JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

Thread Synchronisation in Python

Softwareentwicklung
4CHIT 2016/17

Martin Wölfer

Version 0.2
Begonnen am 16.11.2016
Beendet am 20.11.2016

Note:
Betreuer: RAFM

Inhaltsverzeichnis

1 Einführung

Schreibe ein Programm, welches die Summe von 1 bis zu einer von dem/der Benutzer/in einzugebenden (potentiell sehr großen) Zahl mithilfe von drei Threads berechnet!

1.1 Grundanforderungen

- Zwei eigene Klassen (Consumer und Producer) erben von Thread (E1 und V1)
- Die zwei Klassen sind über einen Queue verbunden
- Der Erzeuger E1 sucht nach Primzahlen. Jede gefundene Primzahl wird über die Queue an den Verbraucher V1 geschickt
- Der Verbraucher gibt die empfangene Zahl in der Konsole aus und schreibt sie außerdem in eine simple Textdatei
- Erzeuger und Verbraucher stimmen sich über `Queue.task_done()` und `Queue.join()` ab
- Kommentare und Sphinx-Dokumentation
- Kurzes Protokoll über deine Vorgangsweise, Aufwand, Resultate, Beobachtungen, Schwierigkeiten, ... Bitte sauberes Dokument erstellen! (Kopf- und Fußzeile etc.)

1.2 Erweiterungen

- Ein weiterer Thread nimmt Benutzereingaben entgegen
- Dieser Thread kann als ein weiterer Erzeuger E2 gesehen werden
- Wird eine (potentiell sehr große) Zahl eingegeben, so wird in einem weiteren Verbraucher V2 überprüft, ob es sich bei dieser produzierten Zahl um eine Primzahl handelt
- E2 und V2 müssen sich nicht über `task_done()` absprechen, d.h. E2 kann mehrere Aufträge in die Queue schicken, bevor V2 mit der Bearbeitung fertig ist
- Wird ëxitëingegeben, so werden alle Threads sauber beendet
- Achte auf Fehlerfälle!

2 Vorgangsweise Grundanforderung

Zuerst wurden die `Erzeuger`- und `Vebraucher`klassen erstellt. Diese waren recht simpel zu erzeugen da man in der Erzeuger Klasse jediglich eine Endlosschleife hat die eine Zahl hochzählt, und jedesmal überpüft ob diese Zahle eine Primzahl ist.

Zum Überprüfen ob es eine Primzahl ist wurde folgender code verwendet:

```
def ist_primzahl(number):
    """
    :param number: The number to be checked if it is prime

    :return: True or False depending if the argument given is a prime number

    """
    for i in range(2, number):
        #If the number can be divided through another number which is not itself or 0 or 1 the method
            return False
        if number % i == 0:
        return False
    #In any other case it returns True
    return True
```

Das Schreiben in ein File wurde mit der Funktion `open()` realisiert. Das schwieriege nun für mich persönlich war es diese Threads sauber zu beenden.

Um die erweiterte Aufgabe zu realisieren muss man irgendwann die Endlosschleife beenden, und ich habe die Zahl 100 gewählt. Weil ich nicht weiß wie man einen Thread sauber schließt habe ich einfach die Endlosschleife im Producer und Consumer beendet sobald die Zahl größer als 100 ist.

3 Vorgangsweise Erweiterungen

Es wurde wieder eine Erzeuger und Verbraucherklasse erstellt welche von `threading.Thread` erben. Der Erzeuger nimmt nun eine Benutzereingabe entgegen, und gibt diese Eingabe wenn es eine Zahl ist einfach in die queue. Wenn die Eingabe `'exit'` ist, wird das Programm beendet.

4 Schwierigkeiten

Bei der erweiterten Aufgabe war für mich persönlich auch das größte Problem wieder das Schließen von den Threads. Ich habe probiert diese Threads über `sys.exit(0)` zu beenden, aber das hat auch nicht funktioniert.

Nach Recherche habe ich nur die Funktion von `threading.Thread ._stop()` gefunden, aber diese hat auch zu keinem Ergebnis geführt.

4.1 Github-Link

Hier Klicken für Git Reposiory