Prova di Programmazione Avanzata
22 dicembre 2020
a.a. 2020-21

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 dicembre".

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 coroutine **attacca(suffisso,myFile)** che riceve in input una parola suffisso ed il nome di un file myFile. Quando la coroutine riceve un oggetto (inviato da send) assume che l'oggetto sia una stringa e concatena a questa stringa la stringa suffisso e scrive la stringa cosi` ottenuta su una nuova linea, alla fine del file di nome myFile. La coroutine attacca smette di funzionare quando la stringa ricevuta termina gia` con suffisso (ad esempio se suffisso="ssimi", la coroutine smette di funzionare se riceve "carissimi").
   Scrivere inoltre la funzione **i_o(myFile,suff)** che prende in input una stringa myFile oltre al suffisso. La funzione crea una coroutine attacca e legge dal file myFile una parola per volta e invia ciascuna parola letta alla coroutine. La funzione i_o deve smettere di inviare parole non appena la coroutine smette di funzionare.

2. Scrivere nel file esercizio2.py una funzione applica(da_fare) che prende in input una tupla da_fare che contiene tuple della forma (m, a1,a2,…) dove m è un metodo di istanza di una certa classe C e a1, a2,… sono gli argomenti diversi da self con cui deve essere invocato m. Nella prima tupla di da_fare, m è la classe C e gli argomenti sono quelli da passare al costruttore per creare l'istanza di C su cui invocare i metodi delle restanti tuple di da_fare. La funzione applica restituisce una lista contenente come primo elemento l'istanza creata e come restanti elementi i valori restituiti dalle invocazioni dei metodi delle restanti tuple.

Prova di Programmazione Avanzata
22 dicembre 2020
a.a. 2020-21

3. Scrivere nel file esercizio3.py un decorator factory **decFact** che prende in input un tipo **t** e restituisce un decoratore che dota la classe decorata di un metodo statico riportaVariabiliDiClasse. Il metodo riportaVariabiliDiClasse non prende in input alcun argomento e restituisce un generatore di triple. Per ciascuna variabile di classe di tipo **t** della classe decorata deve essere generata una tripla contenente come primo elemento il nome della variabile, come secondo elemento il valore della suddetta variabile e come terzo elemento la classe in cui viene trovata la variabile (potrebbe essere la classe decorata o una delle sue superclassi).

4. Scrivere la funzione creaIFile all'interno del file esercizio4.py. Se la funzione ha bisogno di invocare altre procedure, fornire anche queste ultime. La funzione creaIFile prende in input due liste contenenti lo stesso numero di stringhe e il parametro concorrenza quindi si comporta come segue.
   Per ogni indice i delle liste, la funzione creaIFile crea un file con nome uguale alla stringa i-esima della prima lista e copia al suo interno le linee di ordine dispari (la prima, la terza, la quinta, ecc) del file con nome uguale alla stringa i-esima della seconda lista. Facendo uso di c**oncurrent**.**futures**, creaIFile **deve creare ciascun file, scrivere in esso e ottenere il numero di caratteri scritti con un processo separato. Man mano che vengono creati i file**, la funzione creaIFile deve stampare il nome del file creato e il numero di caratteri scritti nel file. **Il callable usato da creaIFile deve creare un nuovo file e scrivere in esso solo a partire dalla conoscenza del nome del file da creare e del nome del file da cui copiare. Se necessario puo` ricevere altri argomenti che pero' non devono essere utilizzati per creare e scrivere nel file. Le stampe NON devono essere effettuate dal callable.**