessari utilitzar un format estructurat.

4 Dades estructurades

La manera b`asica ser`a treballant amb fitxers txt :

4.1 Treballant amb l’estructurat txt

Primer cal definir el format. Per simplificar com llegir-ho, l’escriurem de forma similar a com ho fariem en una taula. Les columnes estaran separades per tabuladors:

In [21]: with open(’sin.txt’, ’w’) as outputFile:

for dataX, dataY in zip(x, y):

outputFile.write(str(dataX) +

"\t" + str(dataY) +

"\n")

5

Les dades s’enmagatzemaran entre l´ınies, la x a la primera columna i la y a la segona, i separats per un tabulador.

In [22]: %less sin.txt

La dificultat m´es gran ´es llegir de nou les dades, ja que s’han de decodificar. Primer s’obre el fitxer i totes le l´ınies de text s´on demanades:

In [23]: with open(’sin.txt’, ’r’) as inputFile:

lines = []

*# The lines are read from the file*

for line in inputFile.readlines():

*# The line is stored in the list*

lines.append(line)

Cad l´ınia t´e dos valors, la x i la y, encara que barrejades amb la tabulaci´o i la nova l´ınia: In [24]: lines[1]

Out[24]: ’0.1*\*t0.09983341664682815*\*n’

Primer ´es necessari separar els dos valors. Per aix`o cal dos passos, en primer lloc cal treure els innecessaris car`acters espais (strip):

In [25]: lineStripped = lines[0].strip()

lineStripped

Out[25]: ’0.0*\*t0.0’

El segon pas ser`a split la l´ınia en dues cel*·*les *\*t.

In [26]: lineSplitted = lineStripped.split(’\t’)

lineSplitted

Out[26]: [’0.0’, ’0.0’]

Finalment, ser`a necessari convertir les cadenes en valors flotants, utilitzant la funci´o float: In [27]: float(lineSplitted[0])

Out[27]: 0.0

Amb aquest proc´es, les dades s´on demanades:

In [28]: xf = []

yf = []

for line in lines:

*# The line is stripped taking out all the unnecesary characters*

lineStripped = line.strip()

*# The line is splitted using the tab character*

lineSplitted = lineStripped.split(’\t’)

xf.append(float(lineSplitted[0]))

yf.append(float(lineSplitted[1]))

6

El proc´es total es pot agrupar:

In [29]: xf = []

yf = []

with open(’sin.txt’, ’r’) as inputFile:

for line in inputFile.readlines():

lineStripped = line.strip()

lineSplitted = lineStripped.split(’\t’)

xf.append(float(lineSplitted[0]))

yf.append(float(lineSplitted[1]))

Ara les dades es poden mostrar:

In [30]: import matplotlib.pyplot as plt

plt.plot(xf, yf)

Out[30]: [<matplotlib.lines.Line2D at 0x113738710>]

Aix`o implica que per cada fitxer, ´es necessari escriure el codi strip i split i processar la informaci´o. Python d´ona una alternativa.

4.2 Utilitzant el m`odul csv

Les mateixes operaciones es poden seguir, per`o amb el m`odul csv, que fa fitxers compatibles amb Excel o Calc. En aquest cas, necessitem utilitzar l’ordinador local, per estar segurs que ambd´os programes ententen correctament els continguts dels fitxers. Veiem un exemple:

7

In [31]: import locale

locale.setlocale(locale.LC\_ALL, ’’)

with open(’sinCSV.csv’, ’w’) as outputFile:

for dataX, dataY in zip(x, y):

outputFile.write(’{:n};{:n}\n’.format(dataX, dataY))

In [32]: %less sinCSV.csv

El fitxer separa les columnes amb semi-comes. Ara ho podem llegir amb Excel o Calc. Podem fer el mateix process, per`o utilitzant el m`odul csv. En aquest cas, per estar segur que tenim el desitjat format natiu, hem de modificar lleugerament el procediment:

In [33]: import locale

import csv

locale.setlocale(locale.LC\_ALL, ’’)

with open(’sinCSV.csv’, ’w’) as outputFile:

writer = csv.writer(outputFile, delimiter=’;’)

dataX = [locale.str(value) for value in x]

dataY = [locale.str(value) for value in y]

writer.writerows(zip(dataX, dataY))

El resultat ´es equivalent. Tamb´e podem llegir el fitxer:

In [34]: %less sinCSV.csv

Aix`o ´es f`acil de fer amb el m`odul csv:

In [35]: import csv

sinList = []

with open(’sinCSV.csv’, ’r’) as csvfile:

sinReader = csv.reader(csvfile,

delimiter=’;’,

quotechar=’"’)

for row in sinReader:

sinList.append(row)

In [36]: len(sinList[0])

Out[36]: 2

En aquest cas les cel*·*les estan completament separades:

In [37]: type(sinList[0][0])

Out[37]: str

Ara els valors es poden convertir a flotants com en el cas previ. Per simplificar podem utilitzar: In [38]: data = [[float(cell) for cell in row] for row in sinList]

8

--------------------------------------------------------------------------- ValueError Traceback (most recent call last)

<ipython-input-38-d02b43a28160> in <module>()

----> 1 data = [[float(cell) for cell in row] for row in sinList]

<ipython-input-38-d02b43a28160> in <listcomp>(.0)

----> 1 data = [[float(cell) for cell in row] for row in sinList]

<ipython-input-38-d02b43a28160> in <listcomp>(.0)

----> 1 data = [[float(cell) for cell in row] for row in sinList]

ValueError: could not convert string to float: ’0,1’

Apareix un problema nou, l’´us de la coma enlloc del punt, no permet convertir la cadena a flotant. Aix`o es pot solucionar utilitzant el m`odul locale altre cop. Hem convertit les cadenes a flotants utilitzant la funci´o atof function de locale:

In [39]: data = [[locale.atof(cell) for cell in row] for row in sinList] Ara aquestes dades es poden transferir a dues llistes:

In [40]: xf = []

yf = []

with open(’sinCSV.csv’, ’r’) as inputFile:

sinReader = csv.reader(inputFile, delimiter=’;’, quotechar=’"’)

for row in sinReader:

xf.append(locale.atof(row[0]))

yf.append(locale.atof(row[1]))

In [41]: %config InlineBackend.figure\_format = ’svg’

import matplotlib.pyplot as plt

plt.plot(xf, yf)

Out[41]: [<matplotlib.lines.Line2D at 0x1138adfd0>]

9

Ara ´es possible llegir els fitxers escrits. 10