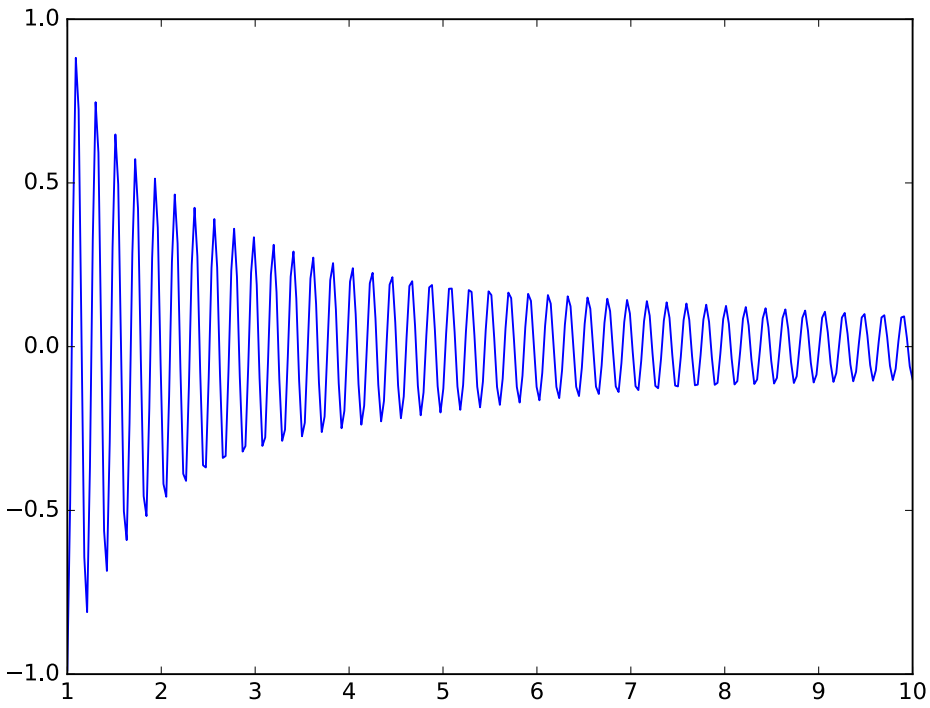
The importance of time sampling with oscillatory phenomena:

Some of you have obtained "surprising" results for project 4 when plotting angular amplitude as function of time: they found long wavelength modulations. This problem happens when you are trying to plot an oscillating function with an insufficient number of points where the number of points per period is close to inverse integer of the period of the function. To make this clear, imagine you want to plot the function `sin(30x)/x`:

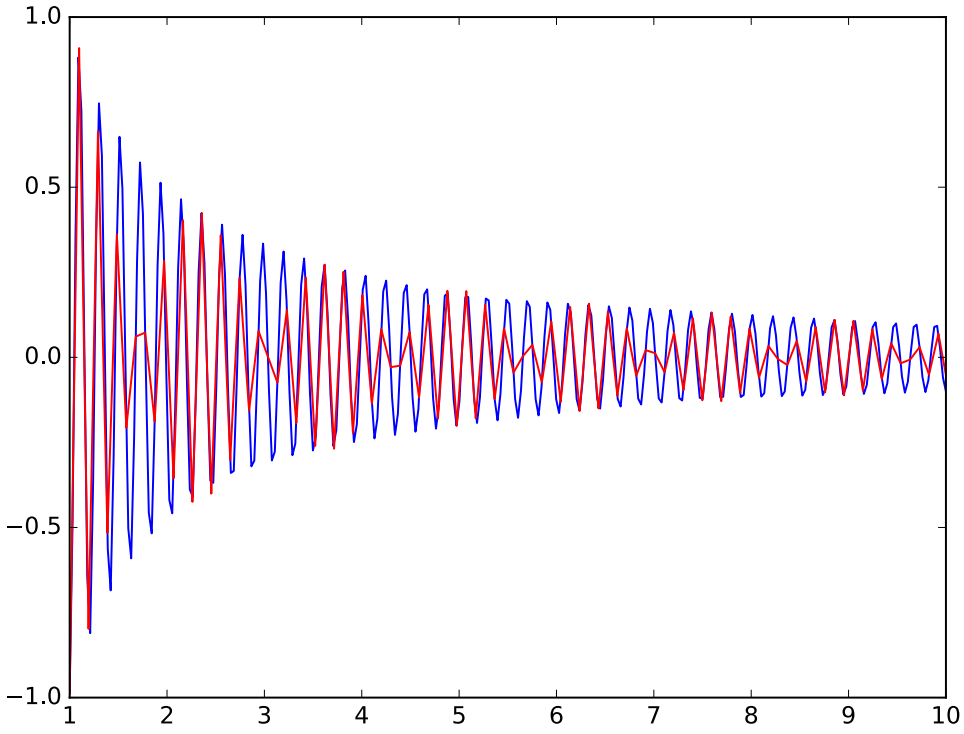
```
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(1,10,300)
plt.plot(x,np.sin(30.*x)/x)
plt.show()
```



The period is $2\pi/30 \sim 0.209$. Now imagine I plot the same function with the following sampling:

```
x2=np.linspace(1,30,300)
plt.plot(x,np.sin(30.*x)/x)
plt.plot(x2,np.sin(30.*x2)/x2,'r')
plt.xlim([1,10])
```

The sampling period in `x2` is `0.1`, that is close to half the function period. You'll get this artificial "beat" (in red)



It is important to be aware of this potential problem when plotting highly oscillating functions. This is also known as "aliasing" problem which occurs when you under-sample a high frequency wave and misinterpret it as a longer wavelength.

Project 5 (Due date Thursday November 24th 12 a.m.)

Back in 1827, the botanist Robert Brown observed that small pollen grains at the surface of water were moving "spontaneously" with there was no apparent cause for this motion. This remained a mystery until A. Einstein in 1905 showed that the cause of this motion was likely the constant random bombardment by water molecules on the pollen grain. This explanation provided the definitive proof of the existence of atoms. A few years later, physicist Jean Perrin performed a series of experiments which confirmed Einstein predictions and Perrin was awarded the Physics Nobel prize in 1926 for this work. This motion was named after its discoverer, the **brownian motion**.
In addition to the major conceptual breakthrough it represented in 1908, the brownian motion had also a fundamental role in the development of some major branches of modern physics: statistical physics and quantum physics. Today, the brownian motion developed in many modern variants of what we call *stochastic phenomena* which have **very** deep ramifications in all areas of science, sociology and art (https://en.wikipedia.org/wiki/Stochastic) and profound connections with the theory of probability in mathematics.

For a quick overview of the brownian motion, read the wikipedia page:
https://en.wikipedia.org/wiki/Brownian_motion

for a more thorough historical and physical description, these notes are very good:
http://userpages.umbc.edu/~dfrey1/ench630/philipse_notes_on_brownian_motion.pdf

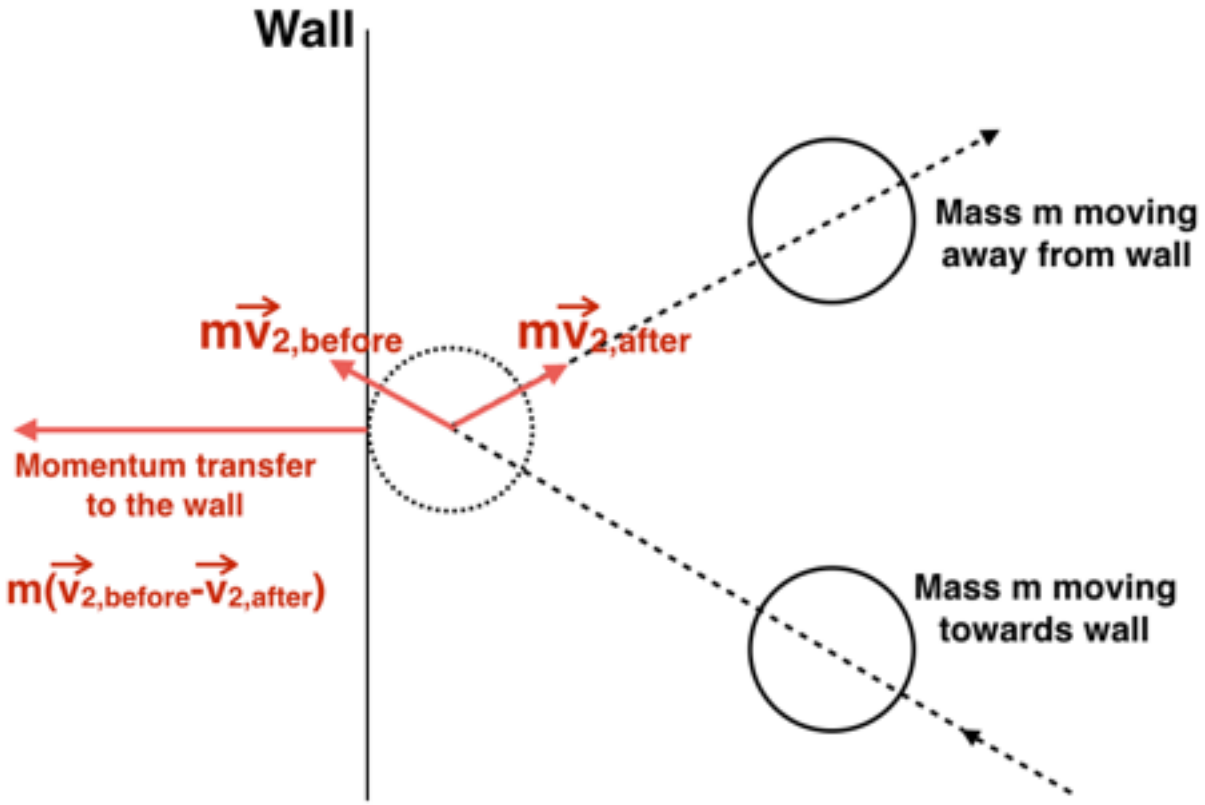
The brownian motion belongs to the family of *diffusion processes* (https://en.wikipedia.org/wiki/Diffusion_process). A particularly interesting property of brownian motion, derived theoretically by A. Einstein, is that the average distance traveled by the grain is proportional to square-root of time, and not proportional to time as one would expect from classical inertial motion. One way of measuring this property is by measuring the average traveled distance squared $\langle r^2 \rangle$ of the grain, and show that it is proportional to time:

$$\langle r^2 \rangle = 2Dt$$

Where *D* is called the diffusion coefficient. In this project, you will simulate an experiment similar to what Robert Brown saw in 1827 and confirm the above prediction on the average traveled distance squared.
The reality of brownian motion of pollen grains is that the grain is a lot more massive than a water molecules, and there is a gigantic number of collisions per second (of the order of 10^{21}). This is beyond the computing capability of our servers, so you will simulate a much smaller number of collisions, and the grain will be much less massive.

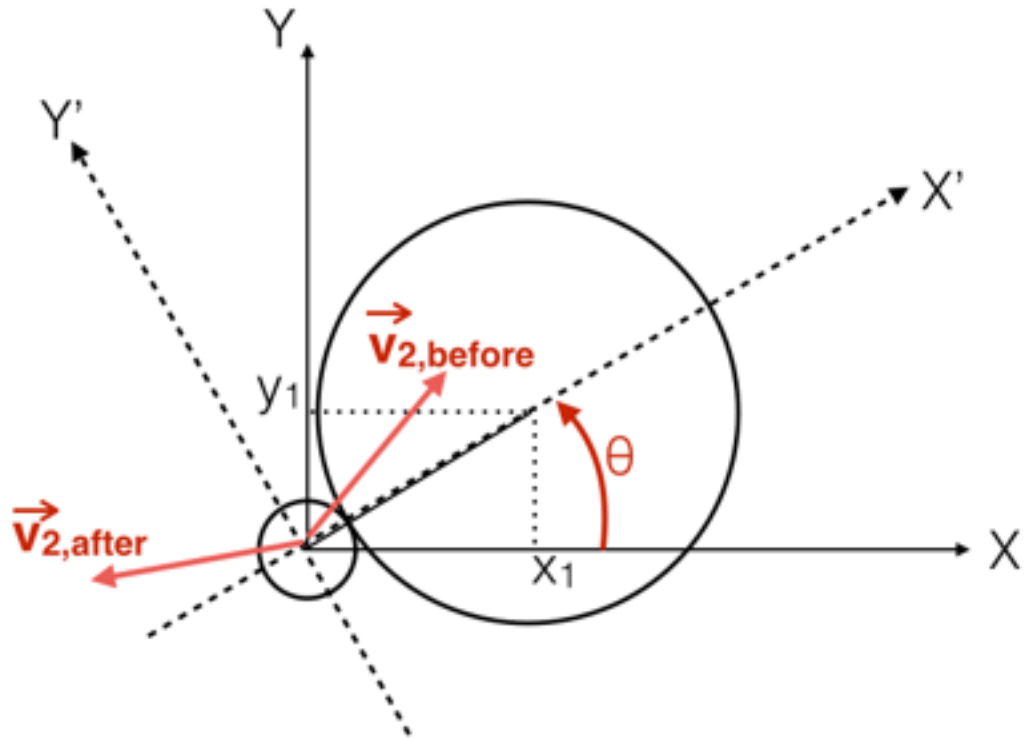
Physically, what is happening is a succession of collisions between the grain and a molecule (let's assume elastic collision where energy and momentum are both conserved).

We will assume that a molecule collision on the grain is similar to a ball (molecule) colliding on a wall (grain) as shown in the figure below. The initial velocity of the molecule is $\mathbf{v_2}=(v_{2x},v_{2y})$. You can see that when colliding with the wall, the component of velocity parallel to the wall (v_{2y}) is unchanged, while the component perpendicular to the wall changes by $-2v_{2x}$. The amount of momentum transferred to the wall is $-2mv_{2x}$, it means the wall gains velocity equal to $-2(m/M)v_{2x}$ where M is the mass of the wall. For a wall, we can take M infinite, so the wall absorbs the momentum and barely moves at all.



For our brownian motion simulation, we replace the wall by a circular grain of mass M (circular because we work in 2 dimension). The grain can move, so at each collision it absorbs momentum $-2mv_{2x}$ but it also gains velocity increment $-2(m/M)v_{2x}$ that is not zero because the grain has a finite mass M.
The difficulty lies in the fact that the grain is circular and not straight like a wall. Therefore there is a local coordinate change to apply in order to calculate the momentum

transfer and hence the velocity change of the grain for each collision, which depends on relative angle where the molecule hit the grain. The geometrical configuration is shown in the next figure.



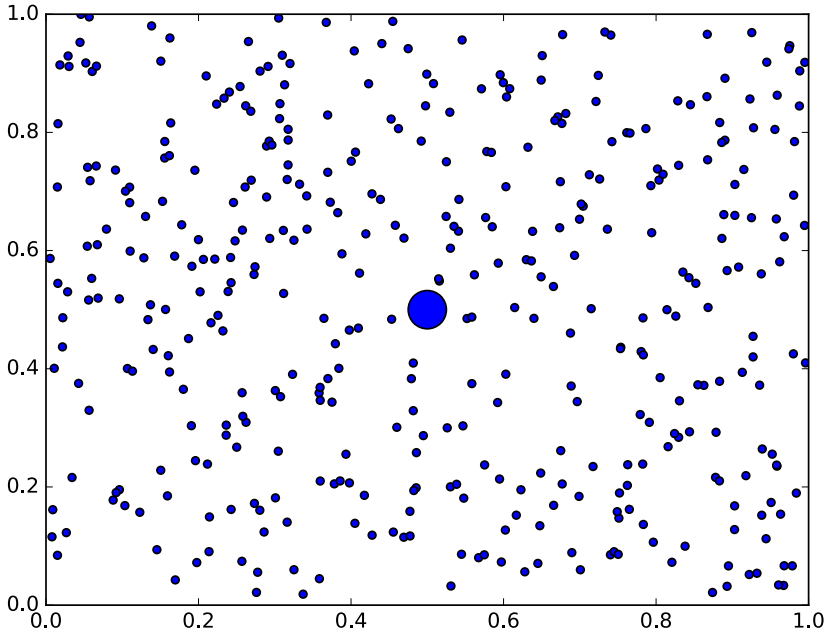
We will use another approximation here: we will assume that the grain velocity change from a collision adds up to the existing velocity of the grain, if it has any. This is physically incorrect as it would result in violating the conservation of momentum, but since `M>>m` and the grain velocity is much smaller than the molecule velocities, this turns out to be a good approximation. Doing the correct calculation would require us to work in the center of mass the pair grain+molecule for each collision and calculate the molecule velocity magnitude change, an unnecessary complication given the experimental conditions.

Objectives: I want you to do two things for this project *1) create an animation of the brownian motion and 2) generate some plots that illustrate the physics of it*.
The simulation should be confined to 2-dimensional box of size one by one, with x coordinate going from 0 to 1 and y coordinate going from 0 to 1. The grain has a radius 0.03 and the molecules are point-like and do not interact with each other. The velocity of the molecules is random, each velocity component of each molecule is a random number

taken from a normal distribution of dispersion one (i.e. use `random.randn()`). The mass of the grain is 20 times the mass of a molecule `M=20m`.

Making the animation:

For the animation, you will use 400 molecules and run over 5000 time steps, with time steps Dt=0.001. The grain starts at the center of the box (i.e. with position 0.5,0.5) and initial zero velocity. It should look like this:



The code that generates the animation should be called **`browniananim.py`**, and your movie called **`browniananim.mp4`**.

Dispersion study:

Building from the file **`browniananim.py`** above, I want you to extend the number of molecules to 4000 (ten times more) and 15000 time steps (three times more). Because this is now going to be a lot more demanding in computing you have to comment out the movie making parts. You will therefore create a new python script called **`brownian-plots.py`** which makes the following plots:

-A plot showing the full trajectory `(x,y)` of the grain over the 15000 time steps. You will call this plot **`brownianpath.pdf`**.

-A second plot showing the distance square from the grain initial position at `(0.5,0.5)` as a function of time, averaged over 20 simulations, each simulation being run with 15000 time steps. You will call this plot **`brownianscatter.pdf`**.

Then, in a text file called **`brownian.txt`**, answer how, in your opinion, the plot in **`brownianscatter.pdf`** agrees with Einstein prediction that dispersion scales with square-root of time. Write your estimate of the diffusion coefficient *D*.

Tips: This is a relatively challenging project where you will have to think how to solve certain issues as they come up when you will start coding the collision between molecules and grain. These issues should come up when you start your first simulation tests! It is important that you succeed doing the animation first, so that the behavior is visually correct, before you more on increasing the number of particles and time steps. To help you get started, here is a code which you can use and build from it which sets the background molecules in motion. There is no grain in here, you have to do this part of coding yourself, and then the collision:

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

npoint=400
nframe=5000
xmin,xmax,ymin,ymax=0,1,0,1
fig, ax = plt.subplots()
plt.xlim(xmin,xmax)
plt.ylim(ymin,ymax)
Dt=0.001

def update_point(num):
    global x,y,vx,vy
    print(num)
    indx=np.where((x < xmin) | (x > xmax))
    indy=np.where((y < ymin) | (y > ymax))
    vx[indx]=-vx[indx]
    vy[indy]=-vy[indy]
    dx=Dt*vx
    dy=Dt*vy
    x=x+dx
    y=y+dy

    data=np.stack((x,y),axis=-1)
    im.set_offsets(data)

x = np.random.random(npoint)
y = np.random.random(npoint)
```

```
vx=np.random.randn(npoint)
vy=np.random.randn(npoint)
im = ax.scatter(x,y)

animation = animation.FuncAnimation(fig, update_point,
nframe,interval=1,repeat=False)
```