Appello del 18 febbraio 2020

Svolgete gli esercizi nei file indicati e poi salvate i file in una cartella zippata il cui nome deve essere formato dal vostro cognome e dal vostro nome (**in minuscolo**) separati da un '_' (ad esempio, rossi_mario). **Il file zippato deve contenere solo i quattro file contenenti il codice richiesto dagli esercizi e deve essere salvato sul desktop.**

NON MODIFICATE IL CODICE GIA` SCRITTO NEI FILE FORNITI DALLA DOCENTE. SE MODIFICATE IL SUDDETTO CODICE, L'ESERCIZIO NON SARA` VALUTATO, SENZA ECCEZIONE ALCUNA.

1. Scrivere nel file esercizio1.py, all'interno della classe Client, il metodo di istanza **entrust** che prende in input due liste. La prima lista contiene delle stringhe mentre la seconda contiene dei caratteri singoli. La funzione entrust crea una catena di gestori e le invia coppie del tipo (s_i, c_i), dove s_i è l'i-esima stringa della prima lista e c_i è l'i-esimo carattere della seconda lista. Ciascun gestore della catena è una coroutine e per una data coppia (s,c) funziona come segue.
    - Se la stringa s non è vuota, comincia con una lettera dell'alfabeto e questa è uguale a c (secondo membro della coppia) allora la richiesta viene gestita dal gestore **handler_eq** che stampa "Richiesta {} gestita da handler_eq".
    - Se la stringa s non è vuota, comincia con una lettera dell'alfabeto e questa è diversa da c (secondo membro della coppia) allora la richiesta viene gestita dal gestore **handler_diff** che stampa "Richiesta {} gestita da handler_diff: richiesta modificata". Il gestore quindi cancella il primo carattere della stringa ricevuta e la invia ad una nuova catena di gestori uguale a quella creata da entrust.
    - Se la stringa s non è vuota e comincia con una numero compreso tra 0 e 9 allora la richiesta viene gestita dal gestore handler_digit che stampa "Richiesta {} gestita da **handler_digit**".
    - Se la stringa s è vuota o comincia con un carattere che non è ne' una lettera ne' un digit allora la richiesta viene gestita dal gestore **DefaultHandler** che stampa "Messaggio da DefaultHandler: non è stato possibile gestire la richiesta {} ".

   Nelle suddette stampe il nome della richiesta deve comparire al posto delle parentesi graffe.

2. Scrivere nel file esercizio2.py un decoratore di classe **decoratoreDiClasse** che dota la classe decorata di un metodo di istanza elencaVariabili. Il metodo elencaVariabili è un generatore che restituisce un iteratore delle variabili di istanza di tipo **int** dell'istanza per cui è invocato.

3. Scrivere un decorator factory **decoratorFactory** che permette di parametrizzare il decoratore dell'esercizio 2 con un tipo t in modo che l'iteratore restituito da elencaVariabili iteri solo sulle variabili di istanza di tipo t. Il codice relativo a questo esercizio deve essere scritto nel file esercizio3.py.

Appello del 18 febbraio 2020

4. Scrivere la funzione creaDizionari all'interno del file esercizio4.py. Se la funzione ha bisogno di invocare altre procedure, fornire anche queste ultime. La funzione creaDizionari prende in input una lista di interi e il parametro concorrenza quindi crea un numero di istanze di **dict** pari al numero di elementi della lista. L'i-esima istanza creata deve avere un numero di chiavi uguale all'i-esimo intero della lista. Le coppie (chiave,valore) dell'i-esimo dizionario sono (1,1+i),(2,2+i),…,(p_i ,p_i +i), dove p_i è l'i-esimo intero della lista. Facendo uso di **Futures** e **Multiprocessing**, creaDizionari deve creare ciascuna istanza di dict con un processo separato e stampare la somma dei valori presenti nel dizionario insieme all'indice i che indica la posizione nella lista dell'intero a partire dal quale è stato creato il dizionario **. Le stampe devono essere effettuate nell'ordine in cui terminano i processi.**