# QGIS Application - Bug report #20878
# Multiprocessing with QGis

2018-12-24 12:08 PM - Alexey T

| | | | |
|---|---|---|---|
| **Status:** | Open | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Category:** | PyQGIS Console | | |
| **Affected QGIS version:** | 3.4.1 | **Regression?:** | No |
| **Operating System:** | win7 | **Easy fix?:** | No |
| **Pull Request or Patch supplied:** | No | **Resolution:** | |
| **Crashes QGIS or corrupts data:** | No | **Copied to github as #:** | 28697 |

**Description**

I tried to make some tasks in parallel with qgis.

There is the test script

```
import concurrent.futures
import time, random             # add some random sleep time

offset = 2                      # you don't supply these so
output1 = list()
output2 = list()
output3 = list()
start = time.time()          # let's see how long this takes

# we can swap out ProcessPoolExecutor for ThreadPoolExecutor
with concurrent.futures.ProcessPoolExecutor() as executor:
    for out1, out2, out3 in executor.map(procedure, range(0, 10)):
        # put results into correct output list
        output1.append(out1)
        output2.append(out2)
        output3.append(out3)
finish = time.time()
# these kinds of format strings are only available on Python 3.6:
# time to upgrade!
print(f'original inputs: {repr(output1)}')
print(f'total time to execute {sum(output2)} = sum({repr(output2)})')
print(f'time saved by parallelizing: {sum(output2) - (finish-start)}')
print(f'returned in order given: {repr(output3)}')

def calc_stuff(parameter=None):   # these are examples.
    sleep_time = random.choice([0, 1, 2, 3, 4, 5])
    time.sleep(sleep_time)
    return parameter / 2, sleep_time, parameter * parameter

def procedure(j):               # just factoring out the
    parameter = j * offset        # procedure
    # call the calculation
    return calc_stuff(parameter=parameter)
```

This script doesn't behave as expected. Instead it launches 3 additional QGIS instances.

## History

**#1 - 2018-12-24 12:09 PM - Alexey T**

```
import concurrent.futures
import time, random             # add some random sleep time

offset = 2                      # you don't supply these so
output1 = list()
output2 = list()
output3 = list()
start = time.time()          # let's see how long this takes

# we can swap out ProcessPoolExecutor for ThreadPoolExecutor
with concurrent.futures.ProcessPoolExecutor() as executor:
    for out1, out2, out3 in executor.map(procedure, range(0, 10)):
        # put results into correct output list
        output1.append(out1)
        output2.append(out2)
        output3.append(out3)
finish = time.time()
# these kinds of format strings are only available on Python 3.6:
# time to upgrade!
print(f'original inputs: {repr(output1)}')
print(f'total time to execute {sum(output2)} = sum({repr(output2)})')
print(f'time saved by parallelizing: {sum(output2) - (finish-start)}')
print(f'returned in order given: {repr(output3)}')

def calc_stuff(parameter=None):   # these are examples.
    sleep_time = random.choice([0, 1, 2, 3, 4, 5])
    time.sleep(sleep_time)
    return parameter / 2, sleep_time, parameter * parameter

def procedure(j):               # just factoring out the
    parameter = j * offset       # procedure
    # call the calculation
    return calc_stuff(parameter=parameter)
```

**#2 - 2018-12-25 12:40 AM - Jürgen Fischer**

*- Description updated*