Esercitazione di Programmazione

Michele Beretta

michele.beretta@unibg.it

Gruppo A	Gruppo B/C	Argomenti
12/03	13/03	Variabili, tipi base, operazioni e funzioni standard di I/O
17/03	20/03	Uso di funzioni e metodi, uso stringhe, conversioni di tipo ed espressioni booleane
16/04	17/04	Costrutto condizionale: `if-elif-else`, strutture iterative `while` e `for`
23/04	24/04	Esercizi su `if-elif-else`, uso dei cicli `while` e `for` (`break`, `continue` ed `else`)
07/05	08/05	Esercizi sui cicli `while` e `for`
14/05	15/05	Liste (introduzione ed esercizi)
21/05	22/05	Moduli e funzioni (introduzione ed esercizi)
27/05	29/05	File, complementi sulle stringhe (introduzione ed esercizi)
04/06	05/06	Dizionari ed esercizi di ripasso

Complementi sulla manipolazione di stringhe (1/2)

```
s.split(sep=' ')
```

Restituisce una lista contenente le sotto-stringhe separate da un certo carattere (se non specificato, lo spazio).

```
> s = 'Il mio cane si chiama: Fido.'

> s.split()
['Il', 'mio', 'cane', 'si', 'chiama:', 'Fido.']

> s.split(':')
['Il mio cane si chiama', ' Fido.']
```

Complementi sulla manipolazione di stringhe (2/2)

```
s.splitlines()
```

Restituisce una lista contenente le sotto-stringhe separate da un «a capo».

```
> s = 'Il testo che segue\nmostra il funzionamento\ndi splitlines.'
> print(s)
Il testo che segue
mostra il funzionamento
di splitlines.

> s.splitlines()
['Il testo che segue', 'mostra il funzionamento', 'di splitlines.']
```

File

I file devono essere **aperti** prima di leggere/scrivere, e **chiusi** prima di terminare il programma.

Ci sono diverse modalità: lettura (`r`), scrittura (`w`), append (`a`), etc.

```
file = open('nome', 'r')
contents = file.read()
file.close()
print(contents)
```

Si può usare with per gestire automaticamente apertura/chiusura. Gestisce in automatico anche errori.

```
with open('nome', 'r') as file:
    contents = file.read()

print(contents)
```

File – lettura riga per riga

Si può usare un ciclo `for` per leggere riga per riga.

```
with open('nome', 'r') as file:
    for line in file:
        print(line)
```

Alternativamente:

```
with open('nome', 'r') as file:
    line = file.readline()
    print(line)
    while len(line) != 0:
        line = file.readline()
        print(line)
```

Esempio

Scrivere un programma che presi due file copii il primo nel secondo.

```
def copia(dest, src):
    with open(src, 'r') as sorgente:
        with open(dest, 'w') as destinazione:
            for line in sorgente:
                destinazione.write(line)
```

Alternativamente

```
def copia(dest, src):
    sorgente = open(src, 'r')
    destinazione = open(dst, 'w')
    for line in sorgente:
        destinazione.write(line)
    sorgente.close()
    destinazione.close()
```

Esercizio 1

Scrivere un programma che legga un file `ruote.txt` contenete i nomi delle ruote del gioco del lotto e ne crei un altro, `estrazione.txt`, contenente di fianco ai nomi di ciascuna ruota anche 5 numeri estratti casualmente. Usare funzioni quando appropriato.

```
import random
num = random.randint(1, 90)
```

ruote.txt

```
BARI
CAGLIARI
FIRENZE
GENOVA
MILANO
NAPOLI
PALERMO
ROMA
```

```
TORINO
VENEZIA
NAZIONALE
```

estrazione.txt

```
BARI 26 55 30 41 15
CAGLIARI 83 44 43 29 37
FIRENZE 35 65 61 02 47
GENOVA 57 28 62 59 37
MILANO 62 42 56 12 21
NAPOLI 55 19 59 72 18
PALERMO 48 34 62 41 55
ROMA 81 53 50 27 51
TORINO 40 84 38 64 35
VENEZIA 05 28 52 03 53
NAZIONALE 39 03 84 75 26
```

Esercizio 2

Scrivere un programma Python che legga il file `estrazione.txt` creato precedentemente e riporti a video i valori numerici tra 1 e 90 **non in esso contenuti**. Quando appropriato definire opportune funzioni.

Esercizio 3

Scrivere un programma Python che legga il file `estrazione.txt` creato precedentemente e riporti a video ogni valore numerico tra 1 e 90 in esso contenuto **con il corrispondente numero di ripetizioni**. Quando appropriato definire opportune funzioni.

Esercizio 4

Scrivere un programma Python che chiesto il nome di un file di password all'utente e una stringa indicante un nome di persona, cerchi nel file le password di tutte le persone con il nome indicato e li riporti a video. Quando appropriato definire opportune funzioni.

Si assuma che il formato delle righe del file sia:

```
Cognome Nome : password
```

Estendere il programma con una funzione che valuti la robustezza di ogni password come

- *Debole*: se composta da almeno 8 caratteri alfanumerici
- *Media*: se composta da almeno 8 caratteri alfanumerici maiuscoli e minuscoli
- *Accettabile*: se composta da almeno 8 caratteri maiuscoli, minuscoli e di punteggiatura
- *Inaccettabile*: in tutti gli altri casi