

LINGUAGGIO PYTHON

I/O SU FILE

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE



PROF. ANDREA PINNA

AA 2019/2020

FILE

Python permette di creare ed aprire file di testo sui quali scrivere e leggere dati, sotto forma di stringhe. Un file è composto da linee ed ogni linea termina con il carattere speciale End of Line.

I file si possono manipolare in quattro modalità:

- 'r' **lettura**: un file in modalità lettura si può leggere ma non modificare.
- 'w' **scrittura**: un file in modalità scrittura si può modificare ma non leggere. Il file viene considerato vuoto.
- 'a' **aggiunta** (append): un file in modalità append può essere scritto e i nuovi dati vengono scritti alla fine del file.
- 'r+' **lettura** e scrittura: modalità speciale che permette di leggere e scrivere sul file.



FILE: OPEN

La funzione di built-in `open` permette di aprire un file in una delle quattro modalità. Ha come argomenti il nome del file (di tipo stringa) e una modalità (di tipo stringa). Restituisce un riferimento alla locazione di memoria in cui è caricato il file.

Sintassi:

```
nomeVariabileFile = open(nomefile, modalità)
```

Esempio: apro il file `visitatori2019.txt` in modalità lettura

```
fileVisitatori = open("visitatori2019.txt", "r")
```



FILE: OPEN

La funzione `open` restituisce un tipo di dato chiamato **wrapper** (involucro). Il wrapper permette di manipolare il file con le istruzioni del linguaggio python, evitando di intervenire direttamente sul processo di lettura e scrittura dei dati registrati.

Nello specifico, chiamando `type` sul risultato di `open` ottengo:

```
>>> type(open("visitatori2019.txt", "r"))  
<class '_io.TextIOWrapper'>
```

Il wrapper del file è un oggetto nel quale, oltre al riferimento al file, è disponibile un insieme di metodi.



FILE: MODALITÀ WRITE

Per **creare** un nuovo file (o sovrascrivere uno esistente) si usa la modalità write ("w")

Esempio:

```
fileDaScrivere=open("prove.txt", "w")
```

Ora possiamo agire sulla variabile `fileDaScrivere` per scrivere sul nostro file `prove.txt`.



FILE: MODALITÀ WRITE

Per scrivere una linea sul file utilizzo il metodo `write` dell'oggetto wrapper del file. Il metodo `write` prende in ingresso una stringa. Restituisce il numero di caratteri da scrivere sul file.
Esempio.

```
>>> fileDaScrivere.write("ciao\n")
```

```
4
```

Il carattere speciale “\n” indica che la linea è finita (End Of Line) e che il prossimo `write` verrà inserito sotto.

Senza il carattere \n, un nuovo `write` verrebbe affiancato al precedente.



FILE: MODALITÀ WRITE

Attenzione:

Il metodo write non modifica istantaneamente il file. Le modifiche **restano in attesa** di essere effettuate fino a che non eseguo il metodo flush.

```
>>> fileDaScrivere.flush()
```

Esempio:

```
>>> fileDaScrivere=open("prove.txt", "w")
>>> fileDaScrivere.write("ciao")
>>> fileDaScrivere.write("prova\n") #a capo
>>> fileDaScrivere.write("Fine...")
>>> fileDaScrivere.flush()
```

Risultato:

prove.txt

```
ciaoprova
Fine...
```



FILE: CHIUSURA

Una volta concluse le operazioni sul file occorre chiuderlo. Il metodo `close` chiude il file eseguendo le ultime modifiche in attesa. Chiudere un file significa disabilitare le operazioni di lettura e scrittura su di esso.

Esempio:

```
>>> fileDaScrivere.close()
>>> fileDaScrivere.write("ciao") #provo a modificarlo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
```



CURRENT WORK DIRECTORY

Dove stiamo scrivendo? Possiamo chiedere all'interprete il percorso della directory in cui vengono creati i nuovi file e dalla quale quale verranno letti file esistenti.

Si può utilizzare la funzione `getcwd()` che fa parte di una libreria del linguaggio chiamata `os`. Una **libreria** è un insieme di nomi di funzioni o di variabili raggruppate per scopo.

La libreria `os` riguarda il sistema operativo. Infatti la gestione delle cartelle (il filesystem) è una delle responsabilità del sistema operativo .

Un altro esempio di libreria è la libreria `math` che contiene funzioni e variabili matematiche (`sin`, `cos`, `pi`).



CURRENT WORK DIRECTORY

Sintassi:

All'inizio del programma (o in qualunque momento se uso la console) eseguo l'import della funzione `getcwd` dalla libreria `os`.

```
from os import getcwd
```

Significa: "dalla libreria `os` importa il nome `getcwd` in essa definito"

Una volta importata, posso usare la funzione `getcwd` che non ha argomenti e restituisce una stringa. Esempio:

```
>>> print(getcwd())  
/home/andrea
```



FILE: MODALITÀ READ

Per leggere il contenuto di un file si usa la modalità lettura "r".

```
fileInLettura = open("prove.txt", "r")
```

Per leggere le linee di un file si usa il metodo `readline()` che non ha argomenti e restituisce una stringa uguale ad una delle riga del file. In particolare, la prima volta che si esegue `readline` restituisce la prima linea, la seconda volta restituisce la seconda linea, e così via.



FILE: MODALITÀ READ

Esempio:

```
fileInLettura = open("prove.txt", "r")
unaLinea = fileInLettura.readline()
secondalinea = fileInLettura.readline()
print(secondalinea)
```

Apparirà a video:

Fine...

prove.txt

```
ciaoprova
Fine...
```



FILE: MODALITÀ READ

Raggiunta la fine del file `readline()` restituisce la stringa vuota.

Esempio:

```
fileInLettura = open("prove.txt", "r")
linea = True
testo = "" #voglio mettere qui dentro il contenuto del file.
while linea:
    linea = fileInLettura.readline()
    testo += linea
print(testo)
```

Dato che la stringa vuota viene valutata come `False` posso fermare il ciclo quando l'ultima linea letta è vuota.



FILE: SEEK E TELL

Per **decidere** da quale posizione leggere con il prossimo `readline` si deve utilizzare il metodo `seek` che prende in ingresso un numero intero positivo e posiziona il cursore nella posizione desiderata.

Esempio:

```
fileInLettura.seek(0)
```

sposta il cursore all'inizio della prima linea.

```
fileInLettura.seek(4)
```

sposta il cursore dopo il quarto carattere.

Per **conoscere** la posizione del cursore posso usare il metodo `tell` senza argomenti

```
fileInLettura.tell()
```



FILE: SEEK

Seek in modalità 2: permette di spostare il cursore in una posizione partendo dalla fine del file. Per ottenere questa modalità di posizionamento è necessario inserire un secondo argomento di valore 2.

La fine del file si raggiunge con questa sintassi:

```
fileAperto.seek(0, 2)
```

Raggiunta la fine del file, usando `tell` posso conoscere il numero di caratteri del file.

```
numCaratteri = fileAperto.tell()
```



FILE: MODALITÀ APPEND

La modalità "a" append è una modalità di scrittura che apre un file esistente. Il metodo `write` usato su un file in modalità append aggiungerà una linea alla fine del file.

Esempio:

```
fileInAppend = open("prove.txt", "a")
fileInAppend.write("Nuovi elementi\n")
fileInAppend.flush()
```

prove.txt

```
ciaoprova
Fine...
Nuovi elementi
```

Linea aggiunta



FILE: MODALITÀ R+

La modalità "r+" read-write è una modalità di lettura e scrittura che apre un file esistente.

Il metodo `write` usato su un file in modalità r+ sovrascriverà gli elementi della linea corrispondente alla posizione attuale del cursore.

Il metodo `readline` si comporterà come nella modalità read.

Posso sempre spostare il cursore con il metodo `seek`.

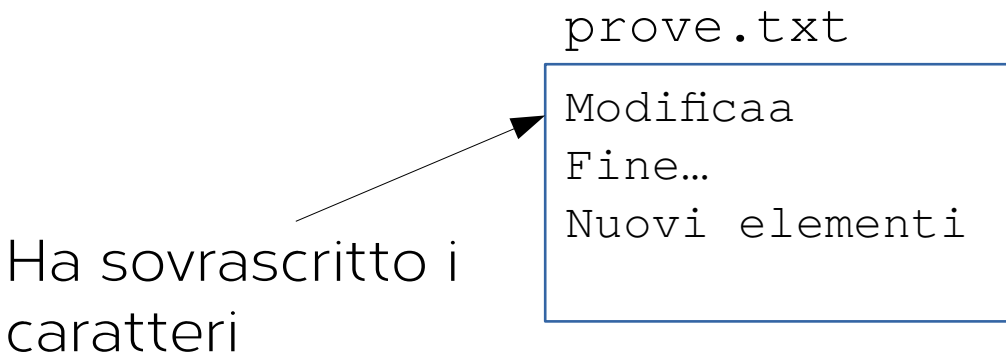


FILE: MODALITÀ R+

Esempio:

```
fileInRW = open("prove.txt", "r+")  
fileInRW.seek(0)  
fileInRW.write("Modifica")  
fileInRW.flush()
```

prove.txt



Modificaa
Fine..
Nuovi elementi

Ha sovrascritto i
caratteri



COSTRUTTO WITH

Il costrutto `with open` permette di eseguire un'operazione sul file e di chiuderlo subito dopo le istruzioni del suo blocco di codice.

```
with open("prove.txt", "r") as fileAperto:  
    linea = fileAperto.readline()  
    print(linea)
```

Al termine del blocco di codice chiude il file ma il dato estratto resta dentro la variabile `linea`.



FILE: READLINES

Dato un file aperto in modalità `r` o `r+` è possibile estrarre tutte le sue linee in una sola volta.

Il metodo `readlines()` del wrapper del file restituisce una lista di stringhe. Esempio con il costrutto `with`:

```
with open("prove.txt", "r") as fileAperto:  
    linee = fileAperto.readlines()  
    print(linee)
```

Ma cos'è una lista? È un tipo di dato strutturato.

