```
from turtle import *
import turtle as tur
""" Compare calculation of \ sum_i x_i ^2) with
i going from zero to N -1.
We use (i) for loops and list , (ii) for -loop , (iii ) list
comprehension
and (iv) numpy .
We use floating numbers to avoid using Python 's long int ( which would
be likely to make the timings less representative ).
"""
import time
import numpy

N = 100000
def timeit (f, args ):
    """ Given a function f and a tuple args containing
    the arguments for f, this function calls f(* args ),
    and measures and returns the execution time in
    seconds .
    Return value is tuple : entry 0 is the time ,
    entry 1 is the return value of f."""
    starttime = time . time ()
    y = f(* args ) # use tuple args as input arguments
    endtime = time . time ()
    return endtime - starttime , y
def forloop1 (N):
    s = 0
    for i in range (N):
        s += float (i) * float (i)
    return s

def forloop2 (N):
    y = [0] * N
    for i in range (N):
        y[i] = float (i) ** 2
    return sum(y)
def listcomp (N):
    return sum ([ float (x) * x for x in range (N)])
def numpy_ (N):
    return numpy . sum ( numpy . arange (0, N, dtype ='d') ** 2)
# main program starts
print ("N =", N)
forloop1_time , f1_res = timeit ( forloop1 , (N ,))
print ("for - loop1 ", forloop1_time)
forloop2_time , f2_res = timeit ( forloop2 , (N ,))
print ("for - loop2 ", forloop2_time)
listcomp_time , lc_res = timeit ( listcomp , (N ,))
print (" listcomp ", listcomp_time)
numpy_time , n_res = timeit (numpy_ , (N ,))
print (" numpy ", numpy_time)
# ensure that different methods provide identical results
assert f1_res == f2_res
assert f1_res == lc_res

def position(event):
    a, b = event.x, event.y
    print('{}, {}'.format(a, b))
```

```
ws = tur.getcanvas()
ws.bind('<Motion>', position)
tur.done()
```