UNIVERSITÀ
DI PARMA

file

dati persistenti

- la maggior parte dei programmi ha necessità di memorizzare informazioni in modo ***persistente*** ⇒ 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
    - ai file su supporti di memoria non volatile
    - alla memoria centrale
    - a una periferica di I/O
    - a un dispositivo di rete

flussi di dati

- ***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!

scrittura su file

- 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 Gestionale\n")
    f1.write("Curriculum Produzione\n")
    f1.write("Curriculum Sostenibilità\n")


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

UNIVERSITÀ DI PARMA

esempi di scrittura

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

```
# scrittura (append) in un file di testo
endl = '\n'
with open('esempio.txt','a') as f:
    f.write('nuova riga'+endl)
```

UNIVERSITÀ DI PARMA

lettura da file

```
with open('corsi.txt','r') as f:
    laurea = f.readline()
    curriculum1 = f.readline()
    curriculum2 = f.readline()
print(laurea, curriculum1, curriculum2)

with open('potenze.txt','r') as pot:
    tutto = pot.read()
print(tutto)

with open('corsi.txt','r') as f2:
    for riga in f2:
        print(riga.strip(),' : ',len(riga))
```

UNIVERSITÀ DI PARMA

esempi di lettura (1)

```
# lettura di una singola riga dal file di testo

myFile = open('1000_parole_italiane_comuni.txt') #apertura file
riga = myFile.readline()                         #lettura riga
print(riga,len(riga))                            #compreso \n
riga_pulita = riga.strip()                       #elimina spazi e \n
print(riga_pulita,len(riga_pulita))              #riga 'pulita'
myFile.close()                                   #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()
```

UNIVERSITÀ DI PARMA

esempi di lettura (2)

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

UNIVERSITÀ DI PARMA

esempi di lettura (3)

```
# ricerca la parole più lunga
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)
```

gestione errori

- 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')
```

UNIVERSITÀ DI PARMA

modulo os (esempi)

- 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("…")
```

UNIVERSITÀ DI PARMA

file csv - lettura

- il modulo csv è integrato nella libreria standard di Python
- fornisce funzionalità per leggere e scrivere dati tabulari in formato CSV (Comma Separated Values)
- lettura della matrice m dal file 'valori.csv'

```
import csv
m = []
with open("valori.csv") as f:
    reader = csv.reader(f).          # reader funzione del modulo per la lettura
    # column_names = next(reader)  # se è presente una prima riga particolare
    for riga in reader: #riga è una lista di stringhe
        m.append(riga)
print(m)
```

UNIVERSITÀ DI PARMA

file csv - scrittura

```
with open("valori-s.csv", "w") as f:
    writer = csv.writer(f)
    for riga in m:
        writer.writerow(riga)  # la lista è trasformata in una stringa con separatori
```

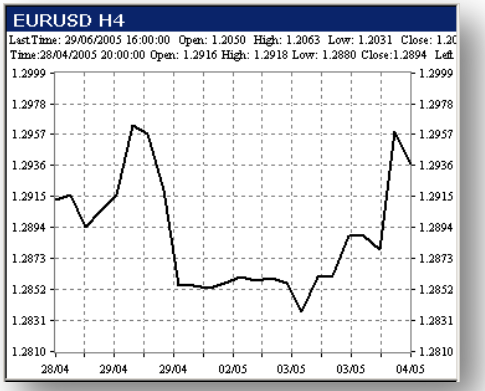
UNIVERSITÀ
DI PARMA

file

esercizi

sequenza di valori

- funzione che:
  - riceve come ***parametro*** il ***nome*** di un ***file***
  - restituisce una ***tupla*** con il valore ***massimo*** e quello ***minimo*** trovati nel file
    - ciascuna riga del file contiene un valore float
- *invocare la funzione con un nome di file inserito dall'utente*
- *visualizzare il risultato*



fusione

- due file di testo contengono ***sequenze di numeri***
  - un valore per ogni riga
  - in ciascun file, i valori sono già ***ordinati***
- generare in output i valori di entrambi i file
  - sequenza di output ordinata

ciclicamente, confrontare la coppia dei primi valori
(ciascuno proveniente da uno dei due stream)
scrivere il minore dei due sul file di uscita
non estrarre un nuovo valore da uno stream,
se quello precedente non è ancora stato scritto in output

righe dispari

- visualizzare tutte le righe dispari del file ‘divina.txt’
- visualizzare tutte le righe del file ‘divina.txt’ che contengono un numero pari di caratteri (contare solo i caratteri alfabetici)
- visualizzare tutto il contenuto del file ‘divina.txt’