collaborazioni.py

Lo scopo del programma è: presi in input il file `partecipazioni.txt` prodotto da `CreaGrafo.java` e il file `title.basics.tsv` ed una sequenza di codici di attori, stampa su standard error l'elenco dei fil in cui gli attori, presi a coppie, hanno lavorato insieme.

Si pate dalla fine, con il comando

```
if __name__ == '__main__':
    main()
```

che mi serve per distingue se sto lanciando lo script direttamente oppure se lo sto importando in un altro script:

- se lo sto lanciando direttamente viene eseguita la funzione `main()`
- se lo sto importando in un altro script allora la funzione `main()` non viene eseguita, ma metto a disposizione le mie funzioni

Successivamente si passa all'esecuzione della funzione `main()`.
La prima cosa che viene fatta è controllare che il numero di argomenti passati sia quello corretto:

```
# Controllo sul numero di argomenti passati da linea di comando
    if(len(sys.argv) < 5):
        print(f"Uso:\n\t {sys.argv[0]} partecipazioni.txt
titile.basics.tsv codice1 codice2 [codice3 codice4 ...]")
        sys.exit(1)
```

viene verificato che non sia strettamente minore di 5, dato che viene richiesta almeno una coppia di codici di attori.

Successivamente come richiesto dalle specifiche, viene stampato su `stdout` il comando eseguito

```
# Stampa del comando passato da linea di comando
print("$", *sys.argv)
```

A questo si assegna alle variabili (stringhe) `filePartecipazioni` e `fileTitle` i due file passati da line di comando:

```
fileParteciapazioni = sys.argv[1]
fileTitle = sys.argv[2]
```

Successivamente viene creata una lista in cui vengono inseriti i codici degli attori passati da linea di comando.
Per prendere tali codici viene effettuato un ciclo `for` a partire dal terzo elemento passato da linea di comando:

```
# Ciclo for che scorre tutti gli elmenti, dal terzo in poi, passati da
linea di comando
    for cod in sys.argv[3:]:

        # Cast da stringa ad intero
        codice = int(cod)
        codiciAttori.append(codice)
```

ovviamente i codici quando vengono passati da linea di comando sono delle stringhe, quindi viene fatto un cast esplicito e successivamente un `append()` per inserire il codice nella lista.

A questo punto avviene la lettura del file `partecipazioni.txt` grazie alla chiamata della funzione `letturaPartecipazioni(..)`:

```
#print("=== INIZIO LETTURA FILE partecipazioni.txt ===\n",sys.stderr);
partecipazioni = letturaPartecipazioni(fileParteciapazioni)
#print("=== FINE LETTURA FILE partecipazioni.txt ===\n",sys.stderr);
```

che restituisce un Dizionario (un insieme di coppie <chiave,valore>) in cui le chiavi sono i codici identificativi degli attori (letti in `partecipazioni.txt`) mentre il valore associato ad ogni codice attore è un Set che racchiude tutti i codici dei film associati alla chiave all'interno del file `partecipazioni.txt`.

Successivamente avviene la lettura del file `title.basics.tsv` grazie alla chiamata della funzione `letturaTitle(..)`

```
#print("=== INIZIO LETTURA FILE title.basics.tsv ===\n",sys.stderr);
titles = letturaTitle(fileTitle)
#print("=== FINE LETTURA FILE title.basics.tsv ===\n",sys.stderr);
```

che restituisce un Dizionario in cui le chiavi sono i codici identificativi dei film letti dal file `title.basics.tsv`, mentre i valore associato alla chiave è il titolo del film (sotto forma di stringa) anch'esso letto dal file `title.basics.tsv`.

Ora si ha un ciclo for che scorre tutti i codici presenti nella lista `codiciAttori`, per ogni codice attore:

- salva in `actor1` e `actor2` il codice i-esimo e quello i+1-esimo
- chiama la funzione `cercaFilmCollaborazione(..)` che restituisce una lista di tuple dove il primo elemento della tupla è il codice del film e il secondo elemento è il titolo esteso
- ottiene tramite la funzione `len(..)` la lunghezza della lista di tuple
- controlla se la lunghezza è uguale a zero:
  - se lo è significa che i due attori non avuto alcuna collaborazione e dunque proce alla stampa di tale conclusione
  - se invece non lo è per prima cosa stampa "codice1.codice2: <num_collaboraioni> collaborazioni: " e con un ciclo for, cicla le singole tuple della lista in modo tale da accedere ai due valori di ogni tutpla e stampare come richiesto dalle specifiche
- tra ogni riga inserisce una linea vuota

```
for i in range(len(codiciAttori) - 1):

    actor1 = codiciAttori[i]
    actor2 = codiciAttori[i + 1]

    filmInCollaborazione = cercaFilmCollaborazione(actor1, actor2,
partecipazioni, titles)

    numCollaborazioni = len(filmInCollaborazione)

    if numCollaborazioni == 0:
        print(f"{actor1}.{actor2} nessuna collaborazione")
    else:
        print(f"{actor1}.{actor2}: {numCollaborazioni} colaborazioni:")
        for filmCode, filmTitle in filmInCollaborazione:
            print(f" {filmCode:7d} {filmTitle}")

    # Inserisco una linea vuota tra ogni coppia di attori
    print()
```

Infine procedo con le stampe finali come richiesto dalle specifiche:

```
print("=== Fine")
```

Descrizioni delle tre funzioni chiamate dal main:

- `letturaPartecipazioni(..)`
- `letturaTitle(..)`
- `cercaFilmCollaborazioni(..)`

letturaPartecipazioni

Prende come argomento la stringa che contiene il nome del file e restituisce un Dizionario in cui le chiavi sono i codici identificativi degli attori mentre il valore associato ad ogni codice attore è un Set che racchiude tutti i codici dei film associati alla chiave all'interno del file `partecipazioni.txt`

La funzione inizia creando il Dizionario `partecipazioni` vuoto:

```
partecipazioni = {}
```

Successivamente si ha un costrutto `with ... as` per la lettura del file: in cui il file passato per argomento viene aperto in sola lettura

```
with open(filePartecipazioni,'r') as file:
    # CORPO DEL WITH ... AS #
```

Al suo interno il corpo inizia con un ciclo for per la lettura di ogni singola riga del file:

```
for line in file:
```

al suo interno viene creata una lista in cui vengono salvati i campi della linea esaminata, questo viene fatto grazie all'utilizzo della funzione `strip()` e della funzione `spllit(..)`:

- `strip()` elimina spazi bianchi alla fine ed all'inizio della riga ed inoltre elimina anche `\n` e `\t` che si trovano all'inizio o alla fine della line
- `slit(<cod>)` separa i singoli campi, infatti non appena trova il separatore `\t` sa che il token letto è finito

```
# Crea una lista di stringhe "campi = ["str1","str2",..]", partendo
# dalla linea del file il metodo strip() elimina spazi pbianchi all'inizio
# e alla fine della riga, elimina anche \n e \t se si trovano all'inizio o
# alla fine, mentre split separa le parole della linea se sono divise da
'\t'
campi = line.strip().split('\t');
```

Successivamente preleva i campi che ci interessano, e crea un Set in cui accogliere tutti i codici dei film:

```
codAtt = int(campi[0])
numFilm = int(campi[1])
```

```
# Creo un Set per i codici dei film
setFilm = set()
```

A questo punto scorre ogni singolo campo dal secondo in poi (perché i primi due sono il codice dell'attore ed il numero di film in cui ha lavorato) tramite un ciclo for ed aggiunge ogni codice di film al Set precedentemente creato

```
# Scorro gli elementi della lista campi a partire dal campo 2 fino a che
non
# arrivo in fondo alla lista ovvero numero di film + 2
for i in range(2, 2 + numFilm):
    setFilm.add(int(campi[i]));
```

Terminata tale operazione viene inserita nel dizionario la coppia chiave valore appena estratta dal file e si passa alla linea successiva:

```
# Inserisco nel Dizionario la coppia <chiave,valore>: codAtt, setFilm
partecipazioni[codAtt] = setFilm
```

Terminato di leggere il file si restituisce il Dizionario al chiamante

```
# Restituisco il Dizionario creato
return partecipazioni
```

---

letturaTitle

Prende come argomento il nome del secondo file passato da linea di comando e ritorna un Dizionario in cui le chiavi sono i codici identificativi dei film letti dal file `title.basics.tsv`, mentre i valore associato alla chiave è il titolo esteso del film che ha tale codice

---

Inizia con la creazione del Dizionario vuoto:

```
titoli = {}
```

Successivamente si incontra il costrutto `with ... as` per l'apertura del file passato per argomento: infatti viene aperto il file tramite la funzione `open(..)` in sola lettura:

```
with open(fileTitle, 'r') as file:
```

Al suo interno salto la prima riga del file, che è quella di intestazione:

```
# Salto la riga di intestazione
file.readline()
```

e tramite un ciclo for procedo alla lettura riga per riga del file:

```
for line in file:
```

All'interno del corpo del for vengono estratti campi del file e slavati in una lista, questo viene fatto grazie all'utilizzo della funzione `strip()` e della funzione `spllit(..)`:

- `strip()` elimina spazi bianchi alla fine ed all'inizio della riga ed inoltre elimina anche `\n` e `\t` che si trovano all'inizio o alla fine della line
- `slit(<cod>)` separa i singoli campi, infatti non appena trova il separatore `\t` sa che il token letto è finito

```
campi = line.strip().split('\t')
```

Successivamente viene preso singolarmente il primo campo, ovvero il codice univo del film, a cui vengono levati i primi due caratteri ('tt') così da ottenere il codice puro e viene preso singolarmente anche il titolo esteso:

```
# prendo il codice che è di queto tipo: tt00000001
codiceAppoggio = campi[0]

# trasformo in intero il codice escludento il 'tt'
codiceFilm = int(codiceAppoggio[2:])

titolo = campi[2]
```

In fine viene aggiunto al Dizionario la coppia <chiave,valore> con chiave il codice univoco estratto da file e valore il titolo completo.
Si passa poi dunque alla riga successiva.

Terminato di leggere tutte le righe, la funzione ritorna al chiamante il Dizionario.

---

cercaFilmCollaborazioni

Prende in input due codici di attori, il Dizionario che ha per chiave il codice dell'attore e per valore il Set di codici dei film cui ha partecipato (`partecipazioni`) e il Dizionario che ha per chiave il codice del film e per valore il titolo esteso (`titles).
Ritorna una lista di tuple, dove il primo elemento è il codice del film ed il secondo è il titolo esteso.

La funzione inizia con un controllo sui codici degli attori passati per argomento, infatti se, uno dei due o entrambi, non sono presenti all'interno del Dizionario `partecipazioni` ritorna una lista vuota:

```
# Controllo l'esistenza dei due attori in "partecipazioni"
if actor1 not in partecipazioni or actor2 not in partecipazioni:
    return []
```

Successivamente prosegue calcolando l'intersezione tra i Set associati ai due codici attori, in modo da trovare solamente i film i codici dei film in comune, e li salva all'interno del Set `filmInComune`:

```
# Faccio intersezione dei Set dei film dei due attori
filmInComune = partecipazioni[actor1] & partecipazioni[actor2]
```

Dichiara una ed inizializza una lista vuota e successivamente è presente u ciclo for per riempirla.
Tale for, cicla ogni elemento del Set `filmInComune`, per ogni codice di film tramite una `get(..)` sul Dizionario `titles`, ottiene il suo titolo esteso ed inserisce con un `append(..)` la tupla corrente:

```
filmInCollaborazione = []
for actorCode in filmInComune:
    titolo = titles.get(actorCode, "< Titolo assente >")
    filmInCollaborazione.append((actorCode,titolo))
```

Dopo aver inserito nella lista tutte le tuple, tale lista viene ordinata in modo crescente in ordine di codice del film, ed infine la funzione ritorna al chiamante la lista di tuple

```
# Ordino in maniera crescente i valori dei codici degli attori
filmInCollaborazione.sort()

return filmInCollaborazione
```