Esercitazione di Laboratorio Informatica@DSS (2025/2026)

Massimo Lauria

Laboratorio 5 - 27/10/2025

Lo scopo del laboratorio è di esercitarsi e misurare la propria preparazione: gli esercizi non sono troppo difficili, se si sono seguite le lezioni. Non vi viene comunque messo alcun voto.

Modalità di lavoro: gli studenti devono lavorare in autonomia o in piccoli gruppi, senza disturbare i colleghi. Il lavoro di gruppo è fruttuoso solo se tutti partecipano e se ognuno scrive una propria soluzione per tutti gli esercizi.

Il docente cercherà per quanto possibile di non occupare il tempo del laboratorio per introdurre materiale nuovo, anche se a volte questo sarà necessario. Il docente è a disposizione per aiutare gli studenti, che possono iniziare a lavorare anche prima che il docente arrivi in aula, se lo desiderano

Raccomandazioni: leggete bene il testo degli esercizi prima di chiedere chiarimenti. In ogni caso sarò in aula con voi.

Uso dei file di test: per aiutarvi a completare questa esercitazione avete a disposizione dei programmi di test per testare la vostra soluzione. Questi sono simili a quelli che avrete in sede di esame, pertanto vi consiglio di imparate ad usarli. Per portare a termine l'esercizio è necessario

- scrivere un file di soluzione **col nome specificato**;
- avere dentro la funzione **col nome specificato**;
- porre il file di test corrispondente **nella stessa cartella**;
- eseguire in quella cartella il comando `python3 <fileditest>`

dove `<fileditest>` va ovviamente sostituito con il nome del file di test appropriato per l'esercizio su cui state lavorando. Per ogni esercizio ci sta un file di test indipendente, così da poter lavorare sugli esercizi uno alla volta con più agio.

Il risultato di ogni test è una schermata (o più schermate) nella quale si mostra:

- se è stato trovato il file con il nome corretto,
- se il file contiene la funzione con il nome corretto,
- se chiamate alla funzione con diversi valori dei parametri terminano restituendo risultati corretti.

Per ogni funzione scritta vengono eseguite chiamate con diversi valori dei parametri. L'esito dei test viene riportato con il carattere

- `E` se la chiamata non può essere eseguita,
- `F` se la funzione non restituisce il risultato corretto,
- `.` se la funzione restituisce il risultato corretto.

1 Conteggio delle vocali

Scrivete una funzione `conteggiovocali(testo)` che restituisca il numero di vocali presenti nella stringa `testo`. Ad esempio

```
print(conteggiovocali("Contate le vocali."))
```

```
7
```

Il programma Python deve essere salvato nel file: `conteggiovocali.py`

File di test: `test_conteggiovocali.py`

2 Buona Password

Scrivete una funzione `buona_password(s)` che prende una stringa `s` in input e verifica (restituisce True) se la stringa in input contiene almeno un numero, una maiuscola, una minuscola e un simbolo speciale (ovvero un carattere che non sia alfanumerico). Ad esempio

```
print(buona_password('c1aO#'))
print(buona_password('ciaO#'))
print(buona_password('c1ao#'))
print(buona_password('c1aO'))
```

```
True
False
False
False
```

Per questo esercizio potreste usare i metodi di stringa

- `str.isalpha()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli caratteri alfabetici;
- `str.isdigit()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli cifre numeriche.
- `str.islower()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli caratteri minuscoli;
- `str.isupper()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli caratteri maiuscoli;

Ad esempio:

```
print('à'.isalpha())
print('3'.isdigit())
print('a'.islower())
print('A'.isupper())
```

```
True
True
True
True
```

Il programma Python deve essere salvato nel file: `buona_password.py`

File di test: `test_buona_password.py`

3 Conteggio delle parole

Scrivete una funzione `conteggioparole(testo)` che restituisca il numero di parole presenti nel stringa `testo`. Ad esempio

```
print(conteggioparole("Python4Dummy? È, tra i manuali python,il peggiore."))
```

```
8
```

Notate che `Python4Dummy` è una parola sola, e che il pezzo di testo `python,il` contiene due parole. In questo esercizio consideriamo come parola una qualunque sequenza alfanumerica (ovvero composta da caratteri alfabetici e numerici). Spazi o segni di interpunzione devono essere considerati come separatori tra parole.

Per questo esercizio potreste usare i metodi di stringa

- `str.isalpha()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli caratteri alfabetici;
- `str.isdigit()` che restituisce `True` quando una stringa è **non vuota** ed è fatta di soli cifre numeriche.

In particolare potete usare questi metodi per verificare se un carattere è una cifra o un carattere alfabetico.

Ad esempio:

```
print('B'.isalpha())
print("3".isdigit())
print("a,b".isalpha())
print("Massimo".isalpha())
print("342".isdigit())
print("Massimo!".isalpha())
print("3 42".isdigit())
```

```
True
True
False
True
True
False
False
```

Il programma Python deve essere salvato nel file: `conteggioparole.py`

File di test: `test_conteggioparole.py`

4 Artista tartaruga

In questo esercizio, che è privo di file di test ed è puramente esplorativo, ci mettiamo a disegnare utilizzando *turtle graphics*. Facciamo finta di essere una tartaruga che cammina su un foglio bianco, con un pennarello agganciato alla coda. Possiamo procedere in avanti per una certa lunghezza, lasciando un segno, oppure girarci a destra o a sinistra di un certo angolo.

Per esempio per disegnare un quadrato di lato 10 (l'unita di misura è puramente convenzionale) possiamo:

1. andare avanti di 10
2. girare a destra di 90 gradi,
3. andare avanti di 10
4. girare a destra di 90 gradi,
5. andare avanti di 10
6. girare a destra di 90 gradi,
7. andare avanti di 10

Per attivare l'ambiente turtle graphics in un programma python si può usare l'istruzione

```
from turtle import *
```

Ed a quel punto il programma avrà a disposizione, tra le altre, le seguenti funzioni che controllano la tartaruga

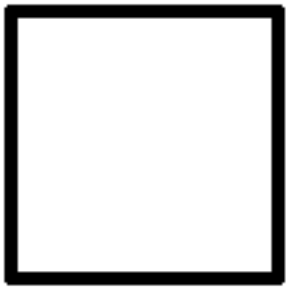
- `forward(N)` procedi in avanti per lunghezza `N`
- `left(x)` voltati a sinistra di `x` gradi
- `right(x)` voltati a destra di `x` gradi

In queste istruzioni `x` è un intero. Ci sono molte altre funzioni disponibili. Tra le altre

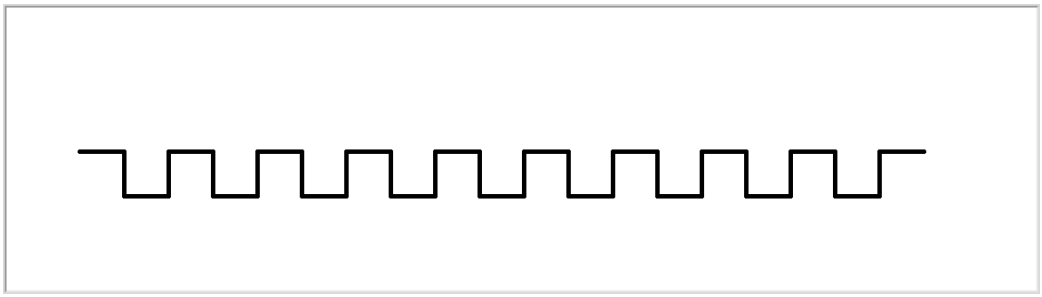
- `setup(larghezza,altezza)` ridimensiona la finestra
- `pensize(d)` cambia lo spessore del tratto
- `color(name)` cambia il colore del tratto

dove `name` può essere ad esempio `"green"`, `"red"`, `"yellow"`, `"blue"`, `"black"`, ...

Per ulteriori informazioni sul modulo `turtle` potete leggere il capitolo 4 del libro di testo. Adesso provate a disegnare immagini simili alle seguenti:



Provate a scrivere una funzione che dato un numero `N`, produca un'onda quadrata con `N` alti e `N-1` bassi. Per esempio con `N=10` il disegno è



Date sfogo alla creatività: usate i cicli per costruire figure interessanti e complesse.

```
1
```

5 Terne pitagoriche

Scrivete una funzione `ternepitagoriche1(N)` che dato il parametro N restituisca il numero di terne pitagoriche, ovvero le terne di numeri naturali (a, b, c) tali che

- $a^2 + b^2 = c^2$

con $1 \leq a < b < c \leq N$.

Per esempio le triple pitagoriche con numeri minori di 10 sono due, ovvero $(3, 4, 5)$ e $(6, 8, 10)$.

Il programma Python deve essere salvato nel file: `ternepitagoriche1.py`

File di test: `test_ternepitagoriche1.py`