Esercizi propedeutici sui files in Python

Anna Ficotto

1 Scrivere in un file

Consegna Crea un file chiamato `saluto.txt` e scrivi al suo interno la frase "Ciao mondo!".
Soluzione

```
file = open("saluto.txt", "w", encoding="utf-8")
file.write("Ciao mondo!")
file.close()
```

Spiegazione `open("saluto.txt", "w", encoding="utf-8")` apre (o crea) il file in modalità scrittura (`"w"`). Se il file non esiste, viene creato. Se esiste già, il contenuto viene cancellato. `encoding="utf-8"` assicura che i caratteri speciali (es: à, è, ò...) vengano gestiti correttamente. `file.write("Ciao mondo!")` scrive nel file la stringa desiderata. `file.close()` chiude il file per salvare correttamente il contenuto e liberare risorse.

2 Leggere il contenuto di un file

Consegna Leggi e stampa il contenuto del file `saluto.txt`.
Soluzione

```
file = open("saluto.txt", "r", encoding="utf-8")
contenuto = file.read()
print(contenuto)
file.close()
```

Spiegazione `open("saluto.txt", "r", encoding="utf-8")` apre il file in modalità lettura (`"r"`). `file.read()` legge tutto il contenuto del file come una singola stringa. Il contenuto viene salvato nella variabile contenuto. `print(contenuto)` lo stampa a video. Infine, `file.close()` chiude il file.

3 Scrivere più righe in un file

Consegna Scrivi una lista di 5 nomi (uno per riga) nel file `nomi.txt`.
Soluzione

```
file = open("nomi.txt", "w", encoding="utf-8")
file.write("Anna\n")
file.write("Marco\n")
file.write("Lucia\n")
file.write("Davide\n")
file.write("Sara\n")
file.close()
```

Spiegazione `open("nomi.txt", "w", encoding="utf-8")` crea o sovrascrive il file. Ogni `file.write(...)` scrive un nome e va a capo grazie a `\n`, che indica una nuova riga.

4 Stampare nomi in maiuscolo

Consegna Leggi tutti i nomi da `nomi.txt` e stampali in maiuscolo, uno per riga.
Soluzione

```
file = open("nomi.txt", "r", encoding="utf-8")
for riga in file:
    print(riga.strip().upper())
file.close()
```

Spiegazione Apriamo il file in lettura. Il ciclo `for riga in file` legge il file riga per riga. `riga.strip()` rimuove eventuali spazi o `\n` finali. `upper()` trasforma la stringa in maiuscolo. Poi, stampa il nome trasformato.
Esempio: "Anna`\n`" diventa "Anna" e poi ancora "ANNA".

5 Aggiungere un nome in fondo al file

Consegna Aggiungi il nome "Francesco" in fondo al file `nomi.txt`, senza cancellare i nomi già presenti.
Soluzione

```
file = open("nomi.txt", "a", encoding="utf-8")
file.write("Francesco\n")
file.close()
```

Spiegazione Usiamo la modalità `"a"` (*append*, cioè aggiunta): tutto ciò che scriviamo viene messo in fondo. `file.write("Francesco\n")` aggiunge una riga con il nuovo nome, e non viene cancellato nulla del contenuto precedente. `\n` assicura che venga inserito su una nuova riga.

6 Contare il numero di righe in un file

Consegna Apri il file `nomi.txt` e conta quante righe contiene. Stampa il risultato.
Soluzione

```
file = open("nomi.txt", "r", encoding="utf-8")
conta = 0
for riga in file:
    conta += 1
file.close()
print("Numero di righe:", conta)
```

Spiegazione: Apriamo il file in lettura, perché dobbiamo solo leggere il suo contenuto. Creiamo una variabile contatore (`conta`) e la inizializziamo a zero. Serve per tenere traccia del numero di righe. Con un ciclo `for riga in file` leggiamo riga per riga. Ogni iterazione del ciclo corrisponde a una riga. Ad ogni riga trovata, aumentiamo `conta` di 1. Alla fine, stampiamo il valore totale con `print`. Chiudiamo il file per buona pratica.

7 Contare quante volte appare una parola

Consegna Nel file `testo.txt` è presente un testo qualsiasi. Conta quante volte compare la parola esatta *"sole"*.
Soluzione

```
file = open("testo.txt", "r", encoding="utf-8")
conta = 0
for riga in file:
    parole = riga.split()
    for parola in parole:
        if parola == "sole":
            conta += 1
file.close()
print("La parola 'sole' appare:", conta, "volte")
```

Spiegazione Apriamo il file in modalità lettura e creiamo una variabile `conta` per contare le volte in cui compare la parola. Con il ciclo leggiamo una riga alla volta. Usando `riga.split()`, dividiamo la riga in parole, in pratica questa funzione separa le parole in base agli spazi. Poi eseguiamo un secondo ciclo su tutte le parole trovate in quella riga. Se una parola è esattamente uguale a "sole", aumentiamo il contatore. Alla fine del file, `conta` conterrà il numero totale di volte in cui è apparsa la parola.

8 Somma dei numeri in un file

Consegna Nel file `numeri.txt` ci sono numeri interi, uno per riga. Calcola e stampa la loro somma.
Soluzione

```
file = open("numeri.txt", "r", encoding="utf-8")
somma = 0
for riga in file:
    numero = int(riga.strip())
    somma += numero
file.close()
print("Somma dei numeri:", somma)
```

Spiegazione Apriamo il file in lettura. Inizializziamo `somma = 0`, che sarà il nostro accumulatore. Per ogni riga nel file, usiamo `strip()` per rimuovere il `\n` finale e convertiamo la riga da stringa a intero con `int(...)`. Aggiungiamo il numero alla somma. Dopo aver letto tutto il file, stampiamo la somma finale.

9 Scrivere una tabella dei quadrati

Consegna Crea un file `quadrati.txt` e scrivi per ogni numero da 1 a 10 il suo quadrato, su righe del tipo:

1 1
2 4
3 9
4 16
5 25

Soluzione

```
file = open("quadrati.txt", "w", encoding="utf-8")
for i in range(1, 11):
    file.write(str(i) + " " + str(i * i) + "\n")
file.close()
```

Spiegazione Apriamo il file in scrittura e usiamo `range(1, 11)` per generare i numeri da 1 a 10. Per ogni numero `i`, calcoliamo il quadrato con `i * i`. Scriviamo nel file la riga usando `str(i) + " " + str(i*i)`, cioè due numeri separati da spazio, poi `\n` per andare a capo.

10 Salvare solo le righe con una certa parola

Consegna Dal file `frasi.txt`, copia nel file `con_sole.txt` solo le righe che contengono la parola *"sole"*. **Soluzione**

```
origine = open("frasi.txt", "r", encoding="utf-8")
destinazione = open("con_sole.txt", "w", encoding="utf-8")

for riga in origine:
    if "sole" in riga:
        destinazione.write(riga)

origine.close()
destinazione.close()
```

Spiegazione Il nostro obiettivo è filtrare un file, copiando solo alcune righe in base a un criterio. Abbiamo due file: `frasi.txt`, che apriamo in lettura e da dove preleviamo le frasi, e `con_sole`, che apriamo in scrittura e dove salveremo solo le righe con la parola *'sole'*.
Per ogni riga nel file sorgente (`frasi.txt`), verifichiamo se contiene la parola sole. Se sì, la scriviamo così come'è nel nuovo file

11 Scrivere in un file CSV

N.B. Un file con estensione .csv è un file di tipo *Comma Separated Value*, cioè un file che contiene dei valori di vario tipo separati da virgole. **Consegna** Crea un file `voti.csv` in cui ogni riga contiene un nome e un voto separati da virgola. Scrivi almeno 5 righe. **Soluzione**

```
file = open("voti.csv", "w", encoding="utf-8")
file.write("Anna,8\n")
file.write("Marco,6\n")
file.write("Sara,9\n")
file.write("Luca,7\n")
file.write("Giulia,10\n")
file.close()
```

Spiegazione Il nostro obiettivo è creare un file CSV, in cui i dati sono separati da virgola. Apriamo, quindi, un nuovo file `voti.csv` in modalità `'w'`. Ogni `write()` scrive una riga nel file composta da un nome, una virgola, un voto e il carattere `\n` per andare a capo.

12 Calcolare la media dei voti da un CSV

Consegna Leggi il file `voti.csv` e calcola la media dei voti.
Soluzione 1 (senza funzioni max() e min())

```
file = open("voti.csv", "r", encoding="utf-8")
somma = 0
conteggio = 0

for riga in file:
    parti = riga.strip().split(",")
    voto = int(parti[1])
    somma += voto
    conteggio += 1

file.close()
media = somma / conteggio
print("Media voti:", media)
```

Spiegazione Apriamo il file `voti.csv` in modalità `'r'`. Inizializziamo due variabili: `somma` per sommare i voti e `conteggio` per contarli.
Per ogni riga rimuoviamo \n con `strip()`, dividiamo la riga stessa in una lista (`[nome, voto]`), convertiamo il voto in `int` e lo sommiamo alla variabile `somma` e infine aumentiamo `conteggio` di 1.
Dopo il ciclo, calcoliamo la media con `somma / conteggio`

13 Trovare lo studente con il voto più alto

Consegna Leggi il file `voti.csv` e stampa il nome dello studente con il voto più alto.
Soluzione

```
file = open("voti.csv", "r", encoding="utf-8")
massimo = -1
nome_top = ""

for riga in file:
    parti = riga.strip().split(",")
    nome = parti[0]
    voto = int(parti[1])
    if voto > massimo:
        massimo = voto
        nome_top = nome

file.close()
print("Il voto più alto è di", nome_top, "con", massimo)
```

Spiegazione Innanzitutto inizializziamo la variabile `massimo` a -1 come voto più alto iniziale (poiché i voti sono maggiori o uguali a 0[1]) e la variabile `nome_top` a una stringa vuota per salvare il nome corrispondente.
Dividiamo ogni riga in parole (sempre con `split(",")`), convertiamo il voto in `int` e, se il voto è maggiore del massimo attuale, aggiorniamo sia `massimo` sia `nome_top`. Infine, stampiamo il risultato alla fine del file.

[1] In generale, però, è sempre meglio inizializzare le variabili che contengono il massimo e il minimo al primo elemento analizzato.

14 Copiare solo alcuni elementi in un nuovo file

Consegna Dal file `voti.csv`, copia solo gli studenti con voto maggiore o uguale a 8 nel file `promossi.csv`. **Soluzione**

```
voti = open("voti.csv", "r", encoding="utf-8")
promossi = open("promossi.csv", "w", encoding="utf-8")

for riga in voti:
    parti = riga.strip().split(",")
    voto = int(parti[1])
    if voto >= 8:
        promossi.write(riga)

voti.close()
promossi.close()

```

Spiegazione Dobbiamo creare un nuovo file filtrato in base ad una condizione sui dati. Apriamo il file con i voti in lettura, e il nuovo file `promossi.csv` in scrittura. Per ogni riga dividiamo la riga stessa in parole, convertiamo il voto in intero e poi, se il voto è maggiore o uguale a 6, scriviamo la riga nel nuovo file con i promossi.

15 Aggiungere una riga in fondo a un CSV

Consegna Aggiungi lo studente "Matteo" con voto 7 in fondo al file `voti.csv` (senza, quindi, cancellare le altre righe). **Soluzione**

```
file = open("voti.csv", "a", encoding="utf-8")
file.write("Matteo,7\n")
file.close()
```

Spiegazione Apriamo il file in modalità `"a"` (append), che permette di scrivere in fondo al file.
Con write(), scriviamo la nuova riga nel formato Nome,Voto\n.

16 Contare il numero totale di parole in un file

Consegna Leggi un file `testo.txt` e conta tutte le parole contenute, anche se ripetute. Ogni riga può avere una o più parole.
Soluzione

```
file = open("testo.txt", "r", encoding="utf-8")
conta = 0
for riga in file:
    parole = riga.strip().split()
    for parola in parole:
        conta += 1
file.close()
print("Numero totale di parole:", conta)
```

Spiegazione Apriamo il file in modalità lettura e inizializziamo `conta`, il contatore globale delle parole, a 0.
Cicliamo riga per riga. Usiamo `strip()` per pulire la riga da spazi laterali e \n, e `split()` per dividere il testo in parole (separate da spazi).
Usiamo un secondo ciclo `for` per scorrere ogni parola in quella riga. Per ogni parola trovata, aumentiamo il contatore.
Una volta arrivati alla fine del file, usciamo dai cicli e stampiamo `conta`.

17 Contare le parole che iniziano con una certa lettera

Consegna Conta quante parole iniziano con la lettera "A" (maiuscola) nel file `parole.txt`.
Soluzione

```
file = open("parole.txt", "r", encoding="utf-8")
conta = 0
for riga in file:
    parole = riga.strip().split()
    for parola in parole:
        if parola.startswith("A"):
            conta += 1
file.close()
print("Parole che iniziano con 'A':", conta)
```

Spiegazione Apriamo il file in lettura e inizializziamo il contatore `conta` a 0.
Per ogni riga puliamo e dividiamo in parole (come nell'esercizio precedente), controlliamo ogni parola usando startswith("A") e se una parola comincia con "A", aumentiamo il contatore.
Stampiamo il numero totale alla fine.

18 Copiare il contenuto da un file all'altro

Consegna Leggi il contenuto di `origine.txt` e copialo esattamente in `copia.txt`.
Soluzione

```
origine = open("origine.txt", "r", encoding="utf-8")
copia = open("copia.txt", "w", encoding="utf-8")

for riga in origine:
    copia.write(riga)

origine.close()
copia.close()
```

Spiegazione Apriamo il file di partenza `origine.txt` in lettura e un nuovo file `copia.txt` in scrittura (se esiste già, sarà sovrascritto).
Con un ciclo `for`, leggiamo una riga alla volta da `origine`. Usiamo `write(riga)` per scrivere la stessa riga nel file `copia`.
N.B. Questo esercizio ci mostra come funziona un copia-incolla programmato tra due file.

19 Verificare se un file esiste

Consegna Controlla se il file `dati.txt` esiste. Se sì, leggilo e stampane il contenuto. Se no, stampa un messaggio d'errore.
Soluzione

```
try:
    file = open("dati.txt", "r", encoding="utf-8")
    contenuto = file.read()
    print("Contenuto del file:\n", contenuto)
    file.close()
except:
    print("Errore: il file 'dati.txt' non esiste.")
```

Spiegazione Usiamo `try/except` per gestire un possibile errore di apertura.
Se il file esiste viene aperto correttamente e il contenuto viene letto tutto con `read()` e viene stampato.
Se il file non esiste, Python lancia un'eccezione e viene eseguita l'istruzione nel blocco `except`.

20 Leggere solo nomi con età maggiore di 18

Consegna Leggi il file `persone.csv` dove ogni riga è: `Nome,Età`. Stampa solo i nomi di chi ha più di 18 anni.
Soluzione

```
file = open("persone.csv", "r", encoding="utf-8")

for riga in file:
    parti = riga.strip().split(",")
    nome = parti[0]
    età = int(parti[1])
    if età > 18:
        print(nome)

file.close()
```

Spiegazione Apriamo il file CSV in lettura. Per ogni riga rimuoviamo spazi e `\n` con `strip()`. Separiamo nome e età con `split(",")`. Convertiamo l'età in intero con `int(...)`. Se l'età è maggiore di 18, stampiamo il nome.

21 Ordinare righe di un file alfabeticamente

Consegna Hai un file `nomi.txt` che contiene un nome per riga. Ordina alfabeticamente i nomi e scrivili in un nuovo file chiamato `ordinati.txt`.
Soluzione

```
# Lettura dei nomi
file = open("nomi.txt", "r", encoding="utf-8")
lista_nomi = []
for riga in file:
    lista_nomi.append(riga.strip())
file.close()

# Ordinamento manuale (bubble sort)
for i in range(len(lista_nomi)):
    for j in range(i + 1, len(lista_nomi)):
        if lista_nomi[i] > lista_nomi[j]:
            temp = lista_nomi[i]
            lista_nomi[i] = lista_nomi[j]
            lista_nomi[j] = temp

# Scrittura su nuovo file
file = open("ordinati.txt", "w", encoding="utf-8")
for nome in lista_nomi:
    file.write(nome + "\n")
file.close()
```

Spiegazione Prima leggiamo tutti i nomi e li salviamo in una lista, poi usiamo un algoritmo di ordinamento (bubble sort semplificato) per metterli in ordine alfabetico, e alla fine scriviamo i nomi ordinati nel file `ordinati.txt`.

22 Trovare la riga più lunga in un file

Consegna Scrivi un programma che legge un file `testo.txt` e stampa la riga più lunga (in termini di numero di caratteri).
Soluzione

```
file = open("testo.txt", "r", encoding="utf-8")
massima_lunghezza = 0
riga_lunga = ""

for riga in file:
    lunghezza = len(riga.strip())
    if lunghezza > massima_lunghezza:
        massima_lunghezza = lunghezza
        riga_lunga = riga.strip()

file.close()
print("Riga più lunga:", riga_lunga)
```

Spiegazione Usiamo una variabile per tenere la lunghezza massima trovata (`massima_lunghezza`) e una per memorizzare la riga (**riga_lunga**). Per ogni riga letta calcoliamo la lunghezza (dopo aver tolto il `\n`). Se è più lunga di tutte le precedenti, la memorizziamo. Alla fine stampiamo la riga trovata.

23 Conta parole distinte

Consegna Leggi un file `testo.txt` e conta quante parole distinte contiene, ignorando maiuscole/minuscole.
Soluzione

```
file = open("testo.txt", "r", encoding="utf-8")
parole_distinte = []

for riga in file:
    parole = riga.strip().split()
    for parola in parole:
        parola = parola.lower()
        if parola not in parole_distinte:
            parole_distinte.append(parola)

file.close()
print("Numero di parole distinte:", len(parole_distinte))
```

Spiegazione Creiamo una lista vuota `parole_distinte`. Per ogni parola del file convertiamo in minuscolo la parola stessa (`lower()`) per uniformare. Se non è già presente nella lista, la aggiungiamo. Alla fine, la lunghezza della lista ci dice quante parole diverse ci sono nel testo.

24 Trova il numero che appare più volte

Consegna Nel file `numeri.txt` c'è un numero per riga. Trova quale numero compare più volte e quante volte.
Soluzione

```python
file = open("numeri.txt", "r", encoding="utf-8")
numeri = []
frequenze = []

for riga in file:
    numero = int(riga.strip())
    trovato = False
    for i in range(len(numeri)):
        if numeri[i] == numero:
            frequenze[i] += 1
            trovato = True
            break
    if not trovato:
        numeri.append(numero)
        frequenze.append(1)

file.close()

# Trova la frequenza massima
massimo = 0
indice = 0
for i in range(len(frequenze)):
    if frequenze[i] > massimo:
        massimo = frequenze[i]
        indice = i

print("Numero più frequente:", numeri[indice])
print("Frequenza:", massimo)
```

Spiegazione Usiamo due liste parallele: `numeri` contiene tutti i numeri diversi, mentre `frequenze` tiene traccia di quante volte ciascun numero compare. Per ogni riga cerchiamo se il numero è già in `numeri`, e se sì, aumentiamo il contatore. Se no, lo aggiungiamo con frequenza 1. Alla fine, cerchiamo l'indice con la frequenza più alta.

25 File con tabella: calcolo medie riga per riga

Consegna Nel file `tabella.txt`, ogni riga contiene numeri separati da spazio. Calcola la media dei valori di ogni riga e salvala in `medie.txt`, una per riga.
Soluzione

```
ingresso = open("tabella.txt", "r", encoding="utf-8")
uscita = open("medie.txt", "w", encoding="utf-8")

for riga in ingresso:
    numeri = riga.strip().split()
    somma = 0
    quanti = 0
    for n in numeri:
        somma += int(n)
        quanti += 1
    media = somma / quanti
    uscita.write(str(media) + "\n")

ingresso.close()
uscita.close()
```

Spiegazione Leggiamo il file `tabella.txt` riga per riga. Ogni riga ha numeri separati da spazi, quindi li dividiamo con split(). Sommiamo tutti i numeri della riga e contiamo quanti sono. Calcoliamo la media e la scriviamo nel file `medie.txt`.

26 Generare un file con numeri casuali in due intervalli

Consegna Scrivi una funzione `makeFile(nome, n, a, b, c)` che:

- Crea un file con nome indicato dal parametro nome
- Il file conterrà esattamente n righe
- Ogni riga ha due numeri casuali interi:
  - Il primo compreso tra a e b
  - Il secondo compreso tra b e c
- I due numeri devono essere separati da almeno uno spazio

Per rendere i risultati ripetibili, la funzione deve iniziare con

```
random.seed(0)
```

Così, ogni volta che si esegue, verranno generati gli stessi numeri casuali.
Soluzione

```
import random

def makeFile(nome, n, a, b, c):
    random.seed(0)
    file = open(nome, "w", encoding="utf-8")

    for i in range(n):
        primo = random.randint(a, b)
        secondo = random.randint(b, c)
        file.write(str(primo) + " " + str(secondo) + "\n")

    file.close()

# Esempio di chiamata
makeFile("out.txt", 5, 1, 10, 20)
```

Spiegazione Innanzitutto importiamo la libreria `random` perché vogliamo generare numeri casuali. `random.seed(0)` serve ad inizializzare il generatore casuale in modo che dia sempre gli stessi numeri (funzione utile nei test o nei compiti).
Apriamo il file in modalità scrittura ("w") con il nome ricevuto come parametro `nome`. Per `n` volte (quindi per il numero di righe desiderato):

- generiamo `primo` con `random.randint(a, b)` (numero tra `a` e `b` inclusi);
- generiamo `secondo` con `random.randint(b, c)` (numero tra `b` e `c` inclusi);
- scriviamo la riga nel file, separando i due numeri con uno spazio.

Alla fine del programma chiudiamo il file.

27 Calcolo delle medie dal file generato

Consegna Usando il file generato con makeFile, scrivi un programma che lo legge e calcola la media del primo numero di ogni riga e del secondo numero di ogni riga.
Il file contiene due numeri interi separati da spazio in ogni riga. Puoi usare split() per separarli.
Soluzione

```
file = open("out.txt", "r", encoding="utf-8")

somma_primo = 0
somma_secondo = 0
righe = 0

for riga in file:
    parti = riga.strip().split()
    primo = int(parti[0])
    secondo = int(parti[1])

    somma_primo += primo
    somma_secondo += secondo
    righe += 1

file.close()

media1 = somma_primo / righe
media2 = somma_secondo / righe

print("Media del primo numero:", media1)
print("Media del secondo numero:", media2)
```

Spiegazione Apriamo il file in lettura. Prepariamo tre variabili:

- `somma_primo` → per sommare tutti i primi numeri
- `somma_secondo` → per sommare tutti i secondi numeri
- `righe` → per contare quante righe abbiamo letto

Per ogni riga:

- `strip()` rimuove il `\n` finale
- `split()` divide la riga in una lista di due elementi (le due parole/numeri)
- convertiamo ogni parte in intero

Successivamente sommiamo i valori, riga per riga.
Dopo aver letto tutto, calcoliamo:

- `media1 = somma_primo / righe`
- `media2 = somma_secondo / righe`

Infine stampiamo i risultati.

28 Analisi completa avanzata di un file CSV

Consegna Hai un file gara.csv con i risultati di una competizione organizzato come segue:

```
Nazione,Oro,Argento,Bronzo
Canada,1,0,1
Cina,1,1,0
Germania,0,0,1
Corea del Sud,1,0,0
Giappone,0,1,1
Russia,0,1,1
Stati Uniti d'America,1,1,0
```

Scrivi una funzione `medaglie(f)` che:

- Legge il file CSV usando la virgola come separatore
- Salva le nazioni in una lista chiamata nazioni
- Salva le medaglie (oro, argento, bronzo) in una matrice chiamata `counts`
- Calcola:
  il totale di ogni tipo di medaglia (sommando tutte le colonne oro, argento, bronzo)
  il totale di medaglie per ogni nazione (somma per riga)
- Scrive i totali per nazione nel file `totali.txt`, uno per riga
- Ritorna dalla funzione una lista con i 3 totali: oro, argento, bronzo

Soluzione

```python
def medaglie(f):
    file = open(f, "r", encoding="utf-8")
    nazioni = []
    counts = []

    prima = True
    for riga in file:
        if prima:
            prima = False  # Saltiamo intestazione
            continue
        parti = riga.strip().split(",")
        nazioni.append(parti[0])
        oro = int(parti[1])
        argento = int(parti[2])
        bronzo = int(parti[3])
        counts.append([oro, argento, bronzo])

    file.close()

    # Calcolo totali per tipo di medaglia
    totale_oro = 0
    totale_argento = 0
    totale_bronzo = 0
    for riga in counts:
```

```
25          totale_oro += riga[0]
26          totale_argento += riga[1]
27          totale_bronzo += riga[2]
28
29      # Scrittura dei totali per nazione su file
30      out = open("totali.txt", "w", encoding="utf-8")
31      for i in range(len(nazioni)):
32          totale_nazione = counts[i][0] + counts[i][1] + counts[i][2]
33          out.write(nazioni[i] + ": " + str(totale_nazione) + "\n")
34      out.close()
35
36      return [totale_oro, totale_argento, totale_bronzo]
```

Spiegazione Apriamo il file e leggiamo le righe. La prima riga (intestazione) viene saltata con una variabile `prima`. Per ogni riga:

- Dividiamo con `split(",")` per ottenere una lista di elementi
- Salviamo il nome della nazione
- Convertiamo i numeri in interi e li mettiamo in `counts`, una lista di liste

Per calcolare i totali per tipo di medaglia usiamo tre variabili `totale_oro`, `totale_argento`, `totale_bronzo` e sommiamo gli elementi corrispondenti per ogni riga.
Per ogni nazione, sommiamo le sue medaglie e scriviamo una riga nel file `totali.txt`.
Infine, ritorniamo una lista con i 3 totali globali.

28.1 Variante senza virgole: `medagliereSuper(f)`

Consegna Scrivi una versione della funzione chiamata `medagliereSuper(f)` che fa esattamente le stesse cose, ma il file di input (es. `gara.txt`) non usa virgole: gli elementi sono separati da spazi o più spazi.
Soluzione

```python
def medagliereSuper(f):
    file = open(f, "r", encoding="utf-8")
    nazioni = []
    counts = []

    prima = True
    for riga in file:
        if prima:
            prima = False
            continue
        parole = riga.strip().split()
        # La nazione può avere più parole: tutto tranne le ultime 3
        nome_nazione = " ".join(parole[:-3])
        oro = int(parole[-3])
        argento = int(parole[-2])
        bronzo = int(parole[-1])
        nazioni.append(nome_nazione)
        counts.append([oro, argento, bronzo])

    file.close()

    # Calcolo totali per tipo di medaglia
    totale_oro = 0
    totale_argento = 0
    totale_bronzo = 0
    for riga in counts:
        totale_oro += riga[0]
        totale_argento += riga[1]
        totale_bronzo += riga[2]

    # Scrittura dei totali per nazione su file
    out = open("totali.txt", "w", encoding="utf-8")
    for i in range(len(nazioni)):
        totale_nazione = counts[i][0] + counts[i][1] + counts[i][2]
        out.write(nazioni[i] + ": " + str(totale_nazione) + "\n")
    out.close()

    return [totale_oro, totale_argento, totale_bronzo]
```

Spiegazione La differenza rispetto all'esercizio precedente è che qui i campi non sono separati da virgole ma da spazi, quindi usiamo `split()` da solo.
Problema: alcune nazioni hanno più parole ("Stati Uniti d'America"), quindi il nome

della nazione non è una sola parola.
Soluzione: prendiamo tutti gli elementi tranne gli ultimi 3 (`parole[:-3]`) come nome della nazione. Gli ultimi 3 sono sempre le medaglie: oro, argento, bronzo.
Il resto del codice è identico al precedente: somma dei totali per tipo e scrittura su `totali.txt`.