

Forum:[Hack Forums](#)

Thread: [Optimizing For Loops](#)

n00b.ca - April 2, 2020, 9:31 am

Hello, I have the following code snippet

Code

```
import numpy as np

init = 0
end = 30
st = 1
end = 1024
dictVals = {'Name':[], 'data': []}

for i in range(0, 70):
    for k in range(0, 1024):
        for j in range(init, end):
            dj=j - st + 1
            if j != st:
                if (np.abs(x[j][k]) > th) and (np.abs(x[j - 1][k]) > th):
                    if (np.dot(x[j][k],x[j - 1][k])) < 0:
                        dictVals ['data'][5][k][i]=dictVals ['data'][5][k][i] + 1

            if j < (en - 1):

                if (np.abs(dx[dj+1]) > th) and (np.abs(dx[dj]) > th):
                    if (np.dot(dx[dj],dx[dj + 1])) < 0:
                        dictVals ['data'][6][k][i]=dictVals ['data'][6][k][i] + 1
```

It currently uses for loops but it takes a bloody long time to run due to the numpy operations it has to perform for each iteration. How would I go about optimizing this code snippet?

I know best practices would have me target the most inner loop and employ some form of list comprehension, lambda function or something but I am unsure of implementation. Any help would be appreciated!

HFDone - April 2, 2020, 10:14 am

threading can solve your problem

ihatevim - April 3, 2020, 12:42 pm

(Apr 2, 2020 10:14 AM)HFDone Wrote: [\[CITE\]](#) threading can solve your problem

Absolutely not. I'll edit this post after I finish reading it. Multi-threading is inherently impossible in python because of GIL. At best you will run in parallel for some IO tasks (which can you can make use of to sort of bypass this restriction in a very limited way- more on that below).

The module you're looking for is asyncio. You will be able to run several code segments simultaneously by taking advantage of a library attempting to bypass the restriction of GIL by passing everything through IO tasks. There are several problems with asyncio like its lack of threadsafe objects, but it should be fine for your purposes.

Regarding actual streamline changes to the code, you could implement lambda like you said by creating a new function with precisely what you need and calling it from within the for loop storing the result in a global (or returning one locally to the main function).

If you're confused with the exact implementation of lambda, you can post your code here and I'll help identify any problems and direct you on how to properly use it. Otherwise, I'm not sure what you're trying to find the dot product/absolute values of. Is this a school problem set?

n00b.ca - April 4, 2020, 12:39 pm

(Apr 3, 2020 12:42 PM)ihatevim Wrote: [\[CITE\]](#) Absolutely not. I'll edit this post after I finish reading it. Multi-threading is inherently impossible in python because of GIL. At best you will run in parallel for some IO tasks (which can you can make use of to sort of bypass this restriction in a very limited way- more on that below).

The module you're looking for is asyncio. You will be able to run several code segments simultaneously by taking advantage of a library attempting to bypass the restriction of GIL by passing everything through IO tasks. There are several problems with asyncio like its lack of threadsafe objects, but it should be fine for your purposes.

Regarding actual streamline changes to the code, you could implement lambda like you said by creating a new function with precisely what you need and calling it from within the for loop storing the result in a global (or returning one locally to the main function).

If you're confused with the exact implementation of lambda, you can post your code here and I'll help identify any problems and direct you on how to properly use it. Otherwise, I'm not sure what you're trying to find the dot product/absolute values of. Is this a school problem set?

Thanks for your reply!

My thoughts exactly, python's GIL will prevent a multithreaded approach.

I can play around with asyncio and see how that works as well as implementing a lambda function. Thanks for the suggestions!