Uso dei file (di testo)

Informatica@SEFA 2018/2019 - Lezione 19

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Mercoledì, 28 Novembre 2018

Dati temporanei

Dati elaborati da un programma

- presenti in memoria RAM (veloce)
- scompaiono se il programma termina

come possiamo mantenere dati permanenti?

Dati permamenti

I **file** sono unità di dati

- permamenti
- memorizzati sul disco rigido o su internet
- letti o scritti da programmi

Permettono lo scambio e la comunicazione tra

- programmi
- computer
- momenti

Successione ordinata di bytes.

Un file di lunghezza L byte è

$$b_0 b_1 b_2 \ldots b_{L-1}$$

dove b_i è il byte (quindi otto bit) alla posizione i del file.

File di testo

Un file può contenere dati di natura arbitraria, ma a noi interesseranno principalmente **file di testo**.

Esempio:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est laborum.
```

Codifica di un file di testo

Un file di testo è un file che può essere **visto** come una sequenza di caratteri $c_0, c_1, \ldots$.

$$\underbrace{b_0 b_1}_{c_0} \; \underbrace{b_2 b_3 b_4}_{c_1} \; \underbrace{b_5}_{c_2} \; \underbrace{b_6 b_7}_{c_3} \ldots$$

La sequenza di byte nel file codifica la sequenza di caratteri del testo. Abbiamo parlato un po' delle codifiche

- ASCII
- Latin1, Latin2, …
- UTF-8

Importanza della codifica (1)

```
                                                                    1
# Una sequenza di byte                                              2
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32,          3
    117, 110, 32, 102, 97, 109, 111, 115, 111, 32, 108, 111,
    103, 105, 99, 111])
                                                                    4
testo = byte.decode('utf-8')                                        5
print(testo)                                                        6
```

```
Gödel è un famoso logico
```

Importanza della codifica (2)

```
# Una sequenza di byte
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32,
    117, 110, 32, 102, 97, 109, 111, 115, 111, 32, 108, 111,
    103, 105, 99, 111])

testo = byte.decode('latin1')
print(testo)
```

```
GÃ¶del Ã¨ un famoso logico
```

Importanza della codifica (3)

```
                                                                    1
# Una sequenza di byte                                              2
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32,          3
    117, 110, 32, 102, 97, 109, 111, 115, 111, 32, 108, 111,
    103, 105, 99, 111])
                                                                    4
testo = byte.decode('ascii')                                        5
print(testo)                                                        6
```

```
Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
UnicodeDecodeError: 'ascii' codec can't decode
                    byte 0xc3 in position 1: ordinal not in range(128)
```

Importanza della codifica (conclusione)

È necessario sapere come è stato codificato il file, per poterlo utilizzare come **file di testo**.

- la maggior parte dei testi oggigiorno è UTF-8
- i programmi python sono file di testo UTF-8
- vecchi testi europei sono Latin1
- vecchi testi americani sono ASCII
- UTF-8 e Latin1 sono compatibili ASCII

Operazioni di base

Apertura e chiusura di un file

Supponiamo di avere un file `documento.txt`

- già presente sul disco
- nella **stessa cartella** da cui eseguiamo il programma
- vogliamo aprirlo in **sola lettura**

```
f = open('documento.txt',encoding='utf-8')    # Apre il file  1
                                                            2
# operazioni varie sul file                                 3
                                                            4
f.close()                                   # chiude il file  5
```

Se un file non esiste…

```
f = open('non_esiste.txt',encoding='utf-8')                    1
```

```
: Traceback (most recent call last):
:   File "<stdin>", line 1, in <module>
: FileNotFoundError: [Errno 2] No such file or directory: 'non_esiste.txt'
```

Schema consigliato

Se invece di usare

```
f = open(...)                1
istruzione1                  2
istruzione2                  3
istruzione3                  4
f.close()                    5
```

usiamo

```
with open(...) as f:         1
    istruzione1              2
    istruzione2              3
    istruzione3              4
```

allora il file viene **sempre** chiuso, anche se

- un errore
- uscita dal programma
- ecc…

Riassumendo

Per aprire un file `documento.txt`

- esistente
- in sola lettura
- codificato UTF-8

```
with open('documento.txt',encoding='utf-8') as f:                1
    print('Il file documento.txt è stato aperto')                2
    print('e la variable "f" permette di accedervi')             3
```

```
Il file documento.txt è stato aperto
e la variable "f" permette di accedervi
```

Percorso del file

"Cartella corrente" indica la cartella nella quale è stato lanciato il comando `python3`.

```
import os                                                    1
print(os.getcwd())  # mostra la cartella corrente            2
```

```
/Users/massimo/lavori/didattica/2018.IntroInformatica/lectures
```

Se un file che vogliamo usare non è nella cartella corrente, allora il primo argomento di `open` deve essere il percorso di quel file.

Percorso del file (esempi)

```
# document.txt è nella cartella corrente                          1
with open('documento.txt',encoding='utf-8') as f1:                2
    ...                                                           3
                                                                  4
# percorso relativo su Mac e Linux                                5
with open('../didattica/eval.text',encoding='utf-8') as f2:       6
    ...                                                           7
                                                                  8
# percorso assoluto su Mac e Linux                                9
with open('/usr/share/dict/README',encoding='utf-8') as f:        10
    ...                                                           11
                                                                  12
# percorso assoluto su windows                                    13
with open('C:\Documents\romanzo.txt',encoding='utf-8') as f:      14
    ...                                                           15
```

Leggere i file di testo

Peculiarità di un file di testo

1. Viene visto da python come una sequenza di **caratteri** invece che come usa sequenza di byte.
2. La codifica di 'andare a capo' è
   - `'\n'` in Mac e Linux
   - `'\r\n'` in Windows

   ma python converte automaticamente.
3. Il file viene anche diviso in righe.

File di esempio dal Progetto Gutemberg

Tipicamente questi file sono codificati con una variante di UTF-8, che in python è indicata come `utf-8-sig`.

Leggere i caratteri dal file

Se l'oggetto file è `f`

- `f.read()` legge tutti i caratteri non ancora letti
- `f.read(n)` legge fino a `n` caratteri non ancora letti

```
with open('alice.txt',encoding='utf-8-sig') as alice:
    alice.read(711)   # roba di copyright, saltiamola
    print(alice.read(32))
    print(alice.read(10))
    print(alice.read(10))
    print(alice.read(10))
    print(alice.read(10))
```

```
CHAPTER I. Down the Rabbit-Hole


Alice was
 beginning
 to get ve
ry tired o
```

Caricare tutto il testo in una stringa

```
with open('alice.txt',encoding='utf-8-sig') as alice:
    testo=alice.read()
    print('# ',len(testo),type(testo))
    print(testo[711:1046])
    print("# Sono rimasti",len(alice.read()),'caratteri.')
```

```
#  163816 <class 'str'>
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, ‘and what is the use of a book,’ thought Alice ‘without pictures or
conversations?’
# Sono rimasti 0 caratteri.
```

Promemoria su `repr`

Se volete vedere anche i caratteri non stampabili (per esempio gli 'a capo') potete usare `repr`

```
testo = 'Prima riga\n\nSeconda riga.'
print(testo)
print(repr(testo))
```

```
Prima riga

Seconda riga.
'Prima riga\n\nSeconda riga.'
```

Leggere il file riga per riga

```
with open('alice.txt', encoding='utf-8-sig') as alice:          1
    print(repr(alice.readline()))                               2
    print(repr(alice.readline()))                               3
    print(repr(alice.readline()))                               4
```

```
'Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll\n'
'\n'
'This eBook is for the use of anyone anywhere at no cost and with\n'
```

Le righe lette con `readlines()` includono i `'\n'`

Caricare tutte le righe in memoria

Possiamo caricare tutto il file in una lista di righe.

```
with open('alice.txt', encoding='utf-8-sig') as alice:          1
    righe=alice.readlines()                                     2
                                                                3
    print(righe[3],end='')                                      4
    print(righe[120],end='')                                    5
    print(righe[154],end='')                                    6
    print('')                                                   7
    print(len(righe))                                           8
```

```
almost no restrictions whatsoever.  You may copy it, give it away or
sort of way, ‘Do cats eat bats? Do cats eat bats?’ and sometimes, ‘Do
into the loveliest garden you ever saw. How she longed to get out of

3736
```

Un oggetto file (di testo) è iterabile

L'iterazione è effettuata sulle righe del file

```
with open('alice.txt', encoding='utf-8-sig') as alice:          1
    maxlength=-1                                                2
    for riga in alice:                                          3
        if maxlength < 0:                                       4
            maxlength = len(riga)                               5
        else:                                                   6
            maxlength = max(maxlength,len(riga))                7
    print("La riga più lunga ha",maxlength,"caratteri.")        8
```

```
la riga più lunga è lunga 79 caratteri.
```

Ricerca in un file

Vediamo un esempio di come ottenere tutte le righe che contengono una data parola, in un file.

```
def ricerca_linee(nome_file, encoding, stringa):                1
    '''Ritorna la lista dei numeri delle linee del              2
    file nome_file in cui appare la stringa.'''                 3
    with open(nome_file, encoding=encoding) as f:               4
        lista_indici = []                                       5
        indice_corrente = 1                                     6
        for linea in f:                                         7
            if linea.find(stringa) != -1:                       8
                lista_indici.append(indice_corrente)            9
            indice_corrente += 1                                10
        return lista_indici                                     11
```

Ricerca in un file (esempio)

```
indici = ricerca_linee('alice.txt', 'utf-8-sig', 'Turtle')    1
print(indici)                                                 2
```

```
[2214, 2354, 2356, 2358, 2372, 2390, 2397, 2403, 2410,
 2411, 2415, 2420, 2422, 2426, 2432, 2439, 2442, 2449,
 2453, 2457, 2464, 2469, 2484, 2486, 2494, 2500, 2509,
 2521, 2533, 2537, 2551, 2560, 2564, 2572, 2578, 2584,
 2588, 2595, 2627, 2633, 2639, 2641, 2680, 2685, 2690,
 2697, 2708, 2713, 2741, 2747, 2752, 2775, 2783, 2785,
 2787, 2790, 2813, 3348, 3358]
```

```
indici = ricerca_linee('alice.txt', 'utf-8-sig', 'Alice')    1
print(len(indici))                                           2
```

```
396
```

UTF-8 vs UTF-8-sig

Praticamente lo stesso encoding, ma un file `utf-8-sig` ha un marcatore iniziale.

```
with open('alice.txt', encoding='utf-8') as f:                    1
    print(repr(f.readline()))                                     2
                                                                  3
with open('alice.txt', encoding='utf-8-sig') as f:                4
    print(repr(f.readline()))                                     5
```

```
'\ufeffProject Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll\r
'Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll\n'
```

Scrivere su file

Aprire un file in scrittura

```
with open('testo.txt', 'w', encoding='utf-8') as f:
    f.write('Questo è il contenuto del file.')

with open('testo.txt',encoding='utf-8') as f:
    print(f.read())
```

```
Questo è il contenuto del file.
```

È sufficiente:

- aggiungere `'w'` come argomento di `open`
- usare `write()` per scrivere testo

Modalità di apertura del file

Si può indicare la **modalità** di apertura del file

Modalità	Operazione	Tipo di file
`r`	Lettura	Testo
`w`	Scrittura ex novo	Testo
`a`	Scrittura in coda	Testo
`rb`	Lettura	Binario
`wb`	Scrittura ex novo	Binario
`ab`	Scrittura in coda	Binario

- `r` è la modalità di default
- `w` scrive cancellando il tutto contenuto precedente
- `a` aggiunge testo in coda

Scrivere nel file sequenze di caratteri

```
with open('testo.txt', 'w', encoding='utf-8') as f:          1
    f.write('Primo*')                                        2
    f.write('Secondo*')                                      3
    f.write('Terzo')                                         4
    f.write('\n')                                            5
    f.write('Quarto e quinto.')                              6
                                                             7
with open('testo.txt',encoding='utf-8') as f:                8
    print(f.read())                                          9
```

```
Primo*Secondo*Terzo
Quarto e quinto.
```

Scrivere nel file sequenze di stringhe

```
with open('testo.txt', 'w', encoding='utf-8') as f:                    1
    f.writelines(['Primo','Secondo','Terzo','Quarto'])                 2
    f.writelines(['Quinto','Sesto'])                                   3
    f.write('\n')                                                      4
    f.writelines(['Primo\n','Secondo\n','Terzo\n','Quarto\n'])         5
    f.writelines(['Quinto\n','Sesto\n'])                               6
                                                                       7
                                                                       8
with open('testo.txt',encoding='utf-8') as f:                          9
    print(f.read())                                                    10
```

```
PrimoSecondoTerzoQuartoQuintoSesto
Primo
Secondo
Terzo
Quarto
Quinto
Sesto
```

Scrivere in coda al file

```
with open('testo.txt', 'w', encoding='utf-8') as f:
    f.write('Linea1.\n')

with open('testo.txt', 'a', encoding='utf-8') as f:
    f.write('Linea2.\n')

with open('testo.txt', encoding='utf-8') as f:
    print(f.read())

with open('testo.txt', 'w', encoding='utf-8') as f:
    f.write('Linea Alternativa.\n')

with open('testo.txt', encoding='utf-8') as f:
    print(f.read())
```

```
Linea1.
Linea2.

Linea Alternativa.
```

Letture

Paragrafi 10.1, 10.2, 10.3, 10.4, 10.5.