Roteiro 2

5 - **Dê uma descrição do que faz cada um desses tipos ou funções.**

A classe Thread, gera uma atividade separadamente do processo atual, que roda separadamente, e pode acessar variáveis globais compartilhadas;

A função Start “avisa” a thread na qual ela foi invocada, que está na hora de executar sua função;

A função Join têm como intuito, fazer com que a thread representante do processo pai, espere com que as threads na qual a função .join() foi acionada, assim, após a finalização da mesma, a processo pai, vai poder retomar seu processamento;

6 - **Rode o programa. O que ele faz? Explique cada linha do resultado da saída do programa.**

O programa tem como saída seu índice na lista de threads, e o valor da variável r, que foi incrementada aleatoriamente, formatados em uma String. (I am “índice”, the results is : “r”).

7 - **Por que as threads do programa não terminam na ordem em que são criadas?**

Porque, como as threads tem um tempo total de processamento elevado, elas terão que passar mais pelo escalonador, e assim, terão fatias de tempo aleatórias, podendo ser uma maior que a outra, fazendo assim, que seu tempo de execução varie e assim dando chance para threads que começaram depois, terminam primeiro.

8 - **O que acontece se você retirar as linhas 21 e 22 do arquivo joinEx.py. O programa precisa dessas linhas para terminar com o resultado esperado?**

nada diferente acontece, pois como o programa pai, não tem nada após essas duas linhas, que estavam o loop e o .join(), então, não faz diferença o pai continuar a thread principal.

9 - **O que a função sys.exit() faz nesse caso?**

Termina o processamento da thread que terminou seu loop;

11 - **Dê uma descrição do que faz cada um desses tipos ou funções.**

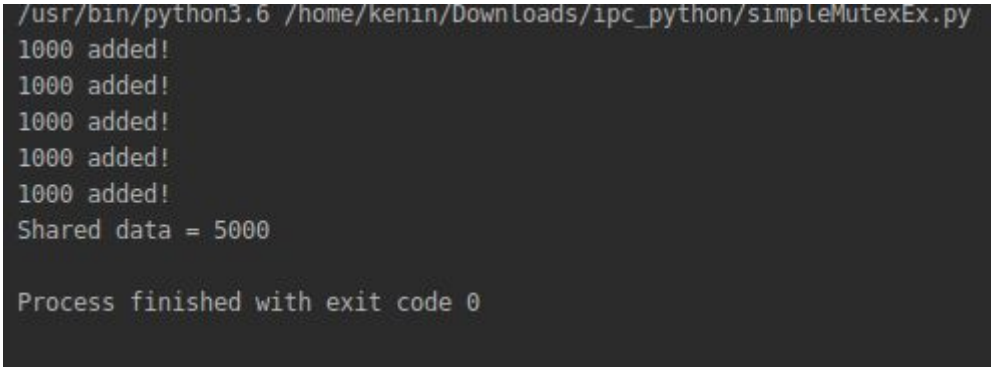
Classe Lock: Uma classe que tem o intuito de servir como locker, tendo dois estados, aberto e fechado.

Lock.acquire(): capaz de bloquear uma classe lock, caso o valor block seja true, e se ao tentar bloquear uma uma classe, ela já estiver bloqueada, retornará um Runtime Error.

Lock.release(): Capaz de liberar um lock bloqueado, e caso o mesmo já esteja liberado, ele retornará um Runtime Error.

12 - **Rode o programa. O que ele faz?**

O programa incrementa a variável global “shared_data” com 1000, em cinco threads, printando na tela “1000 added!” a cada uma, e após isso printa o valor total da variável.



13 - **Retire as diretivas de sincronização "acquire()" e "release()". O que acontece com o programa? Você saberia explicar esse comportamento?**

```
/usr/bin/python3.6 /home/kenin/Downloads/ipc_python/simpleMutexEx.py
1000 added!
1000 added!
1000 added!
1000 added!
1000 added!
Shared data = 2000

Process finished with exit code 0
```

Pois ao remover as funções do lock, todas as threads vão acessar a variável global ao mesmo tempo, criando uma condição de corrida, e assim, apenas uma consegue incrementar a variável.