I FILE

Quando un programma è in esecuzione i suoi dati sono in memoria; nel momento in cui il programma termina o il computer viene spento tutti i dati in memoria vengono irrimediabilmente persi. Per conservare i dati devi quindi memorizzarli in un **file**, solitamente memorizzato su hard disk, floppy, chiavetta USB, CD-ROM.
Lavorando con un gran numero di file è logico cercare di organizzarli: questo viene fatto inserendoli in cartelle (dette anche "**folder**" o "**directory**"). Ogni file all'interno di una cartella è identificato da un nome unico.
Leggendo e scrivendo file i programmi possono scambiare informazioni e anche generare documenti stampabili usando il formato PDF o altri formati simili.
Lavorare con i file è molto simile a leggere un libro: per usarli li devi prima aprire e quando hai finito li chiudi. Mentre il libro è aperto lo puoi leggere o puoi scrivere una nota sulle sue pagine, sapendo in ogni momento dove ti trovi al suo interno. La maggior parte delle volte leggerai il libro in ordine, ma nulla ti vieta di saltare a determinate pagine facendo uso dell'indice.
Questa metafora può essere applicata ai file. Per aprire un file devi specificarne il nome e l'uso che intendi farne (lettura o scrittura).
L'apertura del file crea un oggetto file: nell'esempio che segue useremo la variabile `f` per riferirci all'oggetto file appena creato.

Apri Python in modalità interattiva ed esegui il codice seguente:

```
>>> f = open("test.dat","w")
>>> print f

<open file 'test.dat', mode 'w' at fe820>
```

La funzione open prende due argomenti: il primo è il nome del file ed il secondo il suo "modo". Il modo `"w"` significa che stiamo aprendo il file in scrittura.
Nel caso non dovesse esistere un file chiamato `test.dat` l'apertura in scrittura farà in modo di crearlo vuoto. Nel caso dovesse già esistere, la vecchia copia verrà rimpiazzata da quella nuova e definitivamente persa.

Quando stampiamo l'oggetto file possiamo leggere il nome del file aperto, il modo e la posizione dell'oggetto in memoria.
Per inserire dati nel file invochiamo il metodo `write`:

```
>>> f.write("Adesso")
>>> f.write("chiudi il file")
```

La chiusura del file avvisa il sistema che abbiamo concluso la scrittura e rende il file disponibile alla lettura:

```
>>> f.close()
```

Solo dopo aver chiuso il file possiamo riaprirlo in lettura e leggerne il contenuto. Questa volta l'argomento di modo è `"r"`:

```
>>> f = open("test.dat","r")
```

Se cerchiamo di aprire un file che non esiste otteniamo un errore:

```
>>> f = open("test.cat","r")

IOError: [Errno 2] No such file or directory: 'test.cat'
```

Il metodo `read` legge dati da un file. Senza argomenti legge l'intero contenuto del file:

```
>>> Testo = f.read()
>>> print Testo
Adessochiudi il file
```

Non c'è spazio tra `Adesso` e `chiudi` perchè non abbiamo scritto uno spazio tra le due stringhe al momento della scrittura.

`read` accetta anche un argomento che specifica quanti caratteri leggere:

```
>>> f = open("test.dat","r")
>>> print f.read(5)
Adess
```

Se non ci sono caratteri suffcienti nel file, `read` ritorna quelli effettivamente disponibili. Quando abbiamo raggiunto la fine del file `read` ritorna una stringa vuota:

```
>>> print f.read(1000006)
ochiudi il file
>>> print f.read()
>>>
```

FILE DI TESTO

Un file di testo è un file che contiene **caratteri stampabili** e **spazi bianchi**, **organizzati in linee** separate da **caratteri di ritorno a capo**. Python è stato specificatamente progettato per elaborare file di testo e fornisce metodi molto efficaci per rendere facile questo compito.
Per dimostrarlo creeremo un file di testo composto da tre righe di testo separate da dei ritorno a capo (`\n sta per newline`):

```
>>> f = open("test.dat","w")
>>> f.write("linea uno\nlinea due\nlinea tre\n")
>>> f.close()
```

Il metodo `readline` legge tutti i caratteri fino al prossimo ritorno a capo:

```
>>> f = open("test.dat","r")
>>> print f.readline()
linea uno
>>>
```

`readlines` ritorna tutte le righe rimanenti come lista di stringhe:

```
>>> print f.readlines()
['linea due\n', 'linea tre\n']
```
In questo caso il risultato è in formato lista e ciò significa che le stringhe appaiono racchiuse tra apici e i caratteri di ritorno a capo come **sequenze di escape** (**righe di comando**, come \n, che indica un ritorno a capo).

Alla fine del file `readline` ritorna una stringa vuota e `readlines` una lista
vuota:
```
>>> print f.readline()
>>> print f.readlines()
[]
```
Quello che segue è un esempio di elaborazione di un file: `FiltraFile` fa una copia del file `Originale` omettendo tutte le righe che iniziano con #:

```
def FiltraFile(Originale, Nuovo):
            f1 = open(Originale, "r")
            f2 = open(Nuovo, "w")
            while 1:
                        Linea = f1.readline()
                        if Linea == "":
                                    break
                        if Linea[0] == '#':
                                    continue
                        f2.write(Linea)
            f1.close()
            f2.close()
            return
```

L'istruzione `continue` termina l'iterazione corrente e continua con il prossimo ciclo: il flusso di programma torna all'inizio del ciclo, controlla la condizione e procede di conseguenza.

Non appena `Linea` è una stringa vuota il ciclo termina grazie al `break`. Se il primo carattere di `Linea` è un carattere cancelletto (#) l'esecuzione continua tornando all'inizio del ciclo. Se entrambe le condizioni falliscono (non siamo in presenza della fine del file né il primo carattere è un cancelletto) la riga viene copiata nel nuovo file.

Scrivi ed esegui il seguente programma

```
# apro un file in scrittura
file=open("prova.txt","w")          #se il file prova.txt non esiste, viene creato

# scrivo una riga per volta
file.write("Questa e' una scrittura di prova")
file.write("")
file.write("Non vado a capo")

# chiudo il file
file.close()


# apro un file in lettura
altro=open("prova.txt","r")

# leggo tutto il file e lo stampo
print altro.read()

altro.close()


# apro un file in scrittura: nota che si sovrascrive prova.txt
file=open("prova.txt","w")

file.write("Anche questa e' una scrittura di prova\n")     # \n sta per newline
file.write("\n")
file.write("Pero' vado a capo")

file.close()

altro=open("prova.txt","r")

# stampo una sola riga
print altro.readline()

altro.close()

altro=open("prova.txt","r")

# stampo tutte le righe
listarighe = altro.readlines()
for riga in listarighe:
    print riga

altro.close()
```