

Programmazione Avanzata
Appello del 2/2/2022
a.a. 2021-22

1. Scrivere nel file esercizio1.py una funzione che prende in input una sequenza di richieste (numeri) e passa ciascuna richiesta ad una catena di gestori ciascuno dei quali e' una coroutine.
 - Se il numero è compreso tra 0 e 100 allora la richiesta viene gestita dal gestore `gestore_0_100` che stampa "Richiesta {} gestita da gestore_0_100".
 - Se il numero è compreso tra 101 e 200 allora la richiesta viene gestita dal gestore `gestore_101_200` che stampa "Richiesta {} gestita da gestore_101_200".
 - Se il numero comincia con un numero negativo allora la richiesta viene gestita dal gestore `gestore_negativo` che smette di funzionare immediatamente dopo aver stampato "Richiesta {} gestita da gestore_negativo: la catena smette di funzionare".
 - Se il numero è maggiore di 200, allora la richiesta viene gestita dal gestore `gestoreDiDefault` che stampa "Messaggio da gestoreDiDefault: non e' stato possibile gestire la richiesta {}".

Se un gestore tenta di inviare una richiesta al suo successore e si accorge che questo ha smesso di funzionare allora anch'esso deve smettere di funzionare.

NB: nelle suddette stampe la richiesta (il numero) deve comparire al posto delle parentesi graffe.

2. Scrivere nel file esercizio2.py un decoratore di classe `decoraClasse` che aggiunge alla classe decorata il metodo `conQualiArgomenti` che restituisce una tupla contenente
 - i valori di tutti gli argomenti posizionali (compresi quelli variabili)
 - le coppie (chiave, valore) di tutti gli argomenti keyword (compresi quelli variabili)**passati all'ultima invocazione di `__init__`.**
3. Completare l'implementazione della classe **UtenzeAbitazione** nel file esercizio3.py in modo che le sue istanze possano essere nello **stato** ON oppure nello stato OFF. Oltre ai metodi già forniti, l'interfaccia "pubblica" della classe deve contenere i metodi **apri_acqua**, **accendi_riscaldamento**, e **get_consumi**. Quando un'istanza è nello stato ON, i suddetti metodi si comportano come segue:
 - **apri_acqua(num_cl)** incrementa la variabile di istanza `_contatore_acqua` del numero `num_cl` specificato come argomento
 - **accendi_riscaldamento(num_cl)**
 - decrementa `cl_serbatoio` di un valore pari a `num_cl` se `cl_serbatoio > num_cl`; altrimenti usa tutto il gas disponibile, cioè azzera `cl_serbatoio`.
 - aggiorna il registro in modo che la coppia del dizionario associata all'istanza abbia come valore il numero di cl di gas usati in totale fino a quel momento dall'abitazione.
 - se nel momento in cui viene invocato il metodo, `cl_serbatoio` è minore o uguale di `num_cl` (cioè include il caso in cui è inizialmente uguale a 0) allora il metodo dopo aver eventualmente eseguito i due punti precedenti, lancia `ValueError` in modo che venga visualizzata la stringa "Attenzione: il serbatoio condominiale è vuoto".
 - il metodo di istanza **get_consumi()** restituisce una coppia in cui il primo elemento è il consumo di acqua dell'abitazione e il secondo elemento è il consumo di gas dell'abitazione fino a quel momento.

Programmazione Avanzata
Appello del 2/2/2022
a.a. 2021-22

Quando lo stato dell'istanza è OFF, i suddetti tre metodi non fanno niente e non restituiscono niente.

Le utenze vengono attivate e disattivate dai metodi `attiva_utenze` e `disattiva_utenze`, già presenti nella classe insieme ai metodi `__init__`, `disponibilita_gas()` e `get_nome`.

Saranno valutate solo le implementazioni basate sull'approccio state specific. NON si può modificare il codice già scritto e NON si possono aggiungere altre variabili di istanza o di classe.

4. Scrivere la funzione **`max_num_occorrenze`** all'interno del file `esercizio4.py`. Se la funzione ha bisogno di invocare altre procedure, fornire anche queste ultime. La funzione **`max_num_occorrenze`** prende in input una parola **`P`**, una collezione iterabile di nomi di file **`listaDiFile`**, e il parametro **`concorrenza`**. Facendo uso di **`multiprocessing.JoinableQueue`**, la funzione deve calcolare il numero di occorrenze di **`P`** in ciascuno dei file di **`listaDiFile`** e alla fine restituire la coppia **`(fileconmax,max)`**, dove **`fileconmax`** è il file in cui **`P`** compare il massimo numero di volte e **`max`** è questo massimo numero di volte. Il numero di occorrenze di **`P`** in un file deve essere computato con un processo separato per ogni file di **`listaDiFile`**.