Procesos e Hilos en Python

Alan, Guillermo, Jorge

October 2020

1. Procesos en Python

Python tiene una paquetería específica para la creación de procesos y otra para la creación de hilos, en esta parte nos enfocaremos en la primera. La paquetería `multiprocessing` permite la creación de procesos. En esta, cada proceso es creado con la creación de un objeto `Process` y después usando el método `start`.
Un ejemplo de su uso es el siguiente:

```
from multiprocessing import Process


def greeting():
    print 'hello world'

if __name__ == '__main__':
    p = Process(target=greeting)
    p.start()
    p.join()
```

La utilidad de la condicional `if __name__ == '__main__'` radica en que el interprete de python asigna un nombre `__name__` a cada módulo importado, de forma que `'__main__'` corresponde al programa principal. Así al correr el código no habrá problemas no deseados como el comienzo de nuevos procesos sin ser solicitados. También podemos mostrar los PID de los procesos usando el método `getpid()` de la paquetería `os`. si estamos manejando un proceso hijo, podemos recuperar el PID del proceso padre usando `os.getppid()`:

```
from multiprocessing import Process
import os

def info():
    print ('El nombre del modulo:', __name__)
    if hasattr(os, 'getppid'):
        print ('Proceso padre PID:', os.getppid())
    print ('PID:', os.getpid())

def greeting(nombre):
    print('\nSoy codigo del proceso creado')
    print (f'hello {nombre}')
    print ('Proceso padre PID:', os.getppid())
    print ('PID:', os.getpid())

if __name__ == '__main__':
```

```
    info()
    p = Process(target=greeting, args=('Comunidad de la LCD',))
    p.start()
    p.join()
```

2. Hilos en Python

Para el caso de la creación de hilos tenemos una paquetería nos recuerda bastante a *C* y es `threading` como podemos ver en el código siguiente:

```
import threading

# how many threads we want to start
THREADS_COUNT = 3

class Threaded_worker(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
    def run(self):
        threadName = threading.currentThread().getName()
        print("Hello, I am the thread %s" % threadName)

print('Starting %d threads...' % THREADS_COUNT)
for i in range(THREADS_COUNT):
    td = Threaded_worker()
    td.start()
```

Como podemos observar `THREADS_COUNT` va a declarar el número de hilos que vamos usar, algo curioso es que para este caso tenemos que declarar una clase que donde vamos a definir 2 funciones una para llamar al hilo y otra para iniciarlo `__init__` y `run`, respectivamente donde run recibirá el atributo de hilo que fue llamado previamente para asignarle un nombre con `threading.currentThread().getName()` para echarlo a andar.

Finalmente el último bloque del código imprime los códigos llamándolos uno por uno,en las funciones de la clase definida previamente, asignándoles un identificador y empezandolos con `.start()`.