Appello del 28 gennaio

Istruzioni

Svolga gli esercizi nei file indicati e dopo aver concluso tutti gli esercizi comunichi alla docente di voler consegnare. Dopo aver ricevuto l'ok dalla docente mostri uno alla volta i file degli esercizi alla docente. Da quel momento non potra` piu` modificare il codice dell'esercizio e dovra` salvare tutti i file .py degli esercizi in una cartella zippata il cui nome deve essere formato dal suo cognome e nome separati da un '_' (ad esempio, rossi_mario). Invii la cartella zippata a adebonis@unisa.it con oggetto "Soluzione esercizi appello 28 gennaio".

Durante lo svolgimento della prova, eccezion fatta per il collegamento al meeting di Zoom, è vietato l'uso di internet (navigazione tramite browser, posta, ssh, ftp, accesso remoto al PC, condivisione dello schermo e di qualsiasi cartella, ecc.).
Teams e Zoom devono essere ridotti ad icona.

NON MODIFICHI IL CODICE GIA` SCRITTO NEI FILE FORNITI DALLA DOCENTE (0 punti se cio` viene fatto)

1. Scrivere nel file esercizio1.py una funzione che prende in input una sequenza di richieste (stringhe) e passa ciascuna richiesta ad una catena di gestori ciascuno dei quali e' una coroutine (usando il decoratore @coroutine visto a lezione).
   - Se la stringa comincia con una lettera compresa tra 'a' e 'g' allora la richiesta viene gestita dal gestore gestore_ag che stampa "Richiesta {} gestita da gestore_ag''.
   - Se la stringa comincia con una lettera compresa tra 'h' e 'n' allora la richiesta viene gestita dal gestore gestore_hn che stampa "Richiesta {} gestita da gestore_hn''.
   - Se la stringa NON comincia con una lettera allora la richiesta viene gestita dal gestore gestore_distr che stampa "Richiesta {} gestita da gestore_distr: uso improprio della catena di gestori'' **e poi smette di funzionare.**
   - Se la stringa comincia con una lettera dell'alfabeto successiva ad 'n' o con una maiuscola allora la richiesta viene gestita dal gestore gestoreDiDefault che stampa "Messaggio da gestoreDiDefault: non è stato possibile gestire la richiesta {} ".

   Nelle suddette stampe il nome della richiesta deve comparire al posto delle parentesi graffe.
   Se ad un certo punto un gestore manda la richiesta al suo successore e il successore smette di funzionare allora anche il gestore che inviato la richiesta deve smettere di funzionare. Prima di smettere di funzionare il gestore deve stampare "Il successore di {} ha smesso di funzionare a causa della richiesta {} e di conseguenza smette di funzionare anche {}'', dove al posto della I e III coppia di parentesi graffe deve comparire il nome del gestore e al posto della II coppia il nome della richiesta.

2. Scrivere nel file esercizio2.py un decorator factory myDecoratorFact che prende in input un intero n e restituisce un decoratore di funzione che modifica il comportamento della funzione decorata in modo che se la funzione è invocata con numero di argomenti diverso da n allora la funzione lancia TypeError in modo che il programma fornito dalla docente effettui le stampe desiderate (si veda la parte commentata alla fine del file).

Appello del 28 gennaio

3. Scrivere la classe FSet che estende **frozenset** in modo tale che quando si crea un'istanza di FSet , l'istanza di FSet creata contenga solo gli elementi di ordine dispari dell'oggetto iterabile (il primo, il terzo, ecc.) passata come argomento a FSet(). Se FSet() non prende input niente allora l'istanza creata è vuota.
   NB: gli elementi di ordine dispari corrispondono agli elementi di indice pari.

4. Scrivere la funzione **stampaParole** all'interno del file esercizio4.py. Se la funzione ha bisogno di invocare altre procedure, fornire anche quest'ultime. La funzione **stampaParole** prende in input una lista di stringhe **listaParole** , una lista di nomi di file **listaFile,** e il parametro **concorrenza**. Facendo uso di **multiprocessing.JoinableQueue**, la funzione **stampaParole** deve stampare per ciascuno dei file di listaFile, il nome del file, la parola di listaParole che appare piu` volte nel file e il numero di volte in cui essa appare nel file. Cio` deve essere fatto con un processo separato per ogni file di listaFile e **le stampe devono essere effettuate nell'ordine in cui terminano i processi. Il callable usato per effettuare la ricerca nel singolo file deve prendere in input la lista di parole e il nome del file.**
   Se non si è in grado di usare multiprocessing.JoinableQueue è possibile usare concurrent.futures. In questo caso saranno detratti dei punti.