

# Programming Lab

## Lezione 3

# Note pratiche

**Prima ora:** parte pratica, assicuriamoci di avere tutti l'ambiente che funziona e vediamo un primo esercizio

**Seconda ora:** Introduzione alla gestione dei files con Python + esercizio

# Un primo esercizio

Scrivete una funzione che sommi  
tutti gli elementi di una lista\*.

Poi, committate il file in cui l'avete scritta.

\* se non sapete fare qualcosa, usate Google (o Bing o il vostro motore di ricerca preferito), è parte del corso anche che impariate a destreggiarvi!

# Soluzione

```
def list_sum(the_list):  
    sum=0  
    for item in the_list:  
        sum = sum + item  
    print("Somma: {}".format(sum))  
  
list_sum([1,4,10])
```

```
> python intro.py  
Somma: 15
```

# Soluzione con il “return”

```
def list_sum(the_list):  
    sum=0  
    for item in the_list:  
        sum = sum + item  
    return sum  
  
print("Somma: {}".format(list_sum([1,4,10])))
```

```
> python intro.py  
Somma: 15
```

# I files

I files sono molto comodi per salvare dati ancora prima dei database.

Uno dei formati più standard è il CSV, ovvero “Comma-Separated Values”.

In genere, in un file CSV ogni riga è una “entry”, e ci può essere una intestazione (opzionale) di una o più righe. Estensione **.csv** o alle volte **.txt**

Esempio “shampoo\_sales.csv”:

```
Date,Sales
01-01-2012,266.0
01-02-2012,145.9
01-03-2012,183.1
01-04-2012,119.3
```

# Interagire con i files in Python

Come si farà...?

# Interagire con i files in Python

A screenshot of a Google search interface. The search bar at the top contains the text "python read file". Below the search bar, there are navigation links: "All", "Videos", "Books", "News", "Images", "More", "Settings", and "Tools". The search results are displayed below, showing three entries:

About 424,000,000 results (0.42 seconds)

[www.w3schools.com](#) › [python](#) › [python\\_file\\_open](#) ▾  
**Python File Open - W3Schools**  
f = open("demofile.txt", "r") print(f.read()) Open a file on a different location: f = open("D:\\myfiles\\welcome.txt", "r") Return the 5 first characters of the file: Read one line of the file: Read two lines of the file: Loop through the file line by line: Close the file when you are finish with it:

[docs.python.org](#) › [tutorial](#) › [inputoutput](#) ▾  
**7. Input and Output — Python 3.9.0 documentation**  
Normally, files are opened in text mode, that means, you read and write strings from and to the file, which are encoded in a specific encoding. If encoding is not ...

[www.guru99.com](#) › [reading-and-writing-files-in-python](#) ▾  
**Python File Handling: Create, Open, Append, Read, Write**  
Oct 23, 2020 — Summary. Python allows you to read, write and delete files. Use the function open("filename","w+") to create a file. To append data to an existing file use the command open("Filename", "a") Use the read function to read the ENTIRE contents of a file. Use the readlines function to read the content of the file one by one ...



# Interagire con i files in Python

Si usa l'oggetto "file". Cosa è un oggetto lo vedremo nella prossima lezione, per ora prendiamolo così com'è.

```
my_file = open("shampoo_sales.csv", "r")  
print(my_file.read())  
my_file.close()
```

```
> python intro.py  
Date,Sales  
01-01-2012,266.0  
01-02-2012,145.9  
01-03-2012,183.1
```

...

Modalità di apertura del file, in questo caso "r" sta per "read". Se vorrò anche scriverci, userò "rw"

# Parentesi: “slicing” delle stringhe

Si può tagliare una fetta (“to *slice*”) di una stringa così:

```
mia_stringa[0:50]    # Dal primo al cinquantesimo carattere  
mia_stringa[30:50]   # Dal trentesimo al cinquantesimo carattere  
mia_stringa[0:-1]    # Dal primo al penultimo carattere
```

# Parentesi: “slicing” delle stringhe

Si può usare lo slicing delle stringhe per stampare solo un pezzetto del file:

```
my_file = open("shampoo_sales.txt", "r")  
print(my_file.read()[0:50])  
my_file.close()
```

```
> python intro.py  
Date,Sales  
01-01-2012,266.0  
01-02-2012,145.9  
01-03
```

# Parentesi: “slicing” delle stringhe

Si può usare lo slicing delle stringhe per stampare solo un pezzetto del file  
(versione più sofisticata)

```
# Apro il file
my_file = open("shampoo_sales.txt", "r")

# Leggo il contenuto
my_file_contents = my_file.read()

# Stampo a schermo i primi 50 caratteri
if len(my_file_contents) > 50:
    print(my_file_contents[0:50])
else:
    print(my_file_contents)

# Chiudo il file
my_file.close()
```

# Interagire con i files in Python

Il file si può anche leggere riga per riga, una alla volta:

```
my_file = open("shampoo_sales.csv", "r")  
print(my_file.readline())  
print(my_file.readline())  
my_file.close()
```

```
> python intro.py  
Date,Sales  
  
01-01-2012,266.0
```

# Interagire con i files in Python

Il file si può anche leggere riga per riga tutto in un colpo in modo “pythonico”:

```
my_file = open("shampoo_sales.csv", "r")

for line in my_file:
    print(line)

my_file.close()
```

```
> python intro.py
Date,Sales

01-01-2012,266.0
```

...

# Interagire con i files in Python

Scrivere su un file (nota il “w” nella funzione open):

```
my_file = open("saluti.txt", "w")  
  
my_file.write("Ciao mondo!")  
  
my_file.close()
```

..ma non lo useremo molto in questo corso.

# Leggere i valori di un file CSV

Per leggere i dati da un file CSV bisogna fare un po' di cose nuove:

1) Il metodo “.split” per separare le stringhe su uno specifico carattere:

```
mia_stringa = "Ciao, come va?"  
  
lista_elementi = mia_stringa.split(",")
```



# Leggere i valori di un file CSV

Per leggere i dati da un file CSV bisogna fare un po' di cose nuove:

2) La conversione di una stringa a valore numerico (floating point)

```
mia_stringa = "5.5"  
  
mio_numero = float(mia_stringa)
```

# Leggere i valori di un file CSV

Per leggere i dati da un file CSV bisogna fare un po' di cose nuove:

3) Sapere come aggiungere un elemento ad una lista

```
mia_lista = [1,2,3]  
mia_lista.append(4)
```

# Leggere i valori di un file CSV

```
# Inizializzo una lista vuota per salvare i valori
total_values = []

# Apro e leggo il file, linea per linea
my_file = open("shampoo_sales.csv", "r")
for line in my_file:

    # Faccio lo split di ogni riga sulla virgola
    elements = line.split(',')

    # Se NON sto processando l'intestazione...
    if elements[0] != 'Date':

        # Setto la data e il valore
        date = elements[0]
        value = elements[1]

        # Aggiungo alla lista dei valori questo valore
        total_values.append(float(value))
```

# Esercizio

Scrivete uno script che sommi  
tutti i valori delle vendite degli shampoo del file  
“shampoo\_sales.csv”

Poi, committate il file in cui l'avete scritto.

\* se non sapete fare qualcosa, usate Google (o Bing o il vostro motore di ricerca preferito), è parte del corso anche che impariate a destreggiarvi!