Protokoll: Queue

Aufgabe

Schreibe ein Programm, welches ein simples Erzeuger-Verbraucher-Muster implementiert!

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.)

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 "exit" eingegeben, so werden alle Threads sauber beendet
- Achte auf Fehlerfälle!

Vorgangsweise

Verstehen wie Queues funktionieren und wie mehrere Klassen über eine Queue kommunizieren können. Den bereitgestellten Quelltext als Vorlage benutzen. Überlegen wie man am besten Primzahlen bekommt.

Aufwand

Den Quelltext zu schreiben hat ca. 30min – 1h gedauert.
Die Sphinx-Dokumentation hat diesmal einwandfrei funktioniert mit nur ein paar Klicks.

Resultat

Ich habe nur die Grundanforderungen gemacht, da ich zum Teil es nicht verstehe.
Das Programm an sich endet nie, man muss es manuell abbrechen (gewollt?)
Es werden "alle" Primzahlen ausgegeben, angefangen bei 2 (1 ist keine Primzahl)

Schwierigkeiten

Die **if** Unterscheidung um die Primzahlen zu filtern hat zu mit der Zahl 11 gestartet. Weil alle zahlen davor durch 2,3,5 oder 7 teilbar sind (auch wenn 2,3,5 und 7 Primzahlen sind). Gelöst habe ich dieses Problem mit einer lokalen Liste direkt in der **if Unterscheidung** mit **if number in [2,3,5,7]…** . Hätte nicht gedacht, dass das funktionieren würde.

Mann muss das File in der **while True** Schleife öffnen und schließen, aber wenn man das macht, dann überschreibt sich das File immer wieder mit der neuen Zahl. Es außerhalb zu öffnen und außerhalb zu schließen funktioniert auch nicht. Meine Lösung war dazu eine Liste zu erstellen, die die Nummer nach jedem Durchlauf hinzufügt mit **liste.append(str(number))** und diese Liste mit **"\n".join(liste)** in die File zu schreiben. (Wahrscheinlich nicht die beste Lösung, aber es funktioniert ganz gut)