

# Uso dei file (di testo)

Informatica@DSS 2019/2020 – Il canale

**Massimo Lauria** <massimo.lauria@uniroma1.it>

<https://massimolauria.net/courses/informatica2019/>

# Dati temporanei

Dati elaborati da un programma

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

come possiamo mantenere dati permanenti?

# Dati permanenti

I **file** sono unità di dati

- ▶ permanenti
- ▶ 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

$$b_0b_1b_2 \dots b_{L-1}$$

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  $c_0, c_1, \dots$

$$\underbrace{b_0b_1}_{c_0} \quad \underbrace{b_2b_3b_4}_{c_1} \quad \underbrace{b_5}_{c_2} \quad \underbrace{b_6b_7}_{c_3} \dots$$

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)

```
# Una sequenza di byte 1
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32, 117, 110,2
           32, 102, 97, 109, 111, 115, 111, 32, 108, 111, 103, 105,3
           99, 111]) 4
5
testo = byte.decode('utf-8') 6
print(testo) 7
```

Gödel è un famoso logico

# Importanza della codifica (2)

```
# Una sequenza di byte 1
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32, 117, 110,2
           32, 102, 97, 109, 111, 115, 111, 32, 108, 111, 103, 105,3
           99, 111]) 4
5
testo = byte.decode('latin1') 6
print(testo) 7
```

GÃ¶del Ã¨ un famoso logico



# Importanza della codifica (3)

```
# Una sequenza di byte 1
byte=bytes([71, 195, 182, 100, 101, 108, 32, 195, 168, 32, 117, 110,2
           32, 102, 97, 109, 111, 115, 111, 32, 108, 111, 103, 105,3
           99, 111]) 4
5
testo = byte.decode('ascii') 6
print(testo) 7
```

```
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 oggi giorno 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

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 se si esce dal blocco.

# 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
```

```
/home/massimo/lavori/progetti/informatica2019/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)

```
# documento.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

# File di testo dal Progetto Gutenberg

**Project Gutenberg**

search for books

- Browse Catalog
- Bookshelves
- Main Page
- Categories
- Contact Info

Project Gutenberg appreciates your donation!

[Donate](#)

- Why donate?

in other languages

- Português
- Deutsch
- Français

hosted by **ibiblio**

## Free ebooks - Project Gutenberg

[Book search](#) · [Book categories](#) · [Browse catalog](#) · [Mobile site](#) · [Report errors](#) · [Terms of use](#)

## Some of the Latest Books



## Welcome

**Project Gutenberg** offers over 57,000 free eBooks. Choose among free epub books, free kindle books, download them or read them online. You will find the world's great literature here, with focus on older works for which copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for enjoyment and education.

No fee or registration is required. If you find Project Gutenberg useful, please consider a small [donation](#), to help Project Gutenberg digitize more books, maintain our online presence, and improve Project Gutenberg programs and offerings. Other ways to help include [digitizing more books](#) ☺, [recording audio books](#) ☺, or [reporting errors](#).

# Esempi sulla pagina del corso

Nella sezione "Materiale" del corso ci sono dei libri di pubblico dominio in Inglese, in file `.txt` codificati in UTF-8.

- ▶ *Alice's Adventures in Wonderland* di Lewis Carroll
- ▶ *Frankenstein* di Mary W. Shelley
- ▶ *The Adventures of Sherlock Holmes* by A. C. Doyle
- ▶ *Moby Dick* di Herman Melville
- ▶ *The Prince* di Niccoló Machiavelli
- ▶ *Treasure Island* di Robert Louis Stevenson

# Leggere i caratteri dal file `f`

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

```
with open('../dataset/alice.txt',encoding='utf-8') as alice:      1
                                                                    2
    alice.read(711)      # roba di copyright, saltiamola          3
    print(alice.read(32))                                       4
    print(alice.read(10))                                       5
    print(alice.read(10))                                       6
    print(alice.read(10))                                       7
    print(alice.read(10))                                       8
```

CHAPTER I. Down the Rabbit-Hole

Alice was  
beginning  
to get ve  
ry tired o

# Caricare tutto il testo in una stringa

```
with open('../dataset/alice.txt',encoding='utf-8') as alice:      1
                                                                    2
    testo=alice.read()                                           3
    print('# INFO: ',len(testo),type(testo))                       4
    print(testo[711:1046])                                         5
    print("# Sono rimasti",len(alice.read()),'caratteri.')       6
```

```
# INFO: 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

Visualizza i caratteri non stampabili (per esempio gli 'a capo')

```
testo = 'Prima riga\n\nSeconda riga.' 1  
print(testo) 2  
print(repr(testo)) 3
```

```
Prima riga
```

```
Seconda riga.
```

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

# Leggere il file riga per riga

```
with open('../dataset/alice.txt', encoding='utf-8') 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 `readline` includono gli `\n`**



# Caricare tutte le righe in memoria

Possiamo caricare tutto il file in una lista di righe.

```
with open('../dataset/alice.txt', encoding='utf-8') as alice:      1
    righe=alice.readlines()                                       2
                                                                    3
    print(righe[10],end='')                                       4
    print(righe[120],end='')                                       5
    print(righe[154],end='')                                       6
    print('')                                                       7
    print("Il file ha {} righe ".format(len(righe)))              8
```

Author: Lewis Carroll

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

Il file ha 3736 righe

# Un oggetto file (di testo) è iterabile

L'iterazione è effettuata sulle righe del file

```
with open('../dataset/alice.txt', encoding='utf-8') as alice:      1
    maxlength=-1                                                  2
    for riga in alice:                                           3
        maxlength = max(maxlength,len(riga))                    4
    print("La riga più lunga ha",maxlength,"caratteri.")        5
```

La riga più lunga ha 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('../dataset/alice.txt', 'utf-8', '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('../dataset/alice.txt', 'utf-8', 'Alice') 1  
print(len(indici)) 2
```

# Scrivere su file

# Aprire un file in scrittura

```
with open('testo.txt', 'w', encoding='utf-8') as f:      1
    f.write('Questo è il contenuto del file.')          2
                                                         3
with open('testo.txt',encoding='utf-8') as f:          4
    print(f.read())                                     5
```

```
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:      1
    f.write('Linea1.\n')                                  2
                                                         3
with open('testo.txt', 'a', encoding='utf-8') as f:      4
    f.write('Linea2.\n')                                  5
                                                         6
with open('testo.txt', encoding='utf-8') as f:           7
    print(f.read())                                      8
                                                         9
with open('testo.txt', 'w', encoding='utf-8') as f:     10
    f.write('Linea Alternativa.\n')                      11
                                                         12
with open('testo.txt', encoding='utf-8') as f:          13
    print(f.read())                                      14
```

```
Linea1.
Linea2.

Linea Alternativa.
```