[FIM] FONDAMENTI DI INFORMATICA
per medicina e chirurgia high tech

P08: File

Dott. Giorgio De Magistris

demagistris@diag.uniroma1.it

Corso di Laurea in Medicina e Chirurgia High Tech

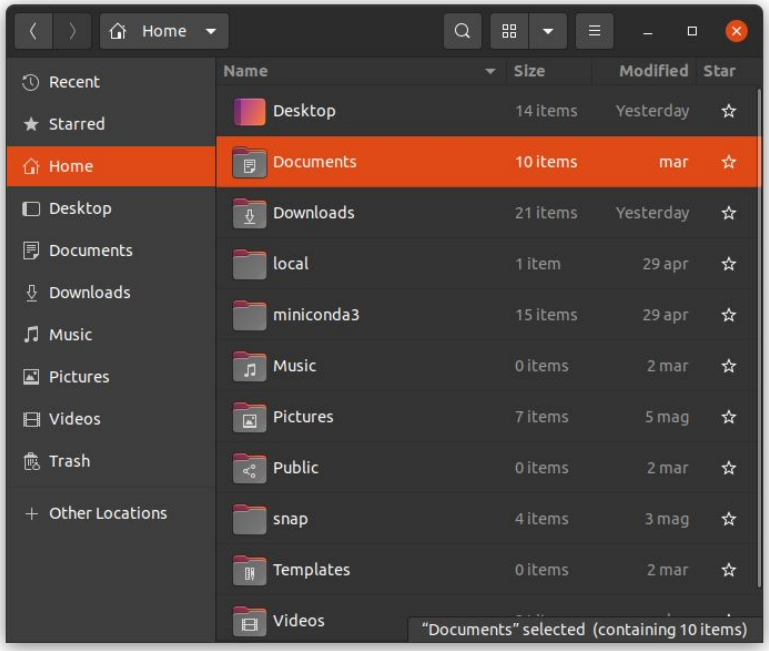
Sapienza
Università di Roma

I3S Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

DIAG Dipartimento di Ingegneria Informatica, Automatica e Gestionale

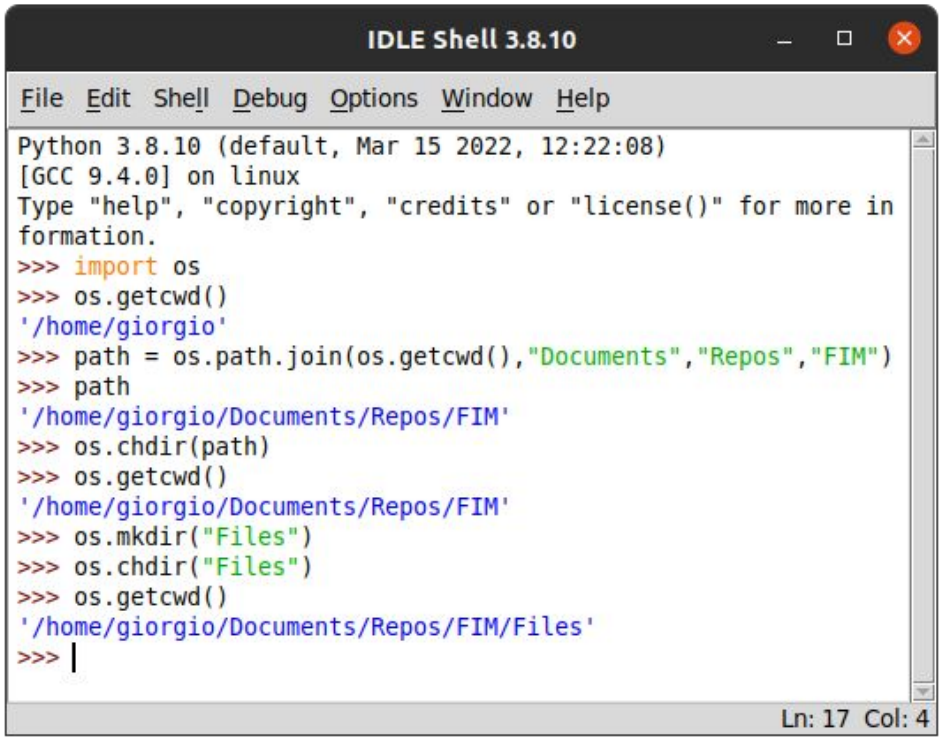
Files

- Quando un programma termina, la sua memoria viene liberata
- Per fare in modo che i dati continuino ad esistere dopo la terminazione del programma, bisogna salvarli su disco
- Generalmente il filesystem è organizzato in file e directory
- Le directory sono dei contenitori che possono contenere file o altre directory
- I file sono le unità in cui è possibile memorizzare i dati



Modulo os

- Permette di accedere a funzionalità che dipendono dal sistema operativo
- Noi utilizzeremo le seguenti funzioni:
  - **os.getcwd( )** : restituisce il path della working directory
  - **os.path.join( )** : unisce due paths utilizzando il delimitatore corretto del proprio sistema operativo (\\ per windows e / per linux/mac)
  - **os.chdir(path)** : cambia il path della working directory
  - **os.mkdir(path)** : crea una directory vuota nel path specificato



Files

- Noi sappiamo che il programma e i suoi dati vivono nella RAM
- I dati persistenti invece vanno salvati su disco
- Il sistema operativo si occupa del trasferimento dei dati da disco a RAM e vice versa
- Noi programmatori Python abbiamo accesso a delle API che ci permettono di chiedere al sistema operativo di leggere e scrivere su file

Files

- In Python un file è rappresentato da un file object

- Su di un file si possono eseguire le seguenti operazioni

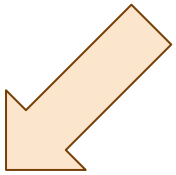
  - **Apertura**: ovvero creare un file object che mi permette di leggere e scrivere sul file
  - **Lettura** e **scrittura**
  - **Chiusura**: serve per liberare la memoria e le risorse allocate dal sistema operativo

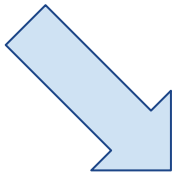
- Per ottenere un file object devo chiamare la funzione built-in **open**

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> type(f)
<class '_io.TextIOWrapper'>
>>> s = f.read()
>>> s
'Matricola,Esame,Voto,Data\n1892748,TDP,28,18/02/2020
\n1782932,DM,18,17/07/2021\n1720389,FI,26,17/03/2020\
n1702347,AD,30,16/06/2021\n1823643,ML,29,10/01/2020'
>>> s = f.read()
>>> s
''
>>> f.close()
>>>
```

Binary vs Text

- In Python un file può essere aperto in due modalità

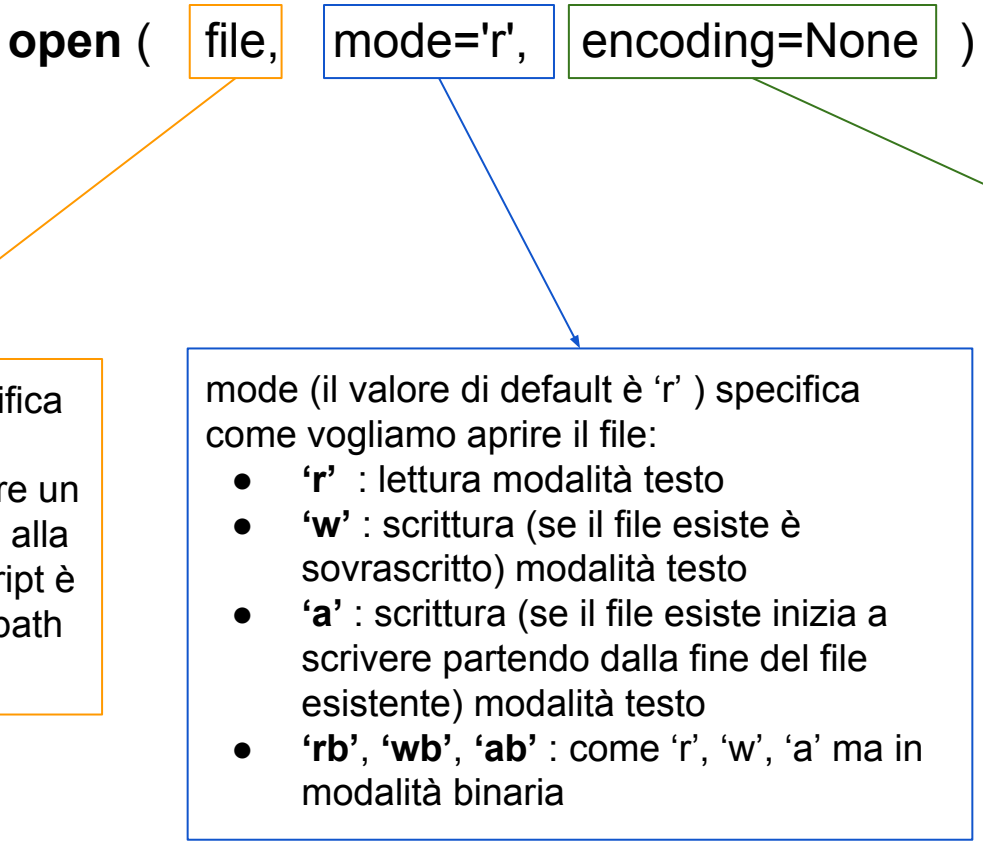




Modalità Binaria
Le unità che leggo e scrivo sul file sono i bytes

Modalità Testuale
Le unità che leggo e scrivo sul file sono caratteri (il numero di bytes che viene letto o scritto per ogni carattere dipende dall'encoding utilizzato)

Open


open (file, mode='r', encoding=None)
una stringa che specifica il path del file nel filesystem. Può essere un path relativo (rispetto alla directory da cui lo script è stato eseguito) o un path assoluto
mode (il valore di default è ‘r’) specifica come vogliamo aprire il file:
‘r’ : lettura modalità testo
‘w’ : scrittura (se il file esiste è sovrascritto) modalità testo
‘a’ : scrittura (se il file esiste inizia a scrivere partendo dalla fine del file esistente) modalità testo
‘rb’, ‘wb’, ‘ab’ : come ‘r’, ‘w’, ‘a’ ma in modalità binaria
va specificato solo se il file è aperto in modalità testo, di solito “utf-8”

Leggere un file

- la funzione open ritorna un **file object**
- il file object contiene un riferimento alla posizione all'interno del file
- esistono due modi per leggere un file
  - **lettura sequenziale**: quando il file viene aperto, la posizione corrente punta all'inizio del file e viene incrementata finchè non raggiunge la fine del file (EOF)
  - **accesso diretto**: specifico direttamente la posizione all'interno del file (in bytes) a cui voglio accedere con **f.seek(pos)** . Questa modalità di accesso è prevalentemente usata per i file binari che non vedremo

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> f.readline()
'Matricola,Esame,Voto,Data\n'
>>> f.readline()
'1892748,TDP,28,18/02/2020\n'
>>> f.readline()
'1782932,DM,18,17/07/2021\n'
>>> f.readline()
'1720389,FI,26,17/03/2020\n'
>>> f.readline()
'1702347,AD,30,16/06/2021\n'
>>> f.readline()
'1823643,ML,29,10/01/2020'
>>> f.readline()
''
>>>
```

Leggere un file

- un file object **f** ha diversi metodi che permettono di leggere il contenuto del file

- alcuni metodi restituiscono l'intero contenuto del file (indipendentemente dalla grandezza) e per questo motivo non possono essere usati per leggere file di grandi dimensioni, essi sono:

  - **f.read( )** : legge l'intero contenuto del file e lo restituisce come stringa

  - **f.readlines( )** : legge l'intero contenuto del file e lo restituisce come lista di stringhe, in cui ogni elemento corrisponde ad una riga del file

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> f.read()
'Matricola,Esame,Voto,Data\n1892748,TDP,28,18/02/2020
\n1782932,DM,18,17/07/2021\n1720389,FI,26,17/03/2020\
n1702347,AD,30,16/06/2021\n1823643,ML,29,10/01/2020'
>>> f.seek(0)
0
>>> f.readlines()
['Matricola,Esame,Voto,Data\n', '1892748,TDP,28,18/02
/2020\n', '1782932,DM,18,17/07/2021\n', '1720389,FI,2
6,17/03/2020\n', '1702347,AD,30,16/06/2021\n', '18236
43,ML,29,10/01/2020']
```

Leggere un file

- Quando utilizzo un metodo per leggere il contenuto il riferimento della posizione viene aggiornato
- Quando la posizione punta alla fine del file **f.read( )** restituisce la stringa vuota
- Attenzione! Se chiamo due volte **f.read( )** o **f.readlines( )** la seconda volta restituiranno rispettivamente la stringa vuota “ ” o la lista vuota **[ ]**
- Per leggere nuovamente il file posso chiuderlo e riaprirlo o cambiare la posizione corrente con **f.seek(pos)**

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> s = f.read()
>>> s
'Matricola,Esame,Voto,Data\n1892748,TDP,28,18/02/2020
\n1782932,DM,18,17/07/2021\n1720389,FI,26,17/03/2020\
n1702347,AD,30,16/06/2021\n1823643,ML,29,10/01/2020'
>>> s = f.read()
>>> s
''
>>> f.seek(0)
0
>>> l = f.readlines()
>>> for line in l:
        print(line)


Matricola,Esame,Voto,Data

1892748,TDP,28,18/02/2020

1782932,DM,18,17/07/2021

1720389,FI,26,17/03/2020

1702347,AD,30,16/06/2021

1823643,ML,29,10/01/2020
>>> l = f.readlines()
>>> l
[]
>>> f.close()
```

Leggere un file

- Se il file è troppo grande potrebbe non entrare in memoria
- In questo caso non posso leggere l'intero contenuto del file
- il metodo **readline( )** legge il file una riga alla volta

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> line = f.readline()
>>> while line != "":
        print(line)
        line = f.readline()


Matricola,Esame,Voto,Data

1892748,TDP,28,18/02/2020

1782932,DM,18,17/07/2021

1720389,FI,26,17/03/2020

1702347,AD,30,16/06/2021

1823643,ML,29,10/01/2020
>>> f.close()
>>>
```

File e Iteratori

- Il file object è iterabile (implementa il metodo **__iter__**)
- Posso leggere il file una riga alla volta semplicemente usando il ciclo for come avrei fatto con qualsiasi iterabile (lista, tupla, dizionario)

```
>>> f = open("voti_esame.txt","r",encoding="utf-8")
>>> for line in f:
        print(line)


Matricola,Esame,Voto,Data

1892748,TDP,28,18/02/2020

1782932,DM,18,17/07/2021

1720389,FI,26,17/03/2020

1702347,AD,30,16/06/2021

1823643,ML,29,10/01/2020
>>> f.close()
>>>
```

Scrivere su File

- Per scrivere su un file posso usare il metodo **f.write(string)**

- il metodo write prende in input una stringa, la scrive su file partendo dalla posizione corrente e restituisce il numero di caratteri scritti su file

```
>>> f1 = open("voti_esame.txt","r",encoding="utf-8")
>>> f2 = open("maggiori20.txt","w",encoding="utf-8")
>>> f2.write(f1.readline())
26
>>> for line in f1:
        linesplit = line.split(",")
        voto = int(linesplit[2])
        if voto > 20:
                f2.write(line)


26
25
25
24
>>> f2.close()
>>> f2 = open("maggiori20.txt","r",encoding="utf-8")
>>> for line in f2:
        print(line)


Matricola,Esame,Voto,Data

1892748,TDP,28,18/02/2020

1720389,FI,26,17/03/2020

1702347,AD,30,16/06/2021

1823643,ML,29,10/01/2020
>>> f2.close()
>>> f1.close()
>>>
```