



UNIVERSITÀ
DI PARMA

file

informatica e laboratorio di programmazione



- la maggior parte dei programmi ha necessità di memorizzare informazioni in modo ***persistente*** \Rightarrow non in memoria centrale
- le variabili e le strutture dati viste fino ad ora non sono persistenti
 - le informazioni vengono «***perse***» al termine dell'esecuzione
- ***file***: struttura di base per la memorizzazione persistente di informazioni
- molti linguaggi gestiscono i file come ***flussi*** di informazioni (***stream***)
 - i flussi di informazioni sono un'astrazione che può essere applicata non solo ai file ...

- ***stream***: astrazione per flussi di informazione
 - lettura o scrittura di informazioni su qualunque dispositivo I/O (file, ma non solo)
- ***file di testo***
 - varie codifiche (UTF-8 o altro)
 - conversioni automatiche, es. `"\n" → "\r\n"`
- ***file binari***
 - I/O preciso byte a byte, senza nessuna conversione
 - qualsiasi file... anche di testo!



- funzione ***open*** per accedere ad un file (di testo)
 - modalità scrittura o lettura o append: "***w***", o "***r***" o "***a***"
- ***scrittura*** su file: funzione **print**, o metodo **write**
- blocco **with**: chiude il file al termine delle operazioni
 - anche in caso di errore

```
with open("corsi.txt", "w") as f1:
    f1.write("Ingegneria dei Sistemi Informativi\n")
    f1.write("Ingegneria Informatica Elettronica e delle Telecomunicazioni\n")

with open("potenze.txt", "w") as f2:
    for x in range(10):
        print(x, x ** 2, file=f2)
```

```
# scrittura in un file di testo
endl = '\n'
myFile = open("mieparole.txt", "w")    #apertura file in scrittura
myFile.write("alfa"+endl)
myFile.write("beta"+endl)
myFile.write(str(123)+endl)
myFile.write(str(1.23)+endl)
myFile.write(str(True)+endl)
myFile.write(""+endl)
myFile.close()
```

```
# scrittura (append) in un file di testo
with open("mieparole.txt", "a") as myFile:
    myFile.write("nuova_riga"+endl)
```

```
with open("corsi.txt", "r") as f1:
    primo_corso = f1.readline() # le stringhe contengono '\n' finale
    secondo_corso = f1.readline()
    # alla fine del file viene letta una stringa vuota

with open("potenze.txt", "r") as f2:
    tutte_potenze = f2.read() # legge intero file in una stringa

with open("corsi.txt", "r") as f3:
    for linea in f3:
        # linea contiene '\n' finale
        # strip() rimuove spazi e \n iniziali e finali
        print(linea.strip(), ': ', len(linea))
```

```
# lettura di una singola riga dal file di testo
myFile = open("1000_parole_italiane_comuni.txt")
riga = myFile.readline()
print(riga, len(riga))
riga_pulita = riga.strip()
print(riga_pulita, len(riga_pulita))
myFile.close()
```

#apertura file
#lettura riga
#... compreso \n
#elimino spazi e \n
#riga 'pulita'
#chiusura file

```
# lettura di tutte le righe dal file di testo
myFile = open("1000_parole_italiane_comuni.txt")
for testo in myFile:
    print(testo.strip(), len(testo.strip()))
myFile.close()
```

#apertura file

#elimino spazi e \n
#chiusura file

```
# blocco with al termine effettua in automatico la chiusura del file
with open("1000_parole_italiane_comuni.txt", "r") as myFile:
    max_len = 0
    parola = ''
    for testo in myFile:
        if len(testo)-1 > max_len:
            max_len = len(testo)-1
            parola = testo.strip()
    print('parola più lunga', parola, max_len)
```

```
# lettura intero file in una stringa
with open("1000_parole_italiane_comuni.txt", "r") as myFile:
    tutto = myFile.read()
    print('tipo', type(tutto))
    print('lunghezza', len(tutto))
    print('contenuto: ----')
    print(tutto)
```


UTF-8 (Unicode Transformation Format, 8 bit) è una codifica di caratteri Unicode in sequenze di lunghezza variabile di byte, creata da Rob Pike e Ken Thompson. UTF-8 usa gruppi di byte per rappresentare i caratteri Unicode, ed è particolarmente utile per il trasferimento tramite sistemi di posta elettronica a 8-bit.

```
# lettura intero file in una stringa (codifica utf-8)
with open("1000_parole_italiane_comuni.txt", mode="r", encoding="utf-8") as myFile:
    tutto = myFile.read()
    print('tipo', type(tutto))
    print('lunghezza', len(tutto))
    print('contenuto: ----')
    print(tutto)
```

- se si verifica un ***errore*** (o una eccezione) Python normalmente si ***interrompe*** e comunica un ***messaggio*** di errore
- le ***eccezioni*** possono essere gestate mediante il costrutto try

```
# errore file
myFile = open('file_inesistente.txt')

# gestione eccezioni
try:
    myFile = open('file_inesistente.txt')
except:
    print('errore apertura file')
```

- il modulo **os** fornisce funzioni per interagire con il Sistema Operativo

```
import os
myDirectory = os.getcwd()
print('directory di lavoro', myDirectory)
```

<https://docs.python.it/html/tut/node12.html>

```
import os

def esplora(nome_directory):
    for nome in os.listdir(nome_directory):
        percorso = os.path.join(nome_directory, nome)
        if os.path.isfile(percorso):
            print(percorso)
        else:
            esplora(percorso)

esplora(".")
```