`BIB`: una collezione di funzioni per la gestione del catalogo di una biblioteca

Barbara guidi, Susanna Pelagatti

Primo assegnamento in itinere 622AA
AA 2022-23

Indice

1 Introduzione

Il primo modulo (9 crediti) del corso di Programmazione e Analisi Dati (622AA) prevede lo svolgimento di due assegnamenti in itinere che esonerano dallo svolgimento del progetto finale. Questo documento descrive il primo assegnamento che prevede la realizzazione di un insieme di funzioni Python pe rla gestione di un semplice catalogo di una bibloteca.

Il software viene sviluppato e documentato utilizzando gli strumenti, le tecniche e le convenzioni presentati durante il corso.

1.1 Materiale in linea

Tutto il materiale relativo al corso può essere reperito sul Moodle ufficiale del corso. Eventuali chiarimenti possono essere richiesti alle docenti per posta elettronica.

1.2 Struttura degli assegnamenti, bonus, tempi di consegna e prova orale

I due assegnamenti possono essere svolti individualmente o in gruppi di 2 studenti e possono essere consegnati entro la data e l'ora dell'appello di Febbraio 2023. Agli studenti che consegnano una realizzazione sufficiente entro 8 giorni

dalla pubblicazione su Moodle vengono assegnati 2 punti di Bonus nella valutazione complessiva di ciascun assegnamento. La valutazione complessiva viene data dalla media delle valutazioni dei due assegnamenti e costituisce la base per la prova orale. La prova orale sará composta di due parti. La prima parte sarà una discussione sugli assegnamenti presentati e tenderà a stabilire se lo studente è realmente l'autore di quanto consegnato (in caso di dubbi la valutazione verrà opportunamente aggiustata) in particolare verrá chiesto di leggere e modificare il codice e di spiegare quanto usato (in particolare costrutti e nmoduli non facenti parte del programma del corso). La seconda parte verterà su tutto il programma del corso. In particolare, l'orale comprenderà:

- una discussione delle scelte implementative
- l'impostazione e la scrittura di semplici programmi Python (sequenziali e concorrenti) di difficoltà medio bassa rispetto a quelli visti nelle esercitazioni in classe
- domande su tutto il programma presentato durante il corso.

Il voto finale sará la media (0-30L) fra la valutazione effettiva del progetto (emendata in caso di dubbi) e la prova orale.

1.3 Consegna degli assegnamenti

La consegna degli assegnamenti avviene *esclusivamente* per posta elettronica secondo le istruzioni presenti nel `README` di ciascun assegnamento.

1.4 Valutazione dell'assegnamento

All'assegnamento viene assegnata una fascia di valutazione da 0 a 30 che tiene conto dei seguenti fattori:

- motivazioni, originalità ed economicità delle scelte implementative
- strutturazione del codice (fattorizzazione del codice in funzioni, uso di strutture dati adeguate etc)
- efficienza e robustezza (numero di operazioni eseguite, fallimenti in caso di input inadeguati etc)
- aderenza alle specifiche
- qualità del codice Python e dei commenti

Tutti gli assegnamenti verranno confrontati automaticamente per verificare situazioni di plagio. Nel caso di elaborati uguali verranno presi provvedimenti per tutti i gruppi coinvolti.

2 L'assegnamento `BIB`

L'assegnamento prevede la realizzazione di un insieme di funzioni per la gestione del catalogo di una biblioteca. La biblioteca contiene solamente libri cartecei e ogni record che descrive un libro é rappresentato da una tupla

```
(cognomeautore, nomeautore, titolo, collocazione, anno, note)
```

I campi `cognomeautore`, `nomeautore`, `titolo`, `note` sono di tipo stringa. Il campo `anno` é un intero positivo. il campo collocazione é una tupla che contiene una stringa e un numero positivo in quest'ordine. Ad esempio:

```
("Adams", "Douglas", "Guida galattica per gli autostoppisti" ,
("F",22), 1980, "Traduzione di Laura Serra")
```

Il catalogo della biblioteca é rappresentato come una lista di tuple.
Si devono realizzare (almeno) le seguenti funzioni (specificati nel file `biblio.py` che riportiamo qua per comoditá)

```
def inserisci(cat,cognome,nome,titolo,anno,collocazione,note=[]):
    """ Inserisce un nuovo record (libro) nel catalogo controllando che i tipi dei
    parametri attuali siano corretti -- non modifica maiuscole e minuscole dei parametri
    :param cat: il catalogo da modificare
    :param cognome: cognome dell'autore (stringa)
    :param nome: nome dell'autore (stringa)
    :param titolo: titolo del libro (stringa)
    :param anno: anno di pubblicazione (intero positivo=
    :param collocazione: tupla (stringa,intero positivo)
    :param note: stringa (opzionale)
    :return: True se l'inserimento e' stato effettuato con successo, False
    altrimenti
    :return: None se i parametri non hanno il tipo corretto
    """
    pass #instruzione che non fa niente --> da sostituire con il codice


def serializza (cat):
    """ Serializza un catalogo rappresentando la sequenza dei record in una singola stringa
    La sottostringa relativa al singolo record
    puo' usare un formato a scelta dello studente,
    i record devono essere separati dal carattere "a capo" --> "\n"

    :param cat: il catalogo da serializzare
    :return: una stringa che rappresenta il catalogo
    """
    pass #istruzione che non fa niente --> da sostituire con il codice


def crea_copia(cat):
    """ Crea una copia completa del catalogo cat (un clone) e lo restituisce
    :param cat: il catalogo da clonare
    :return: il nuovo catalogo clonato
    """
    pass #istruzione che non fa niente --> da sostituire con il codice


def sono_uguali(cat1,cat2):
    """Funzione booleana che stabilisce se due cataloghi contengono
    esattamente gli stessi record con gli stessi dati (eccetto collocazione e nota -- che possono
    essere divers)
    I record possono non essere nello stesso ordine nei due cataloghi
    :param cat1: primo catalogo da confrontare
    :param cat2: secondo catalogo da confrontare
    :return: True se sono uguali, False altrimenti
    """
    pass  # istruzione che non fa niente --> da sostituire con il codice
```

```
def concatena(cat1,cat2):
    """crea un nuovo catalogo concatenando cat1 e cat2 e lo restituisce come risultato --
    se ci sono k record uguali in tutti i campi eccetto il campo "note"
    il risultato contiene un solo record che nel campo note contiene la
    concatenazione delle note dei k record uguali in ordine lessicografico
    :param cat1: primo catalogo da unire (non viene modificato)
    :param cat2: secondo catalogo da unire (non viene modificato)
    :return: il nuovo catalogo
    """

    pass  # instruzione che non fa niente --> da sostituire con il codice


def cancella(cat,  titolo, anno=None):
    """Cancella tutti i record per i quali i campi titolo (e opzionalmente anno)
    coincidono con i parametri (titolo,anno)
    :param cat: il catalogo da modificare
    :param titolo: il titolo del libro (obbligatorio) che deve coincidere a meno di case (maiuscole/minuscole)
    :param anno: l' anno di pubblicazione (opzionale)
    :return: il numero dei record cancellati
    :return: None se i parametri non hanno il tipo corretto
    """
    pass  # istruzione che non fa niente --> da sostituire con il codice

def cerca(cat, pctitolo):
    """Verifica che esista almeno un titolo che contiene la stringa pctitolo come sottoscringa (attenzione agli
     e a maiuscole e minuscole)
    :param cat: il catalogo (non viene modificato)
    :param pctitolo: la sottostringa che deve comparire nel titolo del libro
    :return: True se c'Ã¨ almeno un record che contiene pctitolo, False altrimenti
    :return: None se i parametri non hanno il tipo corretto
    """
    pass  # instruzione che non fa niente --> da sostituire con il codice

def ordina(cat):
    """ Ordina il catalogo alfabeticamente per cognome e nome e
    (in caso di piu' opere dello stesso autore) per anno di pubblicazione e
    infine per Titolo all'interno dello stesso anno come in:

        Dexter Colin L'ultima corsa per Woodstock 2010 ....
        Dexter Colin Il mondo silenzioso di Nicholas Quinn  2012 ...
        Dexter Colin Niente vacanze per l'ispettore Morse 2012 ....
        Simenon Georges Gli intrusi 2015
    :param cat: il catalogo da ordinare
    :return: None (la funzione modifica il catalogo e non restituisce niente)
    """
    pass  # instruzione che non fa niente --> da sostituire con il codice
```

É possibile definire funzioni ausiliarie per la soluzione ed estendere le funzionalitá del catalogo definendo nuove funzioni.

Per ogni funzione, verificare che i tipi dei parametri passati durante la chiamata siano quelli attesi e segnalare le situazioni erronee con opportuni messaggi di errore stampati a schermo.

2.1 Istruzioni

Unitamente a questo file vengono forniti due file Python contenenti l'intestazione delle funzioni da ralizzare (`biblio.py`) e i test che sevono essere superati per consegnare il codice (`main.py`) e un file di istruzioni (`README`).

Consigliamo di leggere attentamente le istruzioni e analizzare il codice fornito per capire come strutturare le funzioni prima di iniziare a progettare e implementare. Ricordiamo

l'importanza di analizzare i vari casi prima di iniziare a scrivere codice e di effettuare test incrementali sul codice durante lo sviluppo.

Possono essere realizzate funzionalità in più rispetto a quelle richieste (ad esempioaltre funzioni).

Le parti opzionali devono essere corredate da test appropriati e documentate da commenti chiari o (in caso sia necessario) da un breve documento descrittivo che puo' essere consegnato insieme al codice e che spiega le motivazioni e la struttura di quanto realizzato.