

**Fondamenti di Informatica**  
**Corso di Laurea in Ingegneria Gestionale**  
**A.A. 2017-18:**  
**Esercitazione #12**

**Federico Scafoglieri**  
scafoglieri@diag.uniroma1.it

**Valsamis (Makis) Ntouskos**  
ntouskos@diag.uniroma1.it

**Domenico Lembo**  
lembo@diag.uniroma1.it

**Luca Becchetti**  
becchetti@diag.uniroma1.it

## Istruzioni

Collegarsi al sito del corso <https://piazza.com/uniroma1.it/spring2018/1017401/home> e scaricare i file necessari all'esercitazione dalla sezione risorse. I file sono compressi in un file .zip , contenente una cartella con i file dell'esercitazione.

## Svolgimento degli esercizi - Leggere attentamente

Leggere attentamente il testo e risolvere gli esercizi proposti. Per ogni esercizio avete unacartella EsercN che contiene un file dal nome *ExN.py* (dove N è il numero dell'esercizio) con lo scheletro della soluzione. Il file *SolN.py* contiene la soluzione dell'esercizio proposta dal docente. E' nel vostro diretto interesse non aprire quest'ultimo file prima di aver dedicato tutto il tempo disponibile a risolvere autonomamente l'esercizio. Non create nuovi file. Per verificare la correttezza di un esercizio DOVETE usare il programma *TestEx.py* (che non va assolutamente modificato), che proverà la vostra soluzione su un certo numero di casi di test, mostrandovi il confronto tra i risultati ottenuti con la vostra soluzione e usando la soluzione proposta dal docente. Ciò ha lo scopo di facilitarvi nella ricerca e correzione di eventuale errori.

## Svolgimento degli esercizi - Classe Persona

La classe Persona presente nel file Persona.py possiede i seguenti metodi:

- `getNome()` restituisce il nome della persona. Tipo restituito **stringa**
- `getCognome()` restituisce il cognome della persona. Tipo restituito **stringa**
- `getDataDiNascita()` restituisce la data di nascita della persona. Tipo restituito una **stringa** nel formato dd/mm/aaaa
- `getGiornoDiNascita()` restituisce il giorno di nascita della persona. Tipo restituito **intero**
- `getMeseDiNascita()` restituisce il mese di nascita. Tipo restituito **intero**
- `getAnnoDiNascita()` restituisce l'anno di nascita. Tipo restituito **intero**
- `getLuogoDiNascita()` restituisce il luogo di nascita. Tipo restituito **stringa**
- `getResidenza()` restituisce la residenza. Tipo restituito **stringa**

## Esercitazione Laboratorio

### Esercizio 1

#### Esercizio 1.0

Valore esercizio: **8 Punti**

Completare la funzione *Ex1(persone)* che prende in ingresso una lista di persone *person*e e restituisce il numero di persone maggiorenni.

#### Esercizio 1.1

Valore esercizio: **8 Punti**

Completare la funzione *Ex1(person)*e che prende in ingresso una lista di persone *person*e e restituisce una

nuova lista *ret* composta dalle persone maggiorenni e romane. Ogni elemento di *ret* è una stringa del tipo "*nome cognome*".

### Esercizio 1.2

Valore esercizio: **8 Punti**

Completare la funzione *Ex1(persone, citta)* che prende in ingresso una lista di persone *persone* e una stringa *citta*. La funzione restituisce una lista *ret* composta da tutte le coppie di persone che vivono in *citta* senza ripetizioni dovute a commutazione. Ogni elemento di *ret* è una coppia del tipo [*"nome cognome"*, *"nome cognome"*].

## Esercizio 2

### Esercizio 2.0

Valore esercizio: **8 Punti**

Completare la funzione *Ex2(persone)* che prende in ingresso una lista di persone *persone* e restituisce un dizionario composto da entry del tipo chiave:cognome e value:luogo di residenza di tutte le persone che hanno il luogo di nascita differente da quello di residenza.

### Esercizio 2.1

Valore esercizio: **8 Punti**

Completare la funzione *Ex2(persone)* che prende in ingresso una lista di persone *persone* e restituisce un dizionario composto da entry del tipo chiave:*c* e value:*l* dove *c*=luogo di nascita e *l*= lista di persone nate in *c*.

## Esercizio 3

### Esercizio 3.0

Valore esercizio: **8 Punti**

Completare la funzione *Ex3(albero)* che prende in ingresso un albero binario e restituisce il massimo dei valori associati ai nodi.

`getLeftChild()`: Restituisce il riferimento al figlio sinistro del albero  
`getRightChild()`: Restituisce il riferimento al figlio destro del albero  
`getRootVal()`: Restituisce il valore associato alla radice del albero

Tabella 1: Metodi della classe **BinaryTree** utili per questo esercizio

## Esercizio 4

### Esercizio 4.0

Valore esercizio: **8 Punti**

Completare la funzione *Ex4(albero)* che prende in ingresso un albero binario e restituisce una lista che contiene, per ogni foglia dell'albero, la somma dei valori associati ai nodi del percorso che dalla radice porta alla foglia.

`getLeftChild()`: Restituisce il riferimento al figlio sinistro del albero  
`getRightChild()`: Restituisce il riferimento al figlio destro del albero  
`getRootVal()`: Restituisce il valore associato alla radice del albero

Tabella 2: Metodi della classe **BinaryTree** utili per questo esercizio