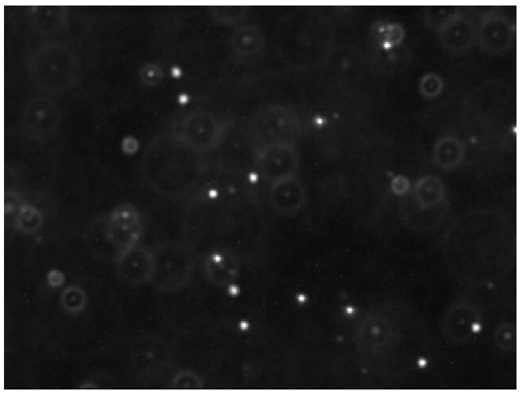
The purpose of this project is to re-affirm the atomic nature of matter by tracking the motion of particles undergoing Brownian motion, fitting this data to Einstein's model, and estimating Avogadro's number.

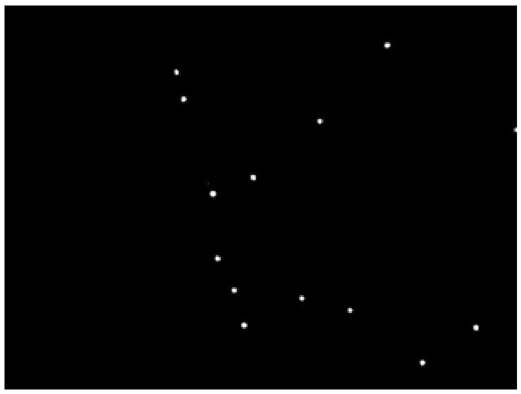
Perspective The atom played a central role in 20th century physics and chemistry, but prior to 1908 the reality of atoms and molecules was not universally accepted. In 1827, the botanist Robert Brown observed the random erratic motion of wildflower pollen grains immersed in water using a microscope. This motion would later become known as *Brownian motion*. Einstein hypothesized that this Brownian motion was the result of millions of tiny water molecules colliding with the larger pollen grain particles.

In one of his "miracle year" (1905) papers, Einstein formulated a quantitative theory of Brownian motion in an attempt to justify the "existence of atoms of definite finite size." His theory provided experimentalists with a method to count molecules with an ordinary microscope by observing their collective effect on a larger immersed particle. In 1908 Jean Baptiste Perrin used the recently invented ultramicroscope to experimentally validate Einstein's kinetic theory of Brownian motion, thereby providing the first direct evidence supporting the atomic nature of matter. His experiment also provided one of the earliest estimates of Avogadro's number. For this work, Perrin won the 1926 Nobel Prize in physics.

The Problem In this project, you will redo a version of Perrin's experiment. Your job is greatly simplified because with modern video and computer technology (in conjunction with your programming skills), it is possible to accurately measure and track the motion of an immersed particle undergoing Brownian motion. We supply video microscopy data of polystyrene spheres ("beads") suspended in water, undergoing Brownian motion. Your task is to write a program to analyze this data, determine how much each bead moves between observations, fit this data to Einstein's model, and estimate Avogadro's number.

Here is a movie (avi, mov) of several beads undergoing Brownian motion. Below is a typical raw image (left) and a cleaned up version (right) using thresholding, as described below.





Each image shows a two-dimensional cross section of a microscope slide. The beads move in and out of the microscope's field of view (the x- and y-directions). Beads also move in the z-direction, so they can move in and out of the microscope's depth of focus; this results in halos, and it can also result in beads completely disappearing from the image.

Problem 1. (*Particle Identification*) The first challenge is to identify the beads amidst the noisy data. Each image is 640-by-480 pixels, and each pixel is represented by a `Color` object which needs to be converted to a luminance value ranging from 0.0 (black) to 255.0 (white). Whiter pixels correspond to beads (foreground) and blacker pixels to water (background). We break the problem into three pieces:

1. *Read the image.* Use the `Picture` data type to read in the image.
2. *Classify the pixels as foreground or background.* We use a simple, but effective, technique known as *thresholding* to separate the pixels into foreground and background components: all pixels with monochrome luminance values strictly below some threshold τ (tau) are considered background, and all others are considered foreground. The two pictures in figure above illustrate the original frame (left) and the same frame after thresholding (right), using $\tau = 180.0$. This value of τ results in an effective cutoff for the supplied data.

3. *Find the blobs.* A polystyrene bead is typically represented by a disc-like shape of at least some minimum number P (typically 25) of connected foreground pixels. A blob or connected component is a maximal set of connected foreground pixels, regardless of its shape or size. We will refer to any blob containing at least P pixels as a bead. The center-of-mass of a blob (or bead) is the average of the x- and y-coordinates of its constituent pixels.

Define a helper data type `Blob` in `blob.py` that has the following API:

method	description
`Blob()`	an empty blob b
`b.add(i, j)`	add a pixel (i, j) to the b
`b.mass()`	the number of pixels in b, ie, its mass
`b.distanceTo(c)`	the distance between the centers of b and c
`str(b)`	string representation of b's mass and center of mass

Next, define a data type `BlobFinder` in `blob_finder.py` that has the following API. Use depth-first search to efficiently identify the blobs.

method	description
`BlobFinder(pic, tau)`	a blob finder bf to find blobs in the picture pic using a luminance threshold tau
`bf.getBeads(P)`	list of all beads with $\geq P$ pixels

Finally, implement a test client `_main()` in `blob_finder` that takes an integer P, a float tau, and the name of a JPEG file as command-line arguments; then creates a `BlobFinder` object using a luminance threshold of tau; prints out all of the beads with at least P pixels; and finally, prints out all of the blobs (beads with at least 1 pixel). Note: For full credit you must write `main()` as if it were a client. That means that you must not access any private methods or private instance variables of the `BlobFinder` data type.

```
$ python blob_finder.py 25 180.0 run_1/frame00001.jpg
13 Beads:
29 (214.7241, 82.8276)
36 (223.6111, 116.6667)
42 (260.2381, 234.8571)
35 (266.0286, 315.7143)
31 (286.5806, 355.4516)
37 (299.0541, 399.1351)
35 (310.5143, 214.6000)
31 (370.9355, 365.4194)
28 (393.5000, 144.2143)
27 (431.2593, 380.4074)
36 (477.8611, 49.3889)
38 (521.7105, 445.8421)
35 (588.5714, 402.1143)

15 Blobs:
29 (214.7241, 82.8276)
36 (223.6111, 116.6667)
1 (254.0000, 223.0000)
42 (260.2381, 234.8571)
35 (266.0286, 315.7143)
31 (286.5806, 355.4516)
37 (299.0541, 399.1351)
35 (310.5143, 214.6000)
31 (370.9355, 365.4194)
28 (393.5000, 144.2143)
27 (431.2593, 380.4074)
36 (477.8611, 49.3889)
38 (521.7105, 445.8421)
35 (588.5714, 402.1143)
13 (638.1538, 155.0000)
```

The program identifies 15 blobs in the sample frame, 13 of which are beads. Our string representation of a blob specifies its mass (number of pixels) and its center of mass (in the 640-by-480 picture). By convention, pixels are measured from left-to-right, and from top-to-bottom (instead of bottom-to-top).

Problem 2. (*Particle Tracking*) The next step is to determine how far a bead moved from one time step t to the next $t+\Delta t$. For our data, $\Delta t = 0.5$ seconds per frame. We assume the data is such that each bead moves a relatively small amount, and that two beads do not collide. However, we must account for the possibility that the bead disappears from the frame, either by departing the microscope's field of view in the x- or y- direction, or moving out of the microscope's depth of focus in the z-direction. Thus, for each bead at time $t + \Delta t$, we calculate the closest bead at time t (in Euclidean distance) and identify these two as the same beads. However, if the distance is too large, ie, greater than Δ (delta) pixels, we assume that one of the beads has either just begun or ended its journey. We record the displacement that each bead travels in the Δt units of time.

Implement a client program `bead_tracker.py` that takes an integer P, a float *tau*, a float *delta*, and a sequence of JPEG filenames as command-line arguments, identifies the beads in each JPEG image using `BlobFinder`, and prints out (one per line, formatted with 4 decimal places to the right of decimal point) the radial distance that each bead moves from one frame to the next (assuming it is no more than *delta*). Note that it is not necessary to explicitly track a bead through a sequence of frames — you only need to worry about identifying the same bead in two consecutive frames.

```
$ python bead_tracker.py 25 180.0 25.0 data/run_1/*.jpg
7.1833
4.7932
2.1693
5.5287
5.4292
4.3962
...
```

Problem 3. (*Data Analysis*) Einstein's theory of Brownian motion connects microscopic properties (eg, radius, diffusivity) of the beads to macroscopic properties (eg, temperature, viscosity) of the fluid in which the beads are immersed. This amazing theory enables us to estimate Avogadro's number with an ordinary microscope by observing the collective effect of millions of water molecules on the beads.

1. *Estimating the self-diffusion constant.* The self-diffusion constant D characterizes the stochastic movement of a molecule (bead) through a homogeneous medium (the water molecules) as a result of random thermal energy. The Einstein-Smoluchowski equation states that the random displacement of a bead in one dimension has a Gaussian distribution with mean zero and variance $\sigma^2 = 2D\Delta t$, where Δt is the time interval between position measurements. That is, a molecule's mean displacement is zero and its mean square displacement is proportional to the elapsed time between measurements, with the constant of proportionality $2D$. We estimate σ^2 by computing the variance of all observed bead displacements in the x and y directions. Let $(\Delta x_1, \Delta y_1), \dots, (\Delta x_n, \Delta y_n)$ be the n bead displacements, and let $r_1, \dots, r_n$ denote the radial displacements. Then

$$\begin{aligned}\sigma^2 &= \frac{(\Delta x_1^2 + \cdots + \Delta x_n^2) + (\Delta y_1^2 + \cdots + \Delta y_n^2)}{2n} \\ &= \frac{r_1^2 + \cdots + r_n^2}{2n}.\end{aligned}$$

   For our data, $\Delta t = 0.5$ so our estimate for σ^2 is an estimate for D as well. Note that the radial displacements in the formula above are measured in meters. The radial displacements output by your `bead_tracker.py` program are measured in pixels. To convert from pixels to meters, multiply by 0.175×10^{-6} (meters per pixel). The value of n is the count of the total number of displacements read.

2. *Estimating the Boltzmann constant.* The Stokes-Einstein relation asserts that the self-diffusion constant D of a spherical particle immersed in a fluid is given by

$$D = \frac{kT}{6\pi\eta\rho},$$

   where, for our data T (absolute temperature) is 297 degrees Kelvin (room temperature), η (viscosity of water) is 9.135×10^{-4} Nsm^{-2} (at room temperature), ρ (radius of bead) is 0.5×10^{-6}, and k is the *Boltzmann constant.* All parameters are given in SI units. The Boltzmann constant is a fundamental physical constant that relates the average kinetic energy of a molecule to its temperature. We estimate k by measuring all of the parameters in the Stokes-Einstein equation, and solving for k.

3. *Estimating Avogadro's number.* Avogadro's number N_A is defined to be the number of particles in a mole. By definition, $k = R/N_A$, where the universal gas constant R is approximately 8.31457 $\text{JK}^{-1}\text{mol}^{-1}$. Use R/k as an estimate of Avogadro's number.

For the final part, implement a client program `avogadro.py` that reads in the displacements from standard input and computes an estimate of Boltzmann's constant and Avogadro's number using the formulae described above.

```
$ python bead_tracker.py 25 180.0 25.0 data/run_1/*.jpg | python avogadro.py
Boltzman = 1.253509e-23
Avogadro = 6.633037e+23
```

Data We provide ten datasets (they are under the `data` directory), obtained by William Ryu (Princeton University) using fluorescent imaging. Each run contains a sequence of two hundred 640-by-480 color JPEG images, `frame00000.jpg` through `frame00199.jpg` and is stored in a subdirectory `run_1` through `run_10`. The directory also contains some reference solutions.

Files to Submit

1. `blob.py`
2. `blob_finder.py`
3. `bead_tracker.py`
4. `avogadro.py`
5. `report.txt`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python run_tests.py [<problems>]
  ```

  where the optional argument `<problems>` lists the numbers of the problems you want to test; all the problems are tested if no argument is given.

- Make sure your programs meet the style requirements by running the following command on the terminal:

  ```
  $ pep8 <program>
  ```

  where `<program>` is the `.py` file whose style you want to check.

- Make sure your report doesn't exceed 400 lines, doesn't contain spelling mistakes, and doesn't contain lines that exceed 80 characters.

Acknowledgements This project is an adaptation of the Atomic Nature of Matter assignment developed at Princeton University by David Botstein, Tamara Broderick, Ed Davisson, Daniel Marlow, William Ryu, and Kevin Wayne.