Esercitazione 9

PYTHON
METODI e FILE

Definire nuove funzioni (1)

- La sintassi è:

```
def funzione(arg1, arg2, opz1=val1, opz2=val2):
    ...
```

- Non bisogna specificare il tipo ritornato
  - L'equivalente delle funzioni "void" del C sono funzioni che ritornano "None"
  - Una funzione può ritornare un oggetto di qualsiasi tipo
- Gli argomenti sono normali variabili e possono essere in qualsiasi numero
  - Se la funzione non accetta argomenti basta usare una lista di argomenti vuota, ad esempio:

```
def foo():
    ...
```

Definire nuove funzioni (2)

- Gli argomenti opzionali possono non essere specificati dal chiamante, in questo caso assumono il valore di default
- Le variabili all'interno della funzione non sono visibili dall'esterno
- Esempio di utilizzo di una funzione:

```
>>> def foo(x, y, z=42, k=12):
...     print x, y, z, k
...
>>> foo(5, 3, k=9)
5 3 42 9
```

doc-string

- Le doc-string o stringhe di documentazione sono stringhe nella prima riga della funzione con lo scopo di documentarla

```
def foo():
    "Documentazione di foo"
```

- È possibile accedere alla doc-string con l'attributo "__doc__" della funzione
  - `print foo.__doc__` → Documentazione di foo
- Usata da tool per generare la documentazione
- Usata dalla funzione "help"
  - "help(foo)" stampa informazioni su "foo"

return

- La sintassi è:

```
return [valore_di_ritorno]
```

- Se il valore di ritorno viene omesso viene ritornato “None”
- Se il flusso del programma esce dalla funzione senza aver trovato un’istruzione “return” viene ritornato “None”
- Esempio:

```
def somma(a, b):
    return a + b
```

Metodi

```
def metodo(*args):
    print args

metodo(1,2,3,4,56)
>>(1, 2, 3, 4, 56)

def metodo(param1, param2, param3='abc'):
    print param1, param2, param3

metodo(1,2)
>>1 2 abc
metodo(1,2,4)
>>1 2 4
metodo(1)
>>TypeError: metodo() takes at least 2 arguments (1 given)
```

File

- I file vengono gestiti in modo molto semplice e simile al C
- “open(nomefile[, modo])” apre “nomefile” in modalità “modo” (“r” è il valore di default) e ritorna un oggetto di tipo “file”
- I metodi principali degli oggetti file sono:
  - “read([n])” ritorna “n” byte dal file. Se “n” è omesso legge tutto il file
  - “readline()” ritorna una riga
  - “readlines()” ritorna una lista con le righe rimanenti nel file
  - “write(str)” scrive “data” sul file

File

- “writelines(list)” scrive tutti gli elementi di “list” su file
- “close()” chiude il file (richiamato automaticamente
- dall’interprete)
- “flush()” scrive su disco i dati presenti in eventuali buffer
- “seek(offset[, posiz])” muove di “offset” byte da “posiz”. I valori di posiz sono:

  - 0: dall’inizio del file (valore di default)
    - 1: dalla posizione corrente
    - 2: dalla fine del file (“offset” è normalmente negativo)
- “tell()” ritorna la posizione corrente
- “truncate([n])” tronca il file a non più di “n” byte. Il valore di default è la posizione corrente

File

- Leggi_matrice.py
- Leggi_file.py
- Scrivere su file:

```
With open('file2.txt','w') as f:
    f.write('ciao')
```

Esercizi

- Scrivere un metodo che calcoli il prodotto scalare fra vettori
- Scrivere un metodo che legga due matrici da due file diversi (ogni riga del file contiene un vettore riga) ed un metodo che calcoli il prodotto tra le due matrici
- Scrivere un programma che risolva equazioni di secondo grado, controllando i casi che si vengono a creare a seconda del valore del Delta.

Esercizi

- Scrivere una funzione che legga da un file i voti assegnati ad un gruppo di studenti. Ogni riga contiene il nome dello studente seguito dai voti ottenuti agli esami.
- Si definisca la funzione di inserimento e cancellazione di un dato studente nella struttura dati appena creata.
- Si definisca una funzione che calcoli la media voto complessiva (i.e. calcolata su tutti gli studenti per tutti gli esami sostenuti).