Programming Principles

Prova scritta

Secondo appello: 31 gennaio

Lo scopo della prova è **creare una classe**, che gestisce una collezione di film costituita dal file *movies.json*.
Il file condiviso è già popolato con una decina di film, da usare come riferimento.
Ogni film è caratterizzato dai campi *title* e *director* che sono stringhe, *year* che è un numero intero e *genre* che è una lista di stringhe.

La classe, dovrà chiamarsi *MovieLibrary* e dovrà essere definita all'interno del file *movie_library.py*.
La classe dovrà avere due attributi di istanza, inizializzati nel metodo costruttore.
Il primo va chiamato *json_file,* e deve essere inizializzato passando, in fase di creazione oggetto,
il percorso assoluto del file *movies.json* sul vostro pc.
Il secondo va chiamato *movies* e rappresenta la collezione di film.
Deve essere inizializzato col contenuto del file json de-serializzato.
In pratica sarà una lista di dizionari, dove ogni dizionario rappresenta un film.
Ogni modifica che viene effettuata a *movies*, in qualsiasi metodo, deve immediatamente riflettersi sul file *movies.json*.

Il cuore della prova sono i **metodi** della classe.
La prova è infatti suddivisa in 18 esercizi, ciascuno riguardante la creazione di un metodo o la modifica di un metodo esistente.

Ogni esercizio che completerete correttamente vi conferirà un massimo di 3 punti.
Il punteggio massimo della prova è 50 punti.
Ma ho voluto darvi un aiutino: se provate a fare 18x3 ottenete un po' più di 50;
in questo modo, anche errando qualche esercizio,potrete comunque raggiungere il punteggio massimo.

Prima di elencarvi gli esercizi, alcuni suggerimenti e raccomandazioni:

- Assicuratevi di assegnare il **nome corretto** a ciascun **metodo** e ai relativi **parametri**;
- Assicuratevi che ogni metodo **restituisca** il **valore** corretto, col **datatype** corretto;
- Una volta creata la classe potrete testarla istanziandone un oggetto e chiamandone i metodi,
  sullo stesso file movie_library.py o su un file a parte.
  Tuttavia, i vostri test non incideranno sul punteggio,
  quel che conta è solo la classe e la corretta definizione e funzionamento dei metodi;
- Nessun metodo richiede stampe mediante funzione print,
  ma potete liberamente usarle per testare la classe;
- Per il voto di ciascun esercizio si terrà conto sia del corretto funzionamento del metodo
  che dell'applicazione delle linee guida di **formattazione** e **documentazione**.
  Assicuratevi perciò che il codice
  sia sufficientemente documentato e formattato (non serve la perfezione :)).
  Opzionalmente, aiutatevi con flake8, autopep8 e ruff,
  ma se quest'ultimo rileva errori complessi non perdeteci tempo.
- Per risolvere gli esercizi vi saranno sufficienti i comandi riportati nelle **slides**.
  E' comunque consentito fare ricerche online
  di comandi e informazioni che vi agevolino nella risoluzione.
  Tuttavia, vi sconsiglio caldamente di copiare porzioni di codice
  o generarle mediante Intelligenza Artificiale,
  poiché verrebbero individuate dai nostri strumenti di rilevamento AI.
- Può essere utile creare un metodo (magari privato) che si occupi
  di aggiornare il file *movies.json* ogni volta che l'attributo *movies* subisce modifiche.
  Tale metodo, che potete nominare magari *__update_json_file*,
  può essere chiamato in tutti i metodi che effettuano modifiche alla libreria di film.

ESERCIZI:

1. Crea un metodo chiamato *get_movies* che restituisce l'intera collezione di film.

2. Crea un metodo chiamato *add_movie* che ha i parametri *title* e *director* di tipo stringa, *year* di tipo intero e *genres* di tipo lista (di stringhe).
   Il metodo aggiunge il film alla collezione e aggiorna il file json.

3. Crea un metodo chiamato *remove_movie* che ha il parametro *title.*
   Il metodo rimuove dalla collezione il film
   che ha titolo corrispondente (NON case sensitive) a *title.*
   Il metodo aggiorna il file json e restituisce il film rimosso.

4. Crea un metodo chiamato *update_movie* che ha il parametro *title*
   e i parametri opzionali *director, year* e *genres*.
   Il metodo ricerca nella collezione il film
   che ha titolo corrispondente (NON case sensitive) a *title.*
   Quindi modifica il film,
   applicando il valore di ciascun parametro opzionale non nullo.
   Il metodo aggiorna il file json e restituisce il film coi valori aggiornati.

5. Crea un metodo chiamato *get_movie_titles*
   che restituisce una lista contenente tutti i titoli dei film nella collezione.

6. Crea un metodo chiamato *count_movies*
   che restituisce il numero totale dei film nella collezione.

7. Crea un metodo chiamato *get_movie_by_title* che ha il parametro *title.*
   Il metodo restituisce il film
   che ha titolo corrispondente (NON case sensitive) a *title.*

8. Crea un metodo chiamato *get_movies_by_title_substring*
   che ha il parametro *substring*.
   Il metodo restituisce una lista di tutti i film che contengono, nel titolo,
   una sottostringa corrispondente (case sensitive) a *substring*.

9. Crea un metodo chiamato *get_movies_by_year* che ha il parametro *year*.
   Il metodo restituisce una lista di tutti i film con anno corrispondente a *year*.

10. Crea un metodo chiamato *count_movies_by_director* che ha il parametro *director*.
    Il metodo restituisce un numero intero che rappresenta,
    quanti film del *director* scelto sono presenti nella collezione.
    Il *director* va confrontato in modo NON case sensitive.

11. Crea un metodo chiamato *get_movies_by_genre* che ha il parametro stringa *genre*.
    Il metodo restituisce una lista di tutti i film che hanno genere corrispondente a *genre*.
    Il *genre* va confrontato in modo NON case sensitive.

12. Crea un metodo chiamato *get_oldest_movie_title* che restituisce
    il titolo del film più antico della collezione.

13. Crea un metodo chiamato *get_average_release_year* che restituisce
    un float rappresentante la media aritmetica degli anni di pubblicazione
    dei film della collezione.

14. Crea un metodo chiamato *get_longest_title* che restituisce il titolo
    più lungo della collezione di film.

15. Crea un metodo chiamato *get_titles_between_years* che ha due parametri:
    start_year e *end_year*.
    Il metodo restituisce una lista contenente i **titoli** dei film pubblicati
    dall'anno *start_year* fino all'anno *end_year* (estremi compresi).

16. Crea un metodo chiamato *get_most_common_year*
    che restituisce l'anno che si ripete più spesso fra i film della collezione.
    Non considerare il caso in cui vi siano pari merito.

17. Modifica il metodo costruttore affinché,
    se nel percorso *json_file* non viene trovato alcun file,
    venga sollevata l'eccezione *FileNotFoundError*
    col messaggio personalizzato "*File not found:* " seguito dal percorso *json_file*.

18. Modifica i metodi *remove_movie* e *update_movie* affinché,
    se non viene trovato alcun film avente come titolo *title*,
    venga sollevata l'eccezione personalizzata *MovieNotFoundError*,
    avente il messaggio "*Movie was not found*".
    Tale eccezione va definita all'interno della classe *MovieLibrary.*

Buon lavoro e buona fortuna,

Prof. Edoardo Marceddu