Euler 0008

The Problem:

In this giant 1000 digit number the greatest product of 4 adjacent numbers is 9*9*8*9 = 5832
We need to find the greatest 10 adjacent numbers

7316717653133062491922511967442657474235534919493496983520312774506326239578318016984801869478851843858615607891129494954595017379583319528532088055111254069874715852386305071569329096329522744304355712540698747158523863050715693290963295227443043557668966489504452445231617318564030987111217223831136222989342338030813533627661428280644448664523874930358907296290491560440772390713810515859307960866701724271218839987979087922749219016997208880937766572733300105336788122023542180975125454059475224352584907711670556013604839586446706324415722155397536978179778461740649551492908625693219784686224828397224137565705605749026140797296865241453510047482166370484403199890008895243450658541227588666881164271714799244429282308634656748139191231628245861786645835912456652947654568284891288314260769004224190226710556263211111093705442175069416589604080719840385096245544436298123098787992724428490918884580156166097919133875499200524063689912560717606058861164671094050775410022569831552000559357297257163626956188267042825248360082325753042075296345
The Approach:

We can just iterate through... 1 thousand (minus 10) calculations is ultimately not much for the computer.

To optimize a little better though, we can make assumptions with the window. When we move the window across, if the number going out is larger than the number going in then we don't need to run a calculation... it is going to be smaller than our current max.
Otherwise we will run the calculation.

The Code:

```
import math

large_number =
"731671765313306249192251196744265747423553491949349698352031277450632623957831801698480186947885184385861560789112949495459501737958331952853208805511125406987471585238630507156932909632952274430435576689664895044524452316173185640309871112172238311362229893423380308135336276614282806444486645238749303589072962904915604407723907138105158593079608667017242712188399879790879227492190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048395864467063244157221553975369781797784617406495514929086256932197846862248283972241375657056057490261407972968652414535100474821663704844031998900088952434506585412275886668811642717147992444292823086346567481391912316282458617866458359124566529476545682848912883142607690042242190226710556263211111093705442175069416589604080719840385096245544436298123098787992724428490918884580156166097919133875499200524063689912560717606058861164671094050775410022569831552000559357297257163626956188267042825248360082325753042075296345 0"
window_size = 13
best_index = 0
best_index_value = math.prod([int(x) for x in large_number[0:window_size]])

for i in range(1,len(large_number) - window_size):
    if int(large_number[i-1]) < int(large_number[i+window_size-1]):
        current_value = math.prod([int(x) for x in large_number[i:i+window_size]])
        if current_value > best_index_value:
            best_index_value = current_value
            best_index = i
            print("newly found at ", best_index, best_index_value)

print(large_number[best_index:best_index+window_size])
```

Revision #1
Created 2025-07-17 14:42:19 UTC by Maxwell
Updated 2025-07-17 14:44:23 UTC by Maxwell