Problem1 : No loops!

This is a notebook that creates $N = 1M$ randomly-distributed vectors of five numbers $\vec{x}_i$. WITHOUT using loops of any kind, compute the weighted average of each of the five numbers with weights $\vec{w} = (0.05, 0.15, 0.30, 0.10, 0.40)$. That is, each vector has elements, and you should compute

$$\frac{1}{N}\sum_{i=0}^{N-1} \vec{x}_i \cdot \vec{w}$$

You are allowed and encouraged to use any numpy function, including [average (http://lagrange.univ-lyon1.fr/docs/numpy/1.11.0/reference/generated/numpy.average.html)](http://lagrange.univ-lyon1.fr/docs/numpy/1.11.0/reference/generated/numpy.average.html) and [mean (http://lagrange.univ-lyon1.fr/docs/numpy/1.11.0/reference/generated/numpy.mean.html)](http://lagrange.univ-lyon1.fr/docs/numpy/1.11.0/reference/generated/numpy.mean.html).

In [1]:

```
import numpy as np
```

Given: 10M Generated vectors

In [2]:

```
x = np.random.normal( size=(1000000,5), loc=[20,30,20,50,30], sc
ale=[5, 5, 5, 5, 5] )
```

In [3]:

```
print("Shape of data: ", x.shape )
print("First vector:  ", x[0])
```

```
Shape of data:  (1000000, 5)
First vector:   [23.5358963  26.80611785 18.45409146
56.65565576 19.53469334]
```

Part a:

Compute

$$\frac{1}{N}\sum_{i=0}^{N-1}\vec{x}_i \cdot \vec{w}$$

Your code goes here

In [4]:

```
w = [0.05,0.15,0.30,0.10,0.40]
wx = np.average(x, weights=w, axis=1)
```

Print the shape of the weighted vectors, and the weighted average

In [5]:

```
print("Shape of weighted data:", wx.shape )
print("Weighted average : %6.2f" % np.mean(wx) )
```

```
Shape of weighted data: (1000000,)
Weighted average :  28.50
```

Part b:

Repeat a, but this time ONLY include the vector in the sum if the third element is larger than the second.

Here we can use fancy indexing to select only vectors where the third element (index 2) is larger than the second element (index 1).

Your code goes here

In [6]:

```
x2 = x[ x[:,1] < x[:,2] ]
wx2 = np.average(x2, weights=w, axis=1)
```

Print the shape of the vectors that satisfy the required criteria, and the weighted average

In [7]:

```
print("Shape of weighted data 2:", wx2.shape )
print("Weighted average 2: %6.2f" % np.mean(wx2) )
```

```
Shape of weighted data 2: (78631,)
Weighted average 2:  29.49
```

In [ ]:

```

```

Problem2 : Projectile motion

You will create your first simulation! (Try to restrain the joy).

The equation for a projectile is

$$\vec{x}(t) = \frac{1}{2}\vec{a}_0 t^2 + \vec{v}_0 t + \vec{x}_0$$

Simulate this situation in `numpy`. Assume the motion is in 2 dimensions (because it is), with x being the horizontal and y being the vertical directions.

- The time variable t should vary from $0 \ldots 0.5$ seconds, with a time step of 0.001 s.
- The initial position is $\vec{x}_0 = (0, 1)$.
- The initial velocity is $\vec{v}_0 = (10, 0)$.
- Gravity is $\vec{a}_0 = (0, -9.8)$.

Part a. Compute the values, and plot the x and y trajectories in `matplotlib`. Label your axes! To get full credit, you must implement this without `for` loops in `python`. If you cannot manage without loops, that's good for partial credit (losing 5 points). (15/25)

Part b. Using array programming (i.e. NOT calulating the trivial solution of the quadratic equation analytically), numerically find the nearest value of t and x where the projectile crosses the y axis. (10/25)

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
t = np.arange(0,0.5,0.001)
```

In [3]:

```
x0 = np.array([0,1]).reshape((2,1))
v0 = np.array([10,0]).reshape((2,1))
a0 = np.array([0,-9.8]).reshape((2,1))
```

Remember! These are your shapes, and think about broadcasting:

In [4]:

```
print("x0 : ", x0.shape, " v0 : ", v0.shape, " a0 ", a0.shape, "
, t ", t.shape)
```

```
x0 :  (2, 1)  v0 :  (2, 1)  a0  (2, 1) , t  (500,)
```

Computation

Let's define `t1` and `x1` as the time and x coordinate where the projectile crosses the y-axis.

In [5]:

```
t1 = 0. ## Time where projectile crosses axis. You compute this.
x1 = 0. ## x position where projectile crosses axis. You compute
this.
```

Your code goes here.

Without using loops, this will take you ~5 lines of code.

In [6]:

```
#### YOUR CODE GOES HERE:

x = x0 + v0 * t + 0.5 * a0*t**2
cut = x[1,:] > 0
xmod = x.T[cut].T
t1 = t[cut][-1]
x1 = xmod[0,-1]
```

Part a answer (plot):

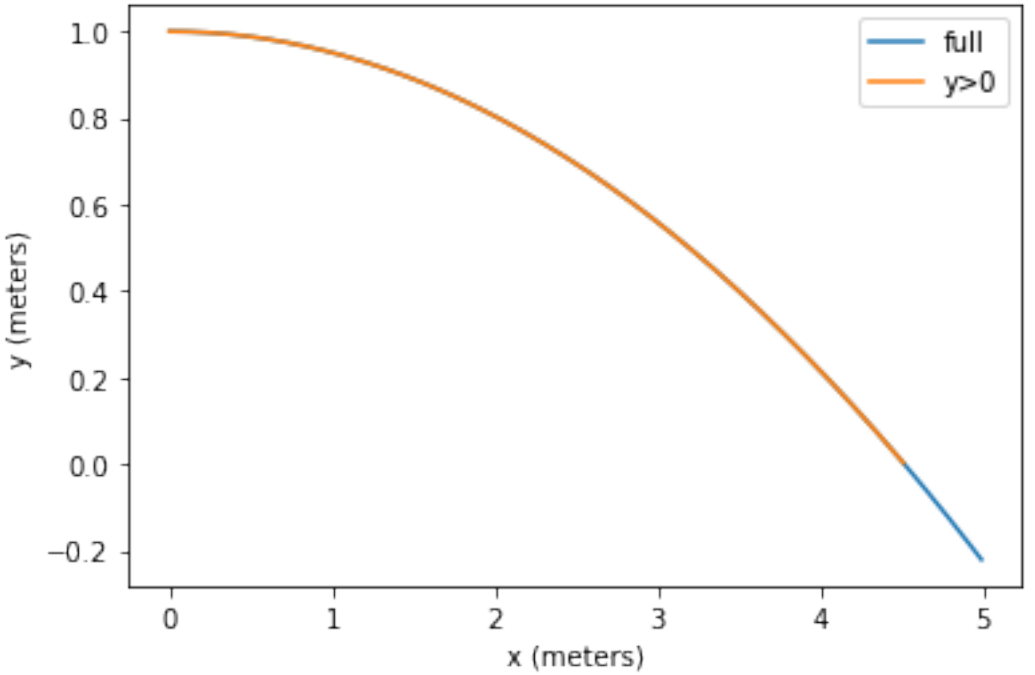
In [7]:

```
#### YOUR CODE GOES HERE:

plt.plot(x[0,:], x[1,:], label='full')
plt.plot(xmod[0,:], xmod[1,:], label='y>0')
plt.xlabel("x (meters)")
plt.ylabel("y (meters)")
plt.legend()
```

Out[7]:

```
<matplotlib.legend.Legend at 0x7f2f30600bd0>
```



Part b answer

(prints the values `t1` and `x1` from above)

In [8]:

```
print ( 'crosses y axis : t = ', t1, 's, x = ', x1, 'meters' )
```

```
crosses y axis : t =  0.451 s, x =  4.51 meters
```

In [ ]:

```

```

Using SWIG with the Point class

Important! The `CompPhys` git directory must be parallel to this one. So it will look something like this:

```
current directory (/results)
    |
    |-- /results/CompPhys
    |
    |-- /results/technical-assignment-5-rappoccio
```

We will reuse code from `CompPhys` .

The Point class

Here, we reuse the Point class that we created in CompPhys. If you want to remind yourself, uncomment the following two commands

In [ ]:

```
#! cat ../CompPhys/ReviewCpp/ClassExample/Point.h
```

In [ ]:

```
#! cat ../CompPhys/ReviewCpp/ClassExample/Point.cc
```

The swig files

Here, we're going to just include the `"Point.h"` file from above with the right syntax. Then we have a `"setup.py"` file to describe how to compile it.

In [ ]:

```
#cat swig_point/point3d.i
```

In [ ]:

```
#cat swig_point/setup3d.py
```

Create the interface with SWIG

Note! We have a function called `"print"`, which is a keyword, so we cannot use that. It will be renamed to `"_print"` with an underscore in the front.

In [1]:

```
! swig -c++ -python -I../ swig_point/point3d.i
```

```
../CompPhys/ReviewCpp/InheritanceExample/Point.h:13:
Warning 314: 'print' is a python keyword, renaming t
o '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point.h:13:
Warning 314: 'print' is a python keyword, renaming t
o '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point3d.h:1
4: Warning 314: 'print' is a python keyword, renamin
g to '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point3d.h:1
4: Warning 314: 'print' is a python keyword, renamin
g to '_print'
```

Here's the compilation of the files

In [2]:

```
! python swig_point/setup3d.py build_ext --inplace
```

```
running build_ext
building '_point3d' extension
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c ../CompPhys/ReviewCp
p/InheritanceExample/Point3d.cc -o build/temp.linux-
x86_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/
Point3d.o -std=c++11 -I../
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c ../CompPhys/ReviewCp
p/InheritanceExample/Point.cc -o build/temp.linux-x8
6_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/Po
int.o -std=c++11 -I../
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c swig_point/point3d_w
rap.cxx -o build/temp.linux-x86_64-3.7/swig_point/po
int3d_wrap.o -std=c++11 -I../
x86_64-linux-gnu-g++ -pthread -shared -Wl,-O1 -Wl,-B
symbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,r
elro -Wl,-Bsymbolic-functions -Wl,-z,relro -g -fstac
k-protector-strong -Wformat -Werror=format-security
-Wdate-time -D_FORTIFY_SOURCE=2 build/temp.linux-x86
_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/Poi
nt3d.o build/temp.linux-x86_64-3.7/../CompPhys/Revie
wCpp/InheritanceExample/Point.o build/temp.linux-x86
_64-3.7/swig_point/point3d_wrap.o -o /results/techni
cal-assignment-5-rappoccio/_point3d.cpython-37m-x86_
64-linux-gnu.so
```

Next setup the path

We need to tell python where to find the libraries we just compiled.

In [3]:

```
import sys
import os

sys.path.append( os.path.abspath("swig_point") )

print (sys.path)
```

```
['/results/technical-assignment-5-rappoccio', '/app/
compphys_python', '/usr/lib/python37.zip', '/usr/lib
/python3.7', '/usr/lib/python3.7/lib-dynload', '', '
/usr/local/lib/python3.7/dist-packages', '/usr/lib/p
ython3/dist-packages', '/usr/local/lib/python3.7/dis
t-packages/IPython/extensions', '/results/.ipython',
'/results/technical-assignment-5-rappoccio/swig_poin
t']
```

Ready!

We have the `Point` class (uppercase "P") in module "point" (lowercase "p").

In [4]:

```
from point3d import Point, Point3d
```

Get help

You can see what was generated by asking for help!

In [ ]:

```

```

In [ ]:

```
help("point3d")
```

Do some math

Here we can use our `operator+` class within python! Woohoo!

In [5]:

```
p1 = Point(1,2)
p2 = Point(2,3)
p3 = p1 + p2
```

In [6]:

```
print(p3.x(), ",", p3.y())
```

```
3.0 , 5.0
```

Problem 3: SWIG for Point3d

Problem 3a

a. You will now repeat the steps above for the `Point3d` class in an interface called `point3d`. You are free to use the `Point3d` class with the inheritance as defined in the `CompPhys/ReviewCpp/InheritanceExample` folder, or write a simpler one.

- Create `swig_point/point3d.i`
- Create `swig_point/setup3d.py`
- Create the interface with the `swig` command.
- Compile with `setup3d.py` similarly to above.
- Import `point3d`, create two 3-d points `v1` and `v2`, set them to `(1,2,3)` and `(4,5,6)`. Set a third point (`v3`) equal to their sum.

In [7]:

```
v1 = Point3d(1,2,3)
v2 = Point3d(4,5,6)
v3 = v1 + v2
print(v3.x(), ",", v3.y(), ", ", v3.z())
```

```
5.0 , 7.0 ,  9.0
```

Problem 3b

b. Write another file called `pointops` that will implement operations on `Point3d`. Define a function to compute the cross product of the two vectors with the following C++ interface:

```
Point3d cross( Point3d const & a, Point3d const & b);
```

- Create `swig_point/cross.i`
- Create `swig_point/setup_cross.py`
- Create the interface with the `swig` command.
- Compile with `cross.py` similarly to above.
- Then take the cross product of 3-d points `v1` and `v2` above.

In [8]:

```
! swig -c++ -python -I../ swig_point/cross.i
```

```
../CompPhys/ReviewCpp/InheritanceExample/Point.h:13:
Warning 314: 'print' is a python keyword, renaming t
o '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point.h:13:
Warning 314: 'print' is a python keyword, renaming t
o '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point3d.h:1
4: Warning 314: 'print' is a python keyword, renamin
g to '_print'
../CompPhys/ReviewCpp/InheritanceExample/Point3d.h:1
4: Warning 314: 'print' is a python keyword, renamin
g to '_print'
```

In [9]:

```
! python swig_point/setup_cross.py build_ext --inplace
```

```
running build_ext
building '_cross' extension
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c ../CompPhys/ReviewCp
p/InheritanceExample/Point3d.cc -o build/temp.linux-
x86_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/
Point3d.o -std=c++11 -I../
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c ../CompPhys/ReviewCp
p/InheritanceExample/Point.cc -o build/temp.linux-x8
6_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/Po
int.o -std=c++11 -I../
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c swig_point/cross.cpp
-o build/temp.linux-x86_64-3.7/swig_point/cross.o -s
td=c++11 -I../
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O
2 -Wall -g -fstack-protector-strong -Wformat -Werror
=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fP
IC -I/usr/include/python3.7m -c swig_point/cross_wra
p.cxx -o build/temp.linux-x86_64-3.7/swig_point/cros
s_wrap.o -std=c++11 -I../
x86_64-linux-gnu-g++ -pthread -shared -Wl,-O1 -Wl,-B
symbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,r
elro -Wl,-Bsymbolic-functions -Wl,-z,relro -g -fstac
k-protector-strong -Wformat -Werror=format-security
-Wdate-time -D_FORTIFY_SOURCE=2 build/temp.linux-x86
_64-3.7/../CompPhys/ReviewCpp/InheritanceExample/Poi
nt3d.o build/temp.linux-x86_64-3.7/../CompPhys/Revie
wCpp/InheritanceExample/Point.o build/temp.linux-x86
_64-3.7/swig_point/cross.o build/temp.linux-x86_64-3
.7/swig_point/cross_wrap.o -o /results/technical-ass
ignment-5-rappoccio/_cross.cpython-37m-x86_64-linux-
gnu.so
```

In [10]:

```
from cross import cross
```

In [11]:

```
v4 = cross(v1,v2)
```

In [12]:

```
print(v4.x(), ", ", v4.y(), ", ", v4.z())
```

```
-3.0 ,  6.0 ,  -3.0
```

In [ ]:

```

```