**Práctica 3- Python (Multithreading)**

En el presente documento se adjuntan unos ejercicios para poner en práctica la programación en multihilo.

Recordad que para la entrega, los ejercicios irán dentro de una carpeta. Hacemos un zip con la siguiente estructura: **[P3-PSP]-AntonioCalabuigPuigvert.zip**. No olvidéis realizar seguimiento con un control de versiones para toda la práctica.

**Ejercicio 1:**

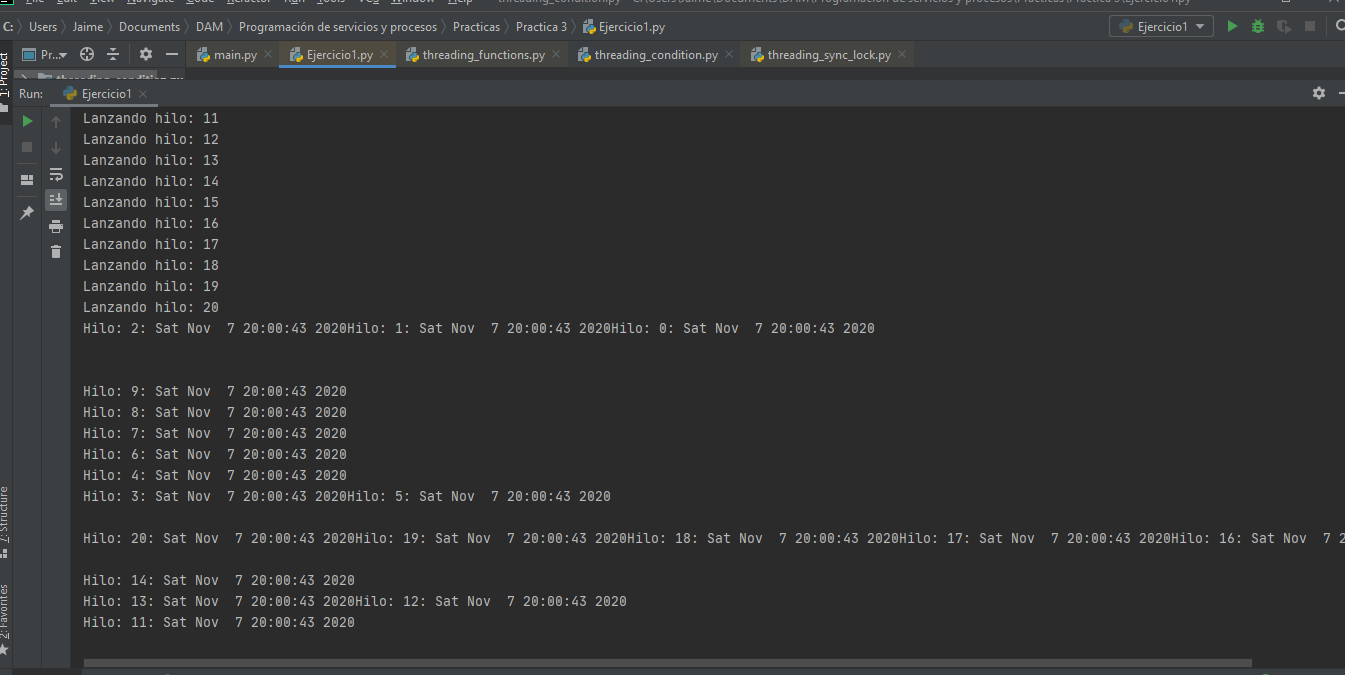
Realizar un programa que genere 20 hilos a partir del programa principal.

Cada hilo ejecutará una función en la que en primer lugar se imprimirá el número de cada hilo empezando desde el 1 hasta el 20.

Esta función realizará una espera de 3 segundos y a continuación imprimirá el número del hilo y cuando ha finalizado.

Finalmente, desde el programa principal, esperamos a que finalicen para juntarlos y mostramos un mensaje final indicando que el programa ha finalizado.

La salida que tenemos que observar es una lista de inicio de procesos ordenada y sin embargo una lista de procesos finalizados desordenada.



**Ejercicio 2:**

En este ejercicio vamos a trabajar los semáforos (from threading import Semaphore)

Para declarar un semáforo: sem = Semaphore(WIDTH). WIDTH indica la cantidad de hilos que pueden ejecutar la sección crítica (1..N)

Realizar un programa que genere 20 hilos a partir del programa principal.

Cada hilo ejecutará una función en la que se accede a un túnel de uno en uno, tomando un tiempo determinado (3s) en la que en primer lugar se imprimirá el número de cada hilo, esperaremos 3s para adquirir el semáforo (.acquire()), imprimimos que ha entrado al túnel, esperaremos 1s para liberar el semaforo (.release()) e imprimimos que ese hilo ha salido del túnel.

Finalmente, desde el programa principal, esperamos a que finalicen para juntarlos y mostramos un mensaje final indicando que el programa ha finalizado.

La salida que tenemos que observar es una lista de inicio de procesos ordenada y sin embargo una lista de procesos finalizados desordenada.

**Ejercicio 3:**

En este ejercicio vamos a trabajar los locks (from threading import Lock). Para declarar un lock: lock = Lock().

Realizar un programa que genere 20 hilos a partir del programa principal.

Cada hilo ejecutará una función en la que simularemos 20 personas bebiendo de un vaso. Cada hilo accede al lock, durante un segundo beberá del vaso y al liberarlo, imprimimos que el hilo con número está satisfecho y a continuación se ejecutará el siguiente hilo.

Finalmente, desde el programa principal, esperamos a que finalicen para juntarlos y mostramos un mensaje final indicando que el programa ha finalizado.

La salida que tenemos que observar es una lista de inicio de procesos ordenada y sin embargo una lista de procesos finalizados desordenada.

**Ejercicio 4 (Para nota):**

En este ejercicio vamos a trabajar los Conditions (from threading import Condition). Este tipo de bloqueo son los más sofisticados ya que tenemos las funciones de .notify(), .wait(), .release(), .acquire().

Para declarar un lock: cond = Condition().

Realizar un programa que genere 20 hilos a partir del programa principal.

Cada hilo ejecutará una función en la que simularemos 20 personas bebiendo de un vaso. En este caso cada hilo tomará el semáforo, mientras pueda, sino esperará 0,5s hasta poder adquirirlo (trabajamos los tiempos de espera de los procesos), una vez conseguimos acceder al semáforo, ejecutamos la función que nos da la

posibilidad de beber. Bebemos durante un tiempo aleatorio (from random import randint, s = randint(1,10)), al finalizar avisamos al siguiente proceso en espera y liberamos la condición.

La salida que tenemos que observar es una lista de inicio de procesos ordenada y sin embargo una lista de procesos finalizados desordenada.

**Ejemplo de salida:**

[+] En hilo 1

[+] En hilo 2

[+] En hilo 3

[+] En hilo 4

[+] En hilo 5

[+] En hilo 7

[+] En hilo 8

[+] En hilo 6

[+] En hilo 10

[+] En hilo 9

[+] En hilo 12

[+] En hilo 11

[+] En hilo 13

[+] En hilo 14

[+] En hilo 15

[+] En hilo 16

[+] En hilo 17

[+] En hilo 18

[+] En hilo 19

[+] En hilo 20

[+] Bebiendo hilo 3.. esperate 10 secs

[-] estaba de muerte, hilos --3--

[-] hilo 3 finalizado

[+] Bebiendo hilo 4.. esperate 9 secs

[-] estaba de muerte, hilos --4--

[-] hilo 4 finalizado

[+] Bebiendo hilo 1.. esperate 1 secs

[-] estaba de muerte, hilos --1--

[-] hilo 1 finalizado

[+] Bebiendo hilo 2.. esperate 10 secs

[-] estaba de muerte, hilos --2--

[-] hilo 2 finalizado

[+] Bebiendo hilo 5.. esperate 6 secs

[-] estaba de muerte, hilos --5--

[-] hilo 5 finalizado

[+] Bebiendo hilo 7.. esperate 5 secs

[-] estaba de muerte, hilos --7--

[-] hilo 7 finalizado

[+] Bebiendo hilo 8.. esperate 4 secs

[-] estaba de muerte, hilos --8--

[-] hilo 8 finalizado

[+] Bebiendo hilo 6.. esperate 2 secs

[-] estaba de muerte, hilos --6--

[-] hilo 6 finalizado

[+] Bebiendo hilo 10.. esperate 8 secs

[-] estaba de muerte, hilos --10--

[-] hilo 10 finalizado

[+] Bebiendo hilo 12.. esperate 10 secs

[-] estaba de muerte, hilos --12--

[-] hilo 12 finalizado

[+] Bebiendo hilo 11.. esperate 9 secs

[-] estaba de muerte, hilos --11--

[-] hilo 11 finalizado

[+] Bebiendo hilo 13.. esperate 7 secs

[-] estaba de muerte, hilos --13--

[-] hilo 13 finalizado

[+] Bebiendo hilo 16.. esperate 8 secs

[-] estaba de muerte, hilos --16--

[-] hilo 16 finalizado

[+] Bebiendo hilo 9.. esperate 5 secs

[-] estaba de muerte, hilos --9--

[-] hilo 9 finalizado

[+] Bebiendo hilo 14.. esperate 5 secs

[-] estaba de muerte, hilos --14--

[-] hilo 14 finalizado

[+] Bebiendo hilo 15.. esperate 4 secs

[-] estaba de muerte, hilos --15--

[-] hilo 15 finalizado

[+] Bebiendo hilo 17.. esperate 8 secs

[-] estaba de muerte, hilos --17--

[-] hilo 17 finalizado

[+] Bebiendo hilo 20.. esperate 6 secs

[-] estaba de muerte, hilos --20--

[-] hilo 20 finalizado

[+] Bebiendo hilo 19.. esperate 7 secs

[-] estaba de muerte, hilos --19--

[-] hilo 19 finalizado

[+] Bebiendo hilo 18.. esperate 7 secs

[-] estaba de muerte, hilos --18--

[-] hilo 18 finalizado

[\*] all threads finished