- Error message in the debug window during script running : " **Memory Block Error: Invalid size** "

Solution:

- maybe the problem of the " Memory Block Error: Invalid size " is in a too big number of names loaded in an internal dictionary at each startup from the Python task (in this case the names in all the .py files used in your Python project) . Please read the Memory Limits paragraph in the easy script user guide

- Another cause could be that one or more pyo files are bigger than 16KB

- You could try to use some of the advices/tips in the following to optimize your code and decide about the best strategy.
  1- re-use a same variable name in different .py files
  2-consider that no space is allocated in the names list for the strings delimited by '' or "" AND ending with \r. In this case you could try to use a method similar to the method removeCRfrom(string) in the file testStrArg.py enclosed, to use strings and save space in the dictionary.
  Pay attention that all the strings with the same name in all py files of your project (e.g. 'same_name') have to be transformed in name +\r (e.g. 'same_name\r'), otherwise you will not have any benefit, and check if the not running version becomes a running version.
  3- reduce the dimension of the compiled file .pyo <16 KB , by splitting the source file in more files and check the related pyo dimensions

- to debug the names loaded at the Python start-up you can use the rtd tool and activate the tasks:
  - **PYTraceDetails** , to trace the variables
  - **PYTraceDebug**, to trace the Python instructions
  - **(SYTraceMem** to trace the RAM . To read the status of the RAM read SAFEHeapPool) OLD
  - **BASTraceMemory** to trace the RAM available. To read the status search for sentences like
    BAS reporting available memory Safe data: 1046528/1048576 (free memory/total memory)
  This method is effective if the script doesn't stop at the start-up

- You could consider a script useful to monitor the names: **dict_debug.py**:

sys.getinterned() is an internal method which permits to list the names allocated in the dictionary

Using MemCheck method you could find how many locations are free (– when the dictionary is not full yet : MemCheck() in the script generates strings to increase the dictionary until the exception occurrs)
Then you need to comment the call of the method when you need to debug your script

```
Current I=0
Current I=20
Memory Block Error: Invalid size
MemCheck:LAST=34;E01=[MemoryError,None]
```

For this example free names are 34 plus/minus 2

- Other tricks:

```
print 'KAMEN0 %d KOKO'% 1
print 'KAMEN1 %d KOKO'% 1
print 'KAMEN2 %d KOKO'% 1
print 'KAMEN3 %d KOKO'% 1
print 'KAMEN4 %d KOKO'% 1
```

doesn't eat memory

```
RR = lambda S:S[:-1]
...
if S == RR('COMMAND_1\r'):pass
elif S == RR('COMMAND_2\r'):pass
```

```
elif S == RR('COMMAND_3\r'):pass
```

doesn't eat memory