プログラミングA2
第11回

練習問題解答例

練習問題：fizz_buzz_mt.py

- fizz_buzz関数内からThreadオブジェクトおよびスレッド名を出力せよ
- 現在のスレッド一覧を取得し，for-in文でThreadオブジェクトおよびスレッド名を出力せよ
- モジュール名を名前空間として利用できるようにimport文を改めよ

実行例：fizz_buzz_mt.py

実行例

```
PS C:¥Users¥admin¥Desktop¥ProA2> python fizz_buzz_mt.py
fizz_buzz start：50000000
fizz_buzz start：40000000
fizz_buzz start：30000000
fizz_buzz start：20000000
fizz_buzz start：10000000
start
```

← Mainスレッドの「print("start")」

```
<_MainThread(MainThread, started 21308)> MainThread
<Thread(Thread-1, started 16624)> Thread-1
<Thread(Thread-2, started 2576)> Thread-2
<Thread(Thread-3, started 22836)> Thread-3
<Thread(Thread-4, started 22972)> Thread-4
<Thread(Thread-5, started 9884)> Thread-5
```

← スレッド一覧

```
<Thread(Thread-5, started 9884)> Thread-5
fizz_buzz end：10000000 所要時間：11.67秒        累積時間：12.13秒
<Thread(Thread-4, started 22972)> Thread-4
fizz_buzz end：20000000 所要時間：21.65秒        累積時間：21.96秒
<Thread(Thread-3, started 22836)> Thread-3
fizz_buzz end：30000000 所要時間：38.44秒        累積時間：38.60秒
<Thread(Thread-2, started 2576)> Thread-2
fizz_buzz end：40000000 所要時間：49.55秒        累積時間：49.59秒
<Thread(Thread-1, started 16624)> Thread-1
fizz_buzz end：50000000 所要時間：51.45秒        累積時間：51.45秒
```

← 各スレッドの終わり間際に実行される

解答例：fizz_buzz_mt.py

fizz_buzz_mt.py

```
import threading

def fizz_buzz(num, bgn0):
    bgn = time.time()
    print(f"fizz_buzz start：{num}")
    省略
    end = time.time()
    print(threading.current_thread(), threading.current_thread().name)
    print(f"fizz_buzz end：{num}¥t所要時間：{end-bgn:.2f}秒¥t累積時間：{end-
bgn0:.2f}秒")
    return result_lst

if __name__ == "__main__":
    bgn0 = time.time()
    num_lst = [i*10000000 for i in range(5, 0, -1)]
    for num in num_lst:
        th = threading.Thread(target=fizz_buzz, args=(num, bgn0))
        th.start()
    print("start")
    for th in threading.enumerate():
        print(th, th.name)
```

練習問題：counter_lck.py

Lockオブジェクトを導入し，スレッド実行を制御することで同時更新を防ぐコードを実装せよ.

実行例

```
PS C:¥Users¥admin¥Desktop¥ProA2> python counter_lck.py
1秒sleep
くさを追加 2秒sleepdefaultdict(<class 'int'>, {'くさ': 1, 'どく': 0})

どくを追加 4秒sleepdefaultdict(<class 'int'>, {'くさ': 1, 'どく': 1})

くさを追加 defaultdict(<class 'int'>, {'くさ': 2, 'どく': 1})
5秒sleep
どくを追加2秒sleep
defaultdict(<class 'int'>, {'くさ': 2, 'どく': 2})
くさを追加 5秒sleepdefaultdict(<class 'int'>, {'くさ': 3, 'どく': 2})

：
集計結果： defaultdict(<class 'int'>, {'くさ': 3, 'どく': 3, 'ほのお': 3, 'ひこう': 1, 'みず': 3})
```

解答例：counter_lck.py

counter_lck.py

```
def counter(cnt_dct, type_, lck):
    lck.acquire()
    now = cnt_dct[type_]
    time.sleep()
    cnt_dct[type_] = now+1
    lck.release()
    print(f"{type_}を追加", cnt_dct)

if __name__ == "__main__":
    lck = threading.Lock()
    for type_lst in types[:9]:
        for type_ in type_lst:
            th = threading.Thread(
                                  target=counter,
                                  args=(cnt_dct, type_, lck)
                                 )
            th.start()
    省略
```

← アンロックとprintのタイミングを逆にするとどうなる？？