Lab07 Randomization (10.14.19) CS103 Fall 2019

author john k johnstone jkj at uab dot edu
course CS103 Fall 2019

version Fall 2019

---

materials

- *lab07_19fa103.pdf* (this document)
- *lab07_19fa103.py* (with functions you will implement)
  I don't expect you to finish all these functions today: good exercises for outside lab too
- *lab07_tests_19fa103.py* (with test calls; go ahead and call as you implement)
- *lab07_render_19fa103.py* (visualizing your data; go ahead and call as you finish)
- argmin_stubbed.py and reverse_stubbed.py
- lab partner: let's switch up your partner this week, so you get to know more people

purpose

- practice generalization
- practice function calls (including functions you have designed)
- practice iteration
- practice using tuples (immutable) and lists (mutable)
- learn a bit about the Gaussian distribution, fundamental throughout science

The ability to build random test data is important. Randomization is also an opportunity to practice standard techniques in Python, such as function calls, types, and iteration. Finally, randomization will be useful in Monte Carlo techniques that we will explore in a later lab.

But this lab also illustrates the principle of generalizing an idea, gradually polishing code to larger and larger solutions.

Throughout this lab, look for opportunities to use functions that you have designed earlier to help implement functions you are designing now. For example, randPt2 would be useful in both randSeg2 and randTri2.

preparation

The normal drill.

Documentation for the random module will be useful: https://docs.python.org/3/library/random.html

Gaussian probability distribution

In your earlier use of the random module, you have been choosing random values from the uniform distribution: equal probability of every value. Rather than choosing a value at random with equal probability of lying anywhere in some interval (a, b), it is often useful to choose the value around some float f with values close to f more likely and values far from f less likely. The Gaussian distribution is the perfect way to do this. (Note: the Gaussian distribution is also called the normal distribution.)

The top right image on the **Wikipedia page** is a nice illustration of the Gaussian distribution.
Quick intro to the Gaussian distribution:
The mean μ specifies the center of the Gaussian curve (0 for the red and blue distributions) and
the standard deviation σ (or variance σ^2) tells you how spread out the distribution is
(σ of red curve is larger than blue curve, so red curve has a larger spread).

Compare the image of the Gaussian to the image of the uniform distribution (a bit boring in contrast):
Wikipedia page on continuous uniform distribution

Python gives you the random.gauss function in the random module, which implements the Gaussian distribution for you. Notice that this function expects the mean mu and the standard deviation sigma.

I don't expect you to know anything more about the Gaussian distribution, just use it in your functions. You will learn a lot more about this crucial distribution in future courses (e.g., CS355 Probability and Statistics for CS and data science/AI courses).

in-class exercise

randPt2 build a random point in 2-space, with uniform distribution
(equal probability of lying anywhere in the box);
useful function: **random.uniform**

exercises

In all of these questions, restrict the random points to the box of radius r centered at origin, where r is an int (or float) parameter to the function. In 2-space, this box has corners $(-r, -r), (r, -r), (r, r), (-r, r)$. Put another way, it is $[-r, r] \times [-r, r] = [-r, r]^2$.

randPt2 build a random point in 2-space, with uniform distribution
randSeg2 build a random line segment in 2-space
randTri2 build a random triangle in 2-space

intermission implement the stubbed functions argmin_stubbed.py using a for loop

randCld2_v1 build a random point cloud of size n in 2-space, with uniform distribution
revisit randSeg2 randSeg2 and randTri2 in light of this new function randCld2_v1;
can you rewrite them to use randCld2_v1?
randCld2_v2 build a random point cloud in 2-space, with uniform distribution, as follows:
continue adding points until you have exactly n points in the first quadrant
randCld3 build a random point cloud of size n in 3-space, with uniform distribution;
what helper function should you probably write first? hint: this looks a lot like randCld2_v1
helper function: a function whose sole purpose is to help implement another function

intermission implement the stubbed function reverse_stubbed.py using a for loop

randGaussPt2 build a random point in 2-space, with Gaussian distribution,
centered at origin, so higher probability of lying near the origin;
useful function: **random.gauss**
randGaussCld2 build a random point cloud of size n in 2-space, sampled from a Gaussian distribution
randUnit2 build a random unit vector in 2-space (a unit vector is a vector of length 1)
randCldUnit2 build many random unit vectors in 2-space

challenges

a build a random equilateral triangle
b build a random right triangle
c can you avoid skinny triangles while building a random triangle?
the extra parameter would be the minimum angle
d one of the turtle challenges from lab06 (if turtle works for you);
(a different one than you did last week if you handed one in)

deliverables

Please strip the docstrings from the functions when you submit, so that the TA can more easily grade them quickly.

B attendance, in-class exercise, and completion of argmin stubbed function
A B exercises + randCld2_v1/randGaussPt2/randUnit
A+ A exercises + one of the challenges