User

I have following C code which takes arguments from python through xlwings UDF - 2 arguments: number of threads and number of random numbers to be generated:

#include <pthread.h>
#include <immintrin.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

unsigned long long \*numbers;

\_\_declspec(dllexport) int rdrand64\_step(unsigned long long \*rand)
{
return \_rdrand64\_step(rand);
}

\_\_declspec(dllexport) void generate\_random\_numbers(int thread\_num, int num\_threads, int num\_numbers)
{
for (int i = 0; i < num\_numbers / num\_threads; i++) {
if (!rdrand64\_step(&numbers[thread\_num \* num\_numbers / num\_threads + i])) {
printf("Failed to generate random number.\n");
}
}
}

\_\_declspec(dllexport) unsigned long long\* get\_numbers()
{
return numbers;
}

\_\_declspec(dllexport) void allocate\_memory(int num\_numbers)
{
numbers = (unsigned long long\*)malloc(num\_numbers \* sizeof(unsigned long long));
}

\_\_declspec(dllexport) void free\_memory()
{
free(numbers);
}

Following is my python script:

@xw.func
def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):
NUM\_THREADS = int(NUM\_THREADS)
NUM\_NUMBERS = int(NUM\_NUMBERS)
C.allocate\_memory(NUM\_NUMBERS)
for i in range(NUM\_THREADS):
C.generate\_random\_numbers(i)
numbers\_ptr = C.get\_numbers()
numbers = [numbers\_ptr[i] for i in range(NUM\_NUMBERS)]
data = [[number] for number in numbers]
C.free\_memory()
return data

Bing

I see that you're trying to generate random numbers using C and Python. However, there seems to be an issue with your Python script. In the generate\_and\_get\_data function, you're calling C.generate\_random\_numbers(i) but this function in your C code expects three arguments: thread\_num, num\_threads, and num\_numbers. You should modify your Python script to pass these arguments correctly. Here's how you can do it:

@xw.func

def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

for i in range(NUM\_THREADS):

C.generate\_random\_numbers(i, NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

numbers = [numbers\_ptr[i] for i in range(NUM\_NUMBERS)]

data = [[number] for number in numbers]

C.free\_memory()

return data

This should resolve the issue. Let me know if you need further assistance! 😊

User

Why my C code expects three arguments? I want to supply number of threads (number of cores in cpu) and number of random numbers to be generated. Can you modify both C and python codes?

Bing

Sure, I can help with that. Here's how you can modify your C code and Python script:

C code:

#include <pthread.h>

#include <immintrin.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

unsigned long long \*numbers;

\_\_declspec(dllexport) int rdrand64\_step(unsigned long long \*rand)

{

return \_rdrand64\_step(rand);

}

\_\_declspec(dllexport) void generate\_random\_numbers(int num\_threads, int num\_numbers)

{

for (int thread\_num = 0; thread\_num < num\_threads; thread\_num++) {

for (int i = 0; i < num\_numbers / num\_threads; i++) {

if (!rdrand64\_step(&numbers[thread\_num \* num\_numbers / num\_threads + i])) {

printf("Failed to generate random number.\n");

}

}

}

}

\_\_declspec(dllexport) unsigned long long\* get\_numbers()

{

return numbers;

}

\_\_declspec(dllexport) void allocate\_memory(int num\_numbers)

{

numbers = (unsigned long long\*)malloc(num\_numbers \* sizeof(unsigned long long));

}

\_\_declspec(dllexport) void free\_memory()

{

free(numbers);

}

Python script:

@xw.func

def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

numbers = [numbers\_ptr[i] for i in range(NUM\_NUMBERS)]

data = [[number] for number in numbers]

C.free\_memory()

return data

In the modified C code, the generate\_random\_numbers function now takes two arguments: num\_threads and num\_numbers. It uses a nested loop to generate random numbers for each thread. The Python script has also been updated to pass the correct number of arguments to this function. Let me know if you need further assistance! 😊

User

I changed the codes as above. Xlwings udf shows following error:

"AttributeError: allocate\_memory
raise AttributeError(name)
File ""D:\Programs\Python\Lib\site-packages\cffi\api.py"", line 907, in make\_accessor
make\_accessor(name)
File ""D:\Programs\Python\Lib\site-packages\cffi\api.py"", line 912, in **getattr**
^^^^^^^^^^^^^^^^^
C.allocate\_memory(NUM\_NUMBERS)
File ""D:\OneDrive - 0yt2k\xlwings\_trial.py"", line 106, in generate\_and\_get\_data
^^^^^^^^^^^
ret = func(\*args)
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 525, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

Bing

The error message you're seeing typically occurs when Python's C Foreign Function Interface (CFFI) can't find the function allocate\_memory in your C library. Here are a few things you can check:

1. \*\*Compilation\*\*: Make sure your C code is compiled correctly and the resulting library is accessible to your Python script. The CFFI module in Python will look for the compiled C library in the same directory as your Python script by default.
2. \*\*Library Loading\*\*: In your Python script, you should have a line that loads the C library, something like `C = ffi.dlopen('my\_library')`. Make sure `'my\_library'` correctly points to your compiled C library.
3. \*\*Function Declaration\*\*: In the CFFI setup, you need to declare the functions that you want to use from the C library. This is typically done with `ffi.cdef("""...""")`. Make sure all your exported functions (`allocate\_memory`, `generate\_random\_numbers`, etc.) are declared here.

If everything seems correct and you're still facing issues, it would be helpful to see the complete Python script including the part where you setup CFFI and load the C library. This will provide more context and help identify potential issues. 😊

User

My python code is as under:

import xlwings as xw
import sys
import numpy as np
from cffi import FFI
import os

print(sys.executable)
ffi = FFI()

# Define the functions in the DLL

ffi.cdef("""
int rdrand64\_step(unsigned long long *rand);
void generate\_random\_numbers(int thread\_num);
unsigned long long* get\_numbers();
""")

# Load the DLL

C = ffi.dlopen('D:\OneDrive - 0yt2k\randomclib2.dll')
@xw.func
def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):
NUM\_THREADS = int(NUM\_THREADS)
NUM\_NUMBERS = int(NUM\_NUMBERS)
C.allocate\_memory(NUM\_NUMBERS)
C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)
numbers\_ptr = C.get\_numbers()
numbers = [numbers\_ptr[i] for i in range(NUM\_NUMBERS)]
data = [[number] for number in numbers]
C.free\_memory()
return data

So I dont find:

Library Loading: In your Python script, you should have a line that loads the C library, something like C = ffi.dlopen('my\_library'). Make sure 'my\_library' correctly points to your compiled C library.

Function Declaration: In the CFFI setup, you need to declare the functions that you want to use from the C library. This is typically done with ffi.cdef("""..."""). Make sure all your exported functions (allocate\_memory, generate\_random\_numbers, etc.) are declared here.

From above information can you find where the bug is?

Bing

I see the issue. In your Python script, you're trying to call the allocate\_memory function from your C library, but this function is not declared in your ffi.cdef block. The ffi.cdef block should contain declarations for all the functions you want to use from your C library.

Here's how you can modify your Python script:

import xlwings as xw

import sys

import numpy as np

from cffi import FFI

import os

print(sys.executable)

ffi = FFI()

# Define the functions in the DLL

ffi.cdef("""

int rdrand64\_step(unsigned long long \*rand);

void generate\_random\_numbers(int num\_threads, int num\_numbers);

unsigned long long\* get\_numbers();

void allocate\_memory(int num\_numbers);

void free\_memory();

""")

# Load the DLL

C = ffi.dlopen('D:\\OneDrive - 0yt2k\\randomclib2.dll')

@xw.func

def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

numbers = [numbers\_ptr[i] for i in range(NUM\_NUMBERS)]

data = [[number] for number in numbers]

C.free\_memory()

return data

In this modified script, I've added the declarations for allocate\_memory and free\_memory to the ffi.cdef block. Also, I've updated the declaration for generate\_random\_numbers to reflect the changes we made to your C code earlier. This should resolve the issue. Let me know if you need further assistance! 😊

User

That worked in xlwings! Only problem is random numbers generated are like:
16296116449175500000
11539563932890200000
18148063750840400000
7457940536569330000
16283204939863800000
2235282490615840000
13676417328878700000
8794529370840660000
4831189143765470000
6833041676174070000
12169945662354200000
11989809183111600000
15490439247310300000
17140552675161200000
7367270322701040000
12608260580633100000
4624318091903260000
17735830700577500000
5592688350307040000
16368760738801400000
4352996364629030000
17386880115342900000
3736098308528830000
5539192991493800000
15294945324218700000
8837222503621510000
5862760230223110000
6229687198564160000
10910883625652100000
1589445713952840000
8273879539491970000
15113101661598600000
11081458577959100000
10827757977746200000
11629915900832500000

As you can see there are trailing zeros after all random numbers. Why is that so?

Bing

The numbers you're seeing are not actually trailing with zeros. They are large numbers represented in the decimal system. The function \_rdrand64\_step generates a 64-bit random number, which can be a very large number when converted to decimal.

The maximum value a 64-bit number can have is 2^64 - 1, which is 18,446,744,073,709,551,615. So the numbers you're seeing are within this range and are not trailing with zeros but are indeed large random numbers.

If you want to generate random numbers within a specific range, you would need to implement a function to scale the output of \_rdrand64\_step to your desired range. Let me know if you need help with that! 😊

User

But why none of the numbers has a non-zero digit at the end? I tried to generate 20000 numbers. All have at least one zero as last digit.

Bing

The reason all your numbers end with at least one zero is due to the way the \_rdrand64\_step function generates random numbers. This function uses a hardware-based random number generator to produce 64-bit random numbers. The last few bits of these numbers are often not as random as the others, which can result in the decimal representation of these numbers often ending in zero.

If you need random numbers that don't have this characteristic, you might consider using a different method to generate your random numbers, or apply a transformation to the output of \_rdrand64\_step to achieve your desired characteristics. For example, you could use a modulo operation to limit the range of your random numbers and increase the likelihood of getting non-zero last digits. Let me know if you need help with that! 😊

User

How to do that: For example, you could use a modulo operation to limit the range of your random numbers and increase the likelihood of getting non-zero last digits. Let me know if you need help with that! 😊?

Bing

Sure, you can use the modulo operation to limit the range of your random numbers. The modulo operation finds the remainder after division of one number by another. So, if you want your random numbers to be in the range from 0 to N, you can simply do random\_number % N.

Here's how you can modify your C code:

#include <pthread.h>

#include <immintrin.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

unsigned long long \*numbers;

unsigned long long N = 1000000000000000000; // Change this to your desired maximum number

\_\_declspec(dllexport) int rdrand64\_step(unsigned long long \*rand)

{

return \_rdrand64\_step(rand);

}

\_\_declspec(dllexport) void generate\_random\_numbers(int num\_threads, int num\_numbers)

{

unsigned long long rand;

for (int thread\_num = 0; thread\_num < num\_threads; thread\_num++) {

for (int i = 0; i < num\_numbers / num\_threads; i++) {

if (!rdrand64\_step(&rand)) {

printf("Failed to generate random number.\n");

}

numbers[thread\_num \* num\_numbers / num\_threads + i] = rand % N;

}

}

}

\_\_declspec(dllexport) unsigned long long\* get\_numbers()

{

return numbers;

}

\_\_declspec(dllexport) void allocate\_memory(int num\_numbers)

{

numbers = (unsigned long long\*)malloc(num\_numbers \* sizeof(unsigned long long));

}

\_\_declspec(dllexport) void free\_memory()

{

free(numbers);

}

In this modified C code, I've added a variable N that you can set to your desired maximum number. Then in the generate\_random\_numbers function, I've used the modulo operation (rand % N) to limit the range of the random numbers. This should increase the likelihood of getting non-zero last digits in your random numbers. Let me know if you need further assistance! 😊

User

I don't know but when I compiled following similar code (not as dll but as exe file) then, there were no zeros trailing at any number:

#include <pthread.h>
#include <immintrin.h>
#include <stdint.h>
#include <stdio.h>
#include <time.h>

#define NUM\_THREADS 16
#define NUM\_NUMBERS 100000

int rdrand64\_step(unsigned long long \*rand)
{
return \_rdrand64\_step(rand);
}

void \*generate\_random\_numbers(void \*arg)
{
int thread\_num = \*(int \*)arg;
char filename[20];
sprintf(filename, "rdrand%d.csv", thread\_num);

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Failed to open file.\n");

return NULL;

}

for (int i = 0; i < NUM\_NUMBERS / NUM\_THREADS; i++) {

unsigned long long rand;

if (rdrand64\_step(&rand)) {

fprintf(file, "%llu\n", rand);

} else {

printf("Failed to generate random number.\n");

}

}

fclose(file);

return NULL;

}

int main()
{
time\_t start\_time = time(NULL);

pthread\_t threads[NUM\_THREADS];

int thread\_nums[NUM\_THREADS];

for (int i = 0; i < NUM\_THREADS; i++) {

thread\_nums[i] = i;

pthread\_create(&threads[i], NULL, generate\_random\_numbers, &thread\_nums[i]);

}

for (int i = 0; i < NUM\_THREADS; i++) {

pthread\_join(threads[i], NULL);

}

// Merge files

FILE \*file = fopen("rdrand.csv", "w");

if (file == NULL) {

printf("Failed to open file.\n");

return 1;

}

for (int i = 0; i < NUM\_THREADS; i++) {

char filename[20];

sprintf(filename, "rdrand%d.csv", i);

FILE \*input\_file = fopen(filename, "r");

if (input\_file == NULL) {

printf("Failed to open file.\n");

return 1;

}

char line[256];

while (fgets(line, sizeof(line), input\_file)) {

fprintf(file, "%s", line);

}

fclose(input\_file);

// Delete the temporary file

remove(filename);

}

fclose(file);

time\_t end\_time = time(NULL);

double time\_spent = difftime(end\_time, start\_time);

int minutes = time\_spent / 60;

int seconds = (int)time\_spent % 60;

printf("Time spent: %d minutes and %d seconds\n", minutes, seconds);

return 0;

}
Code is almost similar uses same intel intrinsic function. Can you compare both this code and the one above and find out the reason for trailing zeros?

Bing

The code you provided is indeed similar to the previous one, but there are a few key differences that might explain why you're not seeing trailing zeros in your output:

1. \*\*Output Format\*\*: In the new code, you're writing the random numbers directly to a file with `fprintf(file, "%llu\n", rand);`. This will write the full decimal representation of the number to the file. In contrast, in your Python script, you're returning the numbers as a list of integers, which might be causing Python to format the numbers in scientific notation when they're very large. This could explain why you're seeing trailing zeros in your Python output but not in your file output.
2. \*\*Random Number Generation\*\*: In both codes, you're using the `\_rdrand64\_step` function to generate random numbers. However, in the new code, you're generating the random numbers directly inside the `generate\_random\_numbers` function and writing them to a file. In contrast, in your Python script, you're storing the random numbers in an array and then returning this array to Python. It's possible that this difference in how you're handling the random numbers could be affecting the output.
3. \*\*Multithreading\*\*: The new code uses multithreading with `pthread\_create` and `pthread\_join`, which could potentially affect the sequence of random numbers generated by `\_rdrand64\_step`.

To get a better understanding of what's happening, you could try printing out the random numbers directly in your C code before they're returned to Python. This would allow you to see if the trailing zeros are present before the numbers are returned to Python.

Let me know if this helps or if you have any other questions! 😊

User

Loooking at your first point:

Cam the python code be tweaked to see the same result as csv file generated in my above code?

Bing

Yes, you can modify your Python code to write the random numbers directly to a CSV file, similar to your C code. Here's how you can do it:

import xlwings as xw

import sys

import numpy as np

from cffi import FFI

import os

import csv

print(sys.executable)

ffi = FFI()

# Define the functions in the DLL

ffi.cdef("""

int rdrand64\_step(unsigned long long \*rand);

void generate\_random\_numbers(int num\_threads, int num\_numbers);

unsigned long long\* get\_numbers();

void allocate\_memory(int num\_numbers);

void free\_memory();

""")

# Load the DLL

C = ffi.dlopen('D:\\OneDrive - 0yt2k\\randomclib2.dll')

@xw.func

def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

C.free\_memory()

In this modified script, I've added a block of code that opens a new CSV file and writes each random number to a new row in the file. This should give you output similar to your C code. Let me know if you need further assistance! 😊

User

When I tried above xw function neither did it returned anything nor it created csv file.

Bing

I'm sorry to hear that the function didn't work as expected. Let's try to debug this issue. Here are a few things you can check:

1. \*\*File Permissions\*\*: Make sure your Python script has the necessary permissions to write files to the directory where it's running.
2. \*\*Error Handling\*\*: Add some error handling in your Python script to catch and print out any exceptions that might be occurring. This can help identify if there's an issue with the file writing or with the random number generation.
3. \*\*Return Statement\*\*: The `generate\_and\_get\_data` function currently doesn't return anything. If you're expecting it to return a value in Excel, you should add a return statement.

Here's how you can modify your Python script:

import xlwings as xw

import sys

import numpy as np

from cffi import FFI

import os

import csv

print(sys.executable)

ffi = FFI()

# Define the functions in the DLL

ffi.cdef("""

int rdrand64\_step(unsigned long long \*rand);

void generate\_random\_numbers(int num\_threads, int num\_numbers);

unsigned long long\* get\_numbers();

void allocate\_memory(int num\_numbers);

void free\_memory();

""")

# Load the DLL

C = ffi.dlopen('D:\\OneDrive - 0yt2k\\randomclib2.dll')

@xw.func

def generate\_and\_get\_data(NUM\_THREADS, NUM\_NUMBERS):

try:

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

C.free\_memory()

return "Random numbers generated and written to 'random\_numbers.csv'"

except Exception as e:

return str(e)

In this modified script, I've added a try-except block to catch any exceptions that might be occurring. If an exception occurs, the function will return the exception message, which should help us identify the issue. Let me know if this helps or if you have any other questions! 😊

User

Traceback (most recent call last):
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke*
return self.*invoke*(dispid, lcid, wFlags, args)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke*
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex*
return func(\*args)
^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 231, in Call
return ToVariant(getattr(obj, method)(\*pargs, \*\*kwargs))
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 726, in import\_udfs
module = get\_udf\_module(module\_name, xl\_workbook)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 397, in get\_udf\_module
module = import\_module(module\_name)
^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\importlib\_*init*\_.py", line 90, in import\_module
return \_bootstrap.\_gcd\_import(name[level:], package, level)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "<frozen importlib.\_bootstrap>", line 1381, in \_gcd\_import
File "<frozen importlib.\_bootstrap>", line 1354, in \_find\_and\_load
File "<frozen importlib.\_bootstrap>", line 1325, in \_find\_and\_load\_unlocked
File "<frozen importlib.\_bootstrap>", line 929, in \_load\_unlocked
File "<frozen importlib.\_bootstrap\_external>", line 990, in exec\_module
File "<frozen importlib.\_bootstrap\_external>", line 1128, in get\_code
File "<frozen importlib.\_bootstrap\_external>", line 1058, in source\_to\_code
File "<frozen importlib.\_bootstrap>", line 488, in \_call\_with\_frames\_removed
File "D:\OneDrive - 0yt2k\xlwings\_trial.py", line 132
except Exception as e:
^^^^^^
SyntaxError: invalid syntax

Bing

The error message indicates a syntax error in your Python script. The error is occurring at the line with except Exception as e:. This usually means there's a problem with the code just before this line.

Looking at the code you provided earlier, it seems like the syntax is correct. However, the error message might be pointing to a different part of your script that wasn't included in your message.

Here are a few things you can check:

1. \*\*Indentation\*\*: Python uses indentation to determine the grouping of statements. Make sure your `try` and `except` blocks are properly indented.
2. \*\*Missing Parentheses or Colons\*\*: Check if you're missing any parentheses `()` or colons `:` in your code. These are common causes of syntax errors.
3. \*\*Code Above the Except Block\*\*: The issue might be with the code just above the `except` block. Check if there's any syntax error there.

If you're still having trouble, please provide the complete Python script or at least a few lines above the except block so I can help you better. 😊

User

@xw.func
def generate\_and\_get\_data\_2(NUM\_THREADS, NUM\_NUMBERS):
NUM\_THREADS = int(NUM\_THREADS)
NUM\_NUMBERS = int(NUM\_NUMBERS)
C.allocate\_memory(NUM\_NUMBERS)
C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)
numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

C.free\_memory()

return "Random numbers generated and written to 'random\_numbers.csv'"

except Exception as e:

return str(e)

Bing

I see the issue. The except block is outside the function definition. It should be inside the function and aligned with the try statement. Here's the corrected code:

@xw.func

def generate\_and\_get\_data\_2(NUM\_THREADS, NUM\_NUMBERS):

try:

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

C.free\_memory()

return "Random numbers generated and written to 'random\_numbers.csv'"

except Exception as e:

return str(e)

In Python, indentation is used to determine the grouping of statements. The try and except blocks should have the same level of indentation, and all the statements inside these blocks should be indented one level further. Let me know if this helps or if you have any other questions! 😊

User

I saw message random numbers generated but I don't find csv file at location: D:\OneDrive - 0yt2k where this xlwings code was run.

Bing

I'm sorry to hear that you're having trouble finding the CSV file. The file should be created in the same directory where your Python script is running. However, if you're running the script from an IDE or a different environment, the working directory might be different.

You can specify an absolute path to make sure the file is created in the desired location. Here's how you can modify your Python script:

# Write numbers to CSV file

with open('D:\\OneDrive - 0yt2k\\random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

In this modified script, I've replaced 'random\_numbers.csv' with the absolute path 'D:\\OneDrive - 0yt2k\\random\_numbers.csv'. This should create the random\_numbers.csv file in the D:\\OneDrive - 0yt2k\\ directory.

Please replace 'D:\\OneDrive - 0yt2k\\random\_numbers.csv' with your desired path and filename. Let me know if this helps or if you have any other questions! 😊

User

Finaly as I saw, there were no trailing zeros in csv file. That means C library is functioning as expected and python script is also correct. But then why in excel do I see zeros at the end of all numbers?

201462784320779000
825021486178699000
989829187380098000
441897735187563000
222008384039722000
595144271714153000
298160303536705000
155233890735582000
40444945714757400
548268316037679000
748474064864679000
515037968350364000
188372801191978000
420918869459495000
That means excel does not allow numbers above certain value?

Bing

Yes, you're correct. Excel has a limit on the precision of numbers. According to Microsoft's documentation, Excel can store numbers with up to 15 significant digits of precision. Any digits beyond this limit are rounded off or replaced with zeros.

This is why you're seeing trailing zeros in Excel for large numbers generated by your C library. The numbers are too large for Excel to represent accurately, so it rounds them off to the nearest number it can represent, which results in trailing zeros.

If you need to work with such large numbers in Excel without losing precision, you might consider storing them as text instead of numbers. This would preserve all the digits but would also mean you can't perform numerical operations on them directly in Excel. Let me know if you need further assistance! 😊

User

Can you change the python code to do that so that I can see all digits? Alternatively can C library code needs to be changed?

Bing

Sure, you can modify your Python script to convert the numbers to strings before returning them to Excel. This will preserve all the digits in Excel. Here's how you can do it:

@xw.func

def generate\_and\_get\_data\_2(NUM\_THREADS, NUM\_NUMBERS):

try:

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('D:\\OneDrive - 0yt2k\\random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

# Convert numbers to strings

numbers = [str(numbers\_ptr[i]) for i in range(NUM\_NUMBERS)]

C.free\_memory()

return numbers

except Exception as e:

return str(e)

In this modified script, I've added a line to convert the numbers to strings before they're returned to Excel: numbers = [str(numbers\_ptr[i]) for i in range(NUM\_NUMBERS)]. This should allow you to see all the digits in Excel. Let me know if this helps or if you have any other questions! 😊

User

Problem is, that creates horizontal stack instead of vertical one columner.

Bing

I see, you want the numbers to be returned as a column in Excel. You can achieve this by making each number a single-item list. Here's how you can modify your Python script:

@xw.func

def generate\_and\_get\_data\_2(NUM\_THREADS, NUM\_NUMBERS):

try:

NUM\_THREADS = int(NUM\_THREADS)

NUM\_NUMBERS = int(NUM\_NUMBERS)

C.allocate\_memory(NUM\_NUMBERS)

C.generate\_random\_numbers(NUM\_THREADS, NUM\_NUMBERS)

numbers\_ptr = C.get\_numbers()

# Write numbers to CSV file

with open('D:\\OneDrive - 0yt2k\\random\_numbers.csv', 'w', newline='') as file:

writer = csv.writer(file)

for i in range(NUM\_NUMBERS):

writer.writerow([numbers\_ptr[i]])

# Convert numbers to strings and make each number a single-item list

numbers = [[str(numbers\_ptr[i])] for i in range(NUM\_NUMBERS)]

C.free\_memory()

return numbers

except Exception as e:

return str(e)

In this modified script, I've changed the line that converts the numbers to strings to also make each number a single-item list: numbers = [[str(numbers\_ptr[i])] for i in range(NUM\_NUMBERS)]. This should return the numbers as a column in Excel. Let me know if this helps or if you have any other questions! 😊

User

This is final conversaion. Just commenting because there were previous cnversation on this same subject

Bing

Sure, I understand. If you have any more questions or need further assistance in the future, feel free to ask. Have a great day! 😊