**Un semaforo** è come un contatore che **permette di controllare l'accesso a una risorsa condivisa**, come una variabile o un'area di memoria, **da parte di più processi o thread contemporaneamente**.

**Funzioni, metodi e attributi utilizzati con i semafori:**

*threading.Semaphore(value):*

Costruisce un oggetto semaforo con un contatore iniziale pari a value.

*acquire():*

Metodo che acquisisce il semaforo. Se il contatore del semaforo è maggiore di zero, viene decrementato e il metodo ritorna immediatamente. Se il contatore è zero, il thread si blocca finché il semaforo non viene rilasciato da un altro thread:

* Se il contatore del semaforo è maggiore di zero, ci sono permessi disponibili e il thread può acquisire il semaforo, il contatore viene decrementato di uno e il metodo ritorna, dopodichè il thread può procedere senza essere bloccato.
* Se il contatore del semaforo è zero, significa che non ci sono permessi disponibili, il thread che cerca di acquisire il semaforo si blocca “stato di attesa” finché uno o più permessi non vengono rilasciati da altri thread. Quando un altro thread rilascia (release()) il semaforo (incrementando il contatore), il thread bloccato verrà sbloccato e potrà acquisire il semaforo, decrementando il contatore e procedendo con l'esecuzione.

*release():*

Metodo che rilascia il semaforo. Incrementa il contatore del semaforo e sveglia un thread bloccato, se presente.

*Mutex*:

In Python, un mutex (Mutual Exclusion Lock) è uno strumento, utilizzato per gestire l'accesso concorrente a risorse condivise da più thread. Quando più thread lavorano su dati condivisi, si può incorrere in problemi come la race condition (condizione di gara), in cui i risultati dipendono dall'ordine di esecuzione dei thread, classe Lock, che può essere utilizzata come mutex.

*locked():*

Metodo che restituisce True se il semaforo è acquisito da un thread, altrimenti False.

*\_\_enter\_\_() e \_\_exit\_\_():*

Metodi utilizzati per utilizzare il semaforo all'interno di un contesto with ( costrutto del linguaggio Python, utilizzato per gestire risorse che devono essere acquisite e rilasciate in modo sicuro), acquisce automaticamente il semaforo all'ingresso del blocco e lo rilascia alla fine.

Esempio:

Quando si entra in un blocco with SemaphoreManager(value) as semaforo:,

il metodo \_\_enter\_\_() della classe SemaphoreManager viene chiamato,

acquisendo il semaforo.

Al termine del blocco, il metodo \_\_exit\_\_() viene chiamato, rilasciando il semaforo, indipendentemente da eventuali eccezioni sollevate all'interno del blocco.

*acquire(timeout=None):*

Metodo che acquisisce il semaforo, bloccando un Thread per il numero di secondi specificato da timeout. Se timeout è None, il thread si blocca indefinitamente finché il semaforo non è disponibile.

*locked():*

Metodo che restituisce un valore booleano che indica se il semaforo è attualmente bloccato o meno.

*release():*

Metodo che rilascia il semaforo, incrementa il contatore del semaforo e sveglia un thread bloccato, se presente.