There are two problems here from the second half of the course. Unlike the homework assignments, these are extensions to the topics we've discussed, that is, you'll be learning new concepts as you work on these problems. Please follow these two guidelines:

- Please work individually and don't share your work. My grading is pretty lenient and even if you don't complete all of it, that's OK.
- For some parts, you are expected to re-use code from class. But you will also need to write your own code. For any code that is your own, please provide short comments explaining what the code is doing.

1 Brownian Motion

The *first-hitting time* of a Brownian motion is defined as the first time the Brownian motion hits a specific value. For any value m, we'll denote it using τ_m. For example, τ_1 is the first time that the Brownian motion reaches the value 1. Here is a well-known result:

The probability that the first-hitting time for any level $m \neq 0$ is given by:

$$\mathcal{P}(\tau_m \leq t) = 2\Phi\left(-\frac{|m|}{\sqrt{t}}\right),$$

where $\Phi(\cdot)$ refers to the cdf of a standard normal distribution.

Your task is to demonstrate this result computationally for two values of m: 0.5 and 1. Here are the steps you are expected to take:

a. Set a value of m.

b. Create an array of length 10000 to store the first-hitting times.

c. Generate a standard Brownian Motion **until it is equal to** m.

d. Store the time from step c. in the array from step b.

e. Repeat c - d 10000 times.

f. Now, for any given t, you can estimate $\mathcal{P}(\tau_m \leq t)$ using the fraction of your array that is less than or equal to t. Instead of doing this manually for every possible value of t, you can use the *ECDF* function in **statsmodels** to compute the empirical probabilities. Create a plot with your array of first-hitting times on the x-axis and the empirical probabilities on the y-axis.

g. Create a new array defined as $2\Phi\left(-\frac{|m|}{\sqrt{t}}\right)$, where m is your value from step a. and t is from the array of first-hitting times.

At the end, you'll have two plots, one for each m.

2 Binomial Model

We've covered option pricing using the Binomial Model in detail. Consider an American Put option on a stock that pays no dividends. As we work backwards through the option tree, at each node we check whether the option is worth exercising or not. Your task is to determine the *exercise boundary* for an American put, which outlines the highest stock price at each time for which the option is worth exercising.

Create a new function that returns *(time, boundary)* using the American Put code developed in class and modifying it in the following way:

a. Create a 1-dimensional array *boundary*. The size should be equal to the number of levels of the option pricing tree.

b. At the final level of the tree, the put option will be worth exercising only if the stock price is below the strike price, K. Therefore, the **maximum stock price at which the option is worth exercising** is K. Set *boundary[n]* $= K$.

c. As you work backwards, for any level j, check for the highest value of the stock price at which the option is worth exercising. Set *boundary[j]* to be that stock price.

d. As you get closer to the beginning of the tree, that is level 0, you'll find that the option won't be worth exercising anymore. In this case, set the *boundary* value at these levels to be equal to the previous value of *boundary*. For example, if you find that at level 5 there is no stock price at which the option is worth exercising, set *boundary[5] = boundary[6]*.

Consider an American option with $S_0 = 100$, $K = 100$, $T = 1/4$, $r_f = 10\%$. Create a plot with t on the x-axis and the exercise boundary on the y-axis for:

- $\sigma = 20\%$ using $n = 100$ and $n = 1000$ (in the same plot)
- $\sigma = 40\%$ using $n = 100$ and $n = 1000$ (in the same plot)