
A07 Protokoll

Thread Synchronisation in Python : Queues

SEW
4CHIT 2016/17

Mladen Vojnovic

Note:
Betreuer:RAFW

Version 0.2
Begonnen am 20.11.2016
Beendet am 20.11.2016

Inhaltsverzeichnis

1	Einführung	1
1.1	Grundanforderungen	1
1.2	Erweiterungen	1
1.3	Voraussetzungen	1
1.4	Aufgabenstellung	1
2	Ergebnisse	2
2.1	Zwei eigene Klassen (Consumer und Producer) erben von Thread (E1 und V1) . . .	2
2.2	Die zwei Klassen sind über einen Queue verbunden	2
2.3	E1 sucht nach Primzahlen, ueber Queue an V1	2
2.4	V1 gibt empfangene Zahl aus und schreibt in ein text file	2
2.5	Der ganze Code	3
2.6	Consol Outpu bei Eingabe 1000	3

1 Einführung

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

1.3 Voraussetzungen

- Python Kenntnisse
- threading Kenntnisse

1.4 Aufgabenstellung

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

2 Ergebnisse

2.1 Zwei eigene Klassen (Consumer und Producer) erben von Thread (E1 und V1)

```

1 import threading
2 class E1(threading.Thread):
3 class V1(threading.Thread):

```

2.2 Die zwei Klassen sind über einen Queue verbunden

Derselbe Konstruktor in beiden Klassen:

```

1 def __init__(self, queue):
2     """
3     :param queue: Die queue welche auch der verbraucher thread hat
4     """
5     threading.Thread.__init__(self)
6     self.queue = queue

```

2.3 E1 sucht nach Primzahlen, ueber Queue an V1

```

1 def run(self):
2     number = 2 #eine lokale variable, beginnt ab 2 da 0,1,2 keine Primzahlen sind
3     b = True
4     while number < 10000: #BIS 10000 sucht das Programm nach Primzahlen
5         number += 1 #+1 zaehler
6         b = True #lokale variable zum pruefen ob die zahl eine primzahl ist
7         for i in range(2, number): #for schleife zum pruefen ob primzahl
8             if number % i == 0:
9                 b = False #falls keine primzahl(ohne rest dividierbar)
10                break
11            if b == True: #falls es doch eine primzahl ist bleibt b true und somit wird
12                #hier dann auch queue.put benutzt um den verbraucher zu
13                # benachrichtigen
14                self.queue.put(number)
15                self.queue.join()
16            number = "fertig" #den verbraucher benachrichtigen das die zaehlung vorbei ist
17            self.queue.put(number) #den verbraucher benachrichtigen das die zaehlung
18                # vorbei ist

```

2.4 V1 gibt empfangene Zahl aus und schreibt in ein text file

```

1 def run(self):
2     tf = open("output.txt", "w") #oeffnen des text files, "w" ->
3                                     # falls keins vorhanden -> neu erstellen
4                                     #falls doch -> ueberschreiben
5     while True:
6         number = self.queue.get() #erhaellt die variable aus der queue und
7                                     # signalisiert das der erzeuger weitermachen kann
8         if number == "fertig": #prueft ob die zaehlung vorbei ist
9             break
10        print("Primzahl: %d" % number) #ausgabe in die konsole
11        self.queue.task_done()
12        tf.write("Primzahl: " + str(number) + "\n") #schreiben in das text file

```

2.5 Der ganze Code

```

1 import threading
2 import queue

4 class El(threading.Thread):
5     def __init__(self, queue):
6         """
7
8         :param queue: Die queue welche auch der verbraucher thread hat
9         """
10        threading.Thread.__init__(self)
11        self.queue = queue

12    def run(self):
13        number = 2 #eine lokale variable, beginnt ab 2 da 0,1,2 keine Primzahlen sind
14        b = True
15        while number < 10000:#bis 10000 sucht das Programm nach Primzahlen
16            number += 1 #+1 zaehler
17            b = True #lokale variable zum pruefen ob die zahl eine primzahl ist
18            for i in range(2, number): #for schleife zum pruefen ob primzahl
19                if number % i == 0:
20                    b = False #falls keine primzahl(ohne rest dividierbar)
21                    break
22            if b == True: #falls es doch eine primzahl ist bleibt b true und somit wird
23                #hier dann auch queue.put benutzt um den verbraucher zu
24                #benachrichtigen
25                self.queue.put(number)
26                self.queue.join()
27            number = "fertig" #den verbraucher benachrichtigen das die zaehlung vorbei ist
28            self.queue.put(number) #den verbraucher benachrichtigen das die zaehlung
29            #vorbei ist

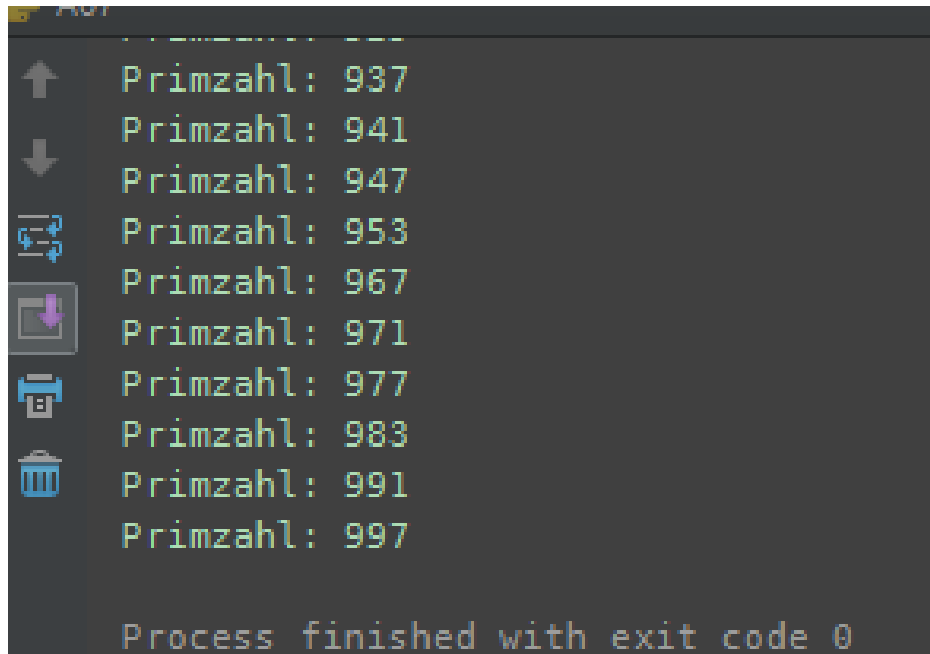
32 class V1(threading.Thread):
33     def __init__(self, queue):
34         """
35
36         :param queue: Die queue welche auch der verbraucher thread hat
37         """
38        threading.Thread.__init__(self)
39        self.queue = queue

40    def run(self):
41        tf = open("output.txt", "w") #oeffnen des text files, "w" ->
42                                     # falls keins vorhanden -> neu erstellen
43                                     #falsl doch -> ueberschreiben
44
45        while True:
46            number = self.queue.get() #erhaellt die variable aus der queue und
47                                     # signalisiert das der erzeuger weitermachen kann
48            if(number == "fertig"): #prueft ob die zaehlung vorbei ist
49                break
50            print("Primzahl: %d" % number) #ausgabe in die konsole
51            self.queue.task_done()
52            tf.write("Primzahl: " + str(number) + "\n") #schreiben in das text file

54 if __name__ == "__main__":
55     queue = queue.Queue()
56     t1 = El(queue)
57     t2 = V1(queue)
58     t1.start()
59     t2.start()
60     t1.join()
61     t2.join()

```

2.6 Consol Outpu bei Eingabe 1000



```
Primzahl: 937
Primzahl: 941
Primzahl: 947
Primzahl: 953
Primzahl: 967
Primzahl: 971
Primzahl: 977
Primzahl: 983
Primzahl: 991
Primzahl: 997

Process finished with exit code 0
```

Abbildung 1: Consolen Output