



Try to preprocess

Use array/vector

Use list comprehension

import numpy as nx

import timeit

N = 10000

diff\_nx = nx.zeros(N, dtype=object)

diff\_py = list(diff\_nx)

start = 1005

def f0():

orig = [start]

for x in diff\_py:

orig.append(orig[-1] + x)

def f1():

diff\_nx[0] = start

nx.add.accumulate(diff\_nx)

t = timeit.Timer("f0()", "from \_\_main\_\_ import f0, f1, diff\_nx, diff\_py, nx, start")

print t.timeit(number=1000)

t = timeit.Timer("f1()", "from \_\_main\_\_ import f0, f1, diff\_nx, diff\_py, nx, start")

print t.timeit(number=1000)

def diff2abs( diffs, start ):

yield start

for diff in diffs:

start += diff

yield start

start = 1005

diffs = [-1, -1, 1, 2]

original\_list = list( diff2abs( diffs, start ))

You put the if at the end:

[y for y in a if y not in b]

List comprehensions are written in the same order as their nested full-specified counterparts, essentially the above statement translates to:

outputlist = []

for y in a:

if y not in b:

outputlist.append(y)

Your version tried to do this instead:

outputlist = []

if y not in b:

for y in a:

outputlist.append(y)

but a list comprehension must start with at least one outer loop.