

---

# A06 Protokoll

## Thread Synchronisation in Python

---

SEW  
4CHIT 2016/17

Mladen Vojnovic

Note:  
Betreuer:RAFW

Version 0.2  
Begonnen am 23.09.2016  
Beendet am 29.09.2016

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Grundanforderungen . . . . .	1
1.2	Erweiterungen . . . . .	1
1.3	Voraussetzungen . . . . .	1
1.4	Aufgabenstellung . . . . .	1
<b>2</b>	<b>Ergebnisse</b>	<b>2</b>
2.1	Eine Klasse erbt von Thread . . . . .	2
2.2	Gemeinsamer Lock und Counter . . . . .	2
2.3	Paramter im Konstruktor . . . . .	2
2.4	Korrekt aufsummieren . . . . .	2
2.5	Der ganze Code . . . . .	3
2.6	Consol Outpu bei Eingabe 1000 . . . . .	5
<b>3</b>	<b>Interpretation</b>	<b>5</b>
<b>4</b>	<b>Massnahmen</b>	<b>5</b>

# 1 Einführung

## 1.1 Grundanforderungen

- Eine eigene Klasse erbt von Thread
- Die Klasse definiert eine gemeinsame Lock sowie einen gemeinsamen Counter
- Im Konstruktor wird über einen Parameter bestimmt, für welche Zahlen dieser Thread zuständig ist
- In der run-Methode wird die Summe korrekt aufsummiert, wobei der Zugriff auf den Counter über die Lock threadsicher gestaltet wird (with-Statement)
- Kommentare und Sphinx-Dokumentation
- Kurzes Protokoll über deine Vorgangsweise, Aufwand, Resultate, Beobachtungen, Schwierigkeiten, ... (Kopf- und Fußzeile etc.)

## 1.2 Erweiterungen

- Miss die Laufzeit!
- Untersuche, wie sich die Laufzeit auf deinem System verhält, wenn du es mit mehr oder weniger Threads verwendest, z.B. Single Threaded (d.h. nur im main-Thread), mit 2 Threads, mit 3 Threads, ...
- Interpretiere die Ergebnisse und halte deine Erkenntnisse im Protokoll fest! Warum verhält es sich so?
- Finde eine Möglichkeit, wie die Performance verbessert werden kann und eventuelle Beschränkungen umgangen werden können!

## 1.3 Voraussetzungen

- Python Kenntnisse
- threading Kenntnisse

## 1.4 Aufgabenstellung

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!

## 2 Ergebnisse

### 2.1 Eine Klasse erbt von Thread

```
1 import threading
   class A06(threading.Thread):
```

### 2.2 Gemeinsamer Lock und Counter

```
2 counter = 0
   lock = threading.Lock()
```

### 2.3 Paramter im Konstruktor

```
2     def __init__(self, tNumbers):
3         """
4         :param tNumbers: Eine Liste aus Nummern welche der jeweilige Thread zu counter dazu addieren soll
5         """
```

### 2.4 Korrekt aufsummieren

```
1     def run(self):
2         for i in range(len(self.tNumbers)):
3             with A06.lock:
4                 # hier wird counter immer um die Zahl an der Stelle 0 der liste erhoeht
5                 # dann wird die Stelle 0 geloescht und alles rueckt nach links(2 -> 1, 1 -> 0, etc.)
6                 A06.counter += self.tNumbers[0]
7                 print("Current Value: " + str(A06.counter))
8                 del(self.tNumbers[0])
```

## 2.5 Der ganze Code

```

1 import threading
2 import time

4 class A06(threading.Thread):
    # globale Variable counter welche von jedem Thread benutzt wird
6     counter = 0
    lock = threading.Lock()
8     def __init__(self, tNumbers):
        """
10         :param tNumbers: Eine Liste aus Nummern welche der jeweilige Thread zu counter dazu addieren soll
12         """
        threading.Thread.__init__(self)
        self.tNumbers = tNumbers
14     def run(self):
        for i in range(len(self.tNumbers)):
16             with A06.lock:
18                 # hier wird counter immer um die Zahl an der Stelle 0 der liste erhoeht
                # dann wird die Stelle 0 geloescht und alles rueckt nach links(2 -> 1, 1 -> 0, etc.)
20                 A06.counter += self.tNumbers[0]
                print("Current Value: " + str(A06.counter))
22                 del(self.tNumbers[0])

24 def getInput():
    """
26     :return: Die eingabe
28     """
    # Die eingabe gleich als int
30     n = int(input("Bitte geben sie die Zahl ein bis zu der addiert werden soll."))
    return n

32 def main1Threads(numberHelp):
    """
34     :param numberHelp: die eingabe
36     """
38     anfang = time.time()#Anfang des Programms
    # numberHelp = getInput()

40     # Die Listen fuer die Threads
42     tN1 = []
    for i in range(1, numberHelp + 1):
44         tN1.append(i)

46     # Die Threads + start() und join()
    t = A06(tN1)

48     t.start()

50     t.join()

52     lz = time.time() - anfang#Die ausgerechnete Laufzeit, Ende - Anfang
54     print("mit 1 Threads: " + str(lz))

56 def main2Threads(numberHelp):
    """
58     :param numberHelp: die eingabe
60     """
62     anfang = time.time()#Anfang des Programms
    # numberHelp = getInput()

64     #Die Listen fuer die Threads
    tN1 = []
    tN2 = []
66     for i in range(1, numberHelp + 1, 2):
68         tN1.append(i)
    for i in range(2, numberHelp + 1, 2):
70         tN2.append(i)

```

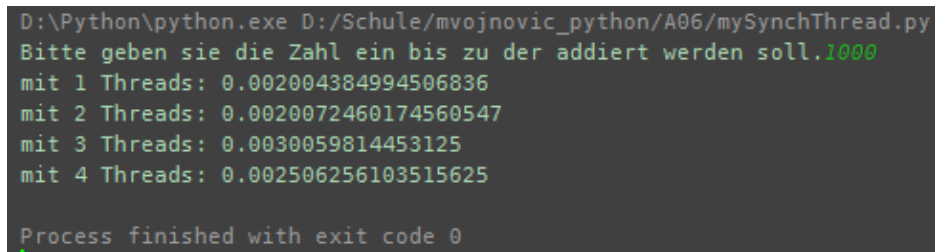
```

72     #Die Threads + start() und join()
73     t = A06(tN1)
74     t2 = A06(tN2)
75
76     t.start()
77     t2.start()
78
79     t.join()
80     t2.join()
81
82     lz = time.time() - anfang #Die ausgerechnete Laufzeit, Ende - Anfang
83     print("mit 2 Threads: " + str(lz))
84
85 def main3Threads(numberHelp):
86     """
87
88     :param numberHelp: die eingabe
89     """
90     anfang = time.time() #Anfang des Programms
91     #numberHelp = getInput()
92
93     # Die Listen fuer die Threads
94     tN1 = []
95     tN2 = []
96     tN3 = []
97     for i in range(1, numberHelp+1, 3):
98         tN1.append(i)
99     for i in range(2, numberHelp+1, 3):
100         tN2.append(i)
101     for i in range(3, numberHelp+1, 3):
102         tN3.append(i)
103
104     # Die Threads + start() und join()
105     t = A06(tN1)
106     t2 = A06(tN2)
107     t3 = A06(tN3)
108
109     t.start()
110     t2.start()
111     t3.start()
112
113     t.join()
114     t2.join()
115     t3.join()
116
117     lz = time.time() - anfang #Die ausgerechnete Laufzeit, Ende - Anfang
118     print("mit 3 Threads: " + str(lz))
119
120 def main4Threads(numberHelp):
121     """
122
123     :param numberHelp: die eingabe
124     """
125     anfang = time.time() #Anfang des Programms
126     #numberHelp = getInput()
127
128     # Die Listen fuer die Threads
129     tN1 = []
130     tN2 = []
131     tN3 = []
132     tN4 = []
133     for i in range(1, numberHelp+1, 4):
134         tN1.append(i)
135     for i in range(2, numberHelp+1, 4):
136         tN2.append(i)
137     for i in range(3, numberHelp+1, 4):
138         tN3.append(i)
139     for i in range(3, numberHelp + 1, 4):
140         tN4.append(i)
141
142     # Die Threads + start() und join()

```

```
144     t = A06(tN1)
145     t2 = A06(tN2)
146     t3 = A06(tN3)
147     t4 = A06(tN4)
148
149     t.start()
150     t2.start()
151     t3.start()
152     t4.start()
153
154     t.join()
155     t2.join()
156     t3.join()
157     t4.join()
158
159     lz = time.time() - anfang#Die ausgerechnete Laufzeit, Ende – Anfang
160     print("mit 4 Threads: " + str(lz))
161
162 def main():
163     #hier rufe ich die einzelnen Methoden auf und dazwischen muss ich
164     #counter auf 0 setzen da es sich um ein Globale Variable handelt
165     numberHelp = getInput()
166     main1Threads(numberHelp)
167     A06.counter = 0
168     main2Threads(numberHelp)
169     A06.counter = 0
170     main3Threads(numberHelp)
171     A06.counter = 0
172     main4Threads(numberHelp)
173
174 main()
```

## 2.6 Consol Output bei Eingabe 1000



```
D:\Python\python.exe D:/Schule/mvojinovic_python/A06/mySynchThread.py
Bitte geben sie die Zahl ein bis zu der addiert werden soll.1000
mit 1 Threads: 0.002004384994506836
mit 2 Threads: 0.0020072460174560547
mit 3 Threads: 0.0030059814453125
mit 4 Threads: 0.002506256103515625

Process finished with exit code 0
```

Abbildung 1: Consolen Output

## 3 Interpretation

Es ist erkennbar bei wiederholten Versuchen dass die Erhoeung der Thread-Anzahl zur verlangsamung des Programmes fuehrt

## 4 Massnahmen

So wenige Threads wie moeglich verwenden