Problem Set #5

Due Thursday, March 4, 2015, at the beginning of class.

Be sure to read each question carefully and answer all its parts.

1) **Functional Enrichment Analysis** (30 points in total):

A. (5 points) Write a function `my_factorial(n)` that calculates and returns the factorial of n (usually denoted by n!). Remember, n! is the product of all positive integers less than or equal to n. For example, 5! = 5 x 4 x 3 x 2 x 1 = 120. Note that according to the convention, the value of 0! is defined as 1.

B. (5 points) Write a function `my_choose(n,k)` that calculates and returns the binomial coefficient \( \binom{n}{k} \), also known as “n choose k”. Remember, the binomial coefficient can be calculated according to the following formula:

\[
\binom{n}{k} = \frac{n!}{k!(n-k)!}.
\]

You can (and should) use the `my_factorial` function you implemented above.

C. (5 points) Write a function `my_hypergeometric(N, m, n, k)` that calculates and returns the probability that a set of n balls, drawn randomly from an urn with N balls of which m are blue, includes exactly k blue balls. Remember, this probability is given by:

\[
P(X = k) = \binom{m}{k} \binom{N-m}{n-k} \frac{1}{\binom{N}{n}}.
\]

You can (and should) use the `my_choose` function you implemented in the previous section.

D. (10 points) Finally, write a function `my_hypergeometric_tail(N, m, n, k)` that calculates and returns the probability that a set of n balls, drawn randomly from an urn with N balls of which m are blue, includes k or more blue balls.

E. (5 points) Now, assume that you tested a set of 100 genes, of which 25 are annotated with the function: splicing. You found that 10 of the 100 genes are up-regulated in a certain cancer tissue. Of these 10, 5 are annotated with the splicing function. Use the functions you wrote above to determine whether this set of 10 up-regulated genes is enriched for the splicing function. What is the p value of the enrichment (i.e., what is the probability of finding 5 genes or more annotated with splicing).

2) **Simple Random Walks** (30 points in total):

A simple random walk is an iterative process, in which in each step you can randomly add either +1 or -1.

A. (20 points) Write a function that simulates an M-step random walk, starting at 0, and returning the absolute value of the maximum deviation from 0 (either positive or negative) that was reached during the walk. M (the length of the walk) should be passed to the function as an argument.
You can use the following functions:

`random.random()`, returns a random floating point number between 0 and 1. (Don’t forget to import the module `random`).

`math.fabs(x)` returns the absolute value of x. (Don’t forget to import the module `math`).

B. (10 points) Using the function you implemented above, find the average maximum deviation from 0 (remember – use the absolute value of the deviation) during a 10-step random walk. Your program should calculate the average over 100 different walks. Similarly, calculate the average maximum deviation during a 100-step walk, a 1,000-step walk, and a 10,000-step walk.

C. (Challenge problem) Can you guess what is the relationship between the length of the walk and the expected maximum deviation?

3) (15 points) Write a Python class called `Dice`, which represents a set of playing dice. The constrictor of this class should get as arguments the number of dice in the set and the number of sides in these dice (a standard dice have 6 sides, but dice with more sides do exist). The class should provide a function `Roll()` that returns a list of numbers that represent a roll of the dice. See example below:

```python
>>> craps_dice = Dice(2, 6) # Craps is played with a pair of standard dice
>>> print craps_dice.Roll()
[3, 4]
>>> print craps_dice.Roll()
[6, 6]
>>> print craps_dice.Roll()
[5, 1]
>>> quest_dice = Dice(3, 10) # Quest is played with a 3 10-sided dice
>>> print quest_dice.Roll()
[7, 4, 9]
>>> print quest_dice.Roll()
[1, 10, 3]
```

4) (25 points) Add the function `addDays(n)` to the class `Date`. This function should add n days to the current date. Make sure to correctly handle transitions across months AND across years (when necessary). Take into account the different number of days in each month. You can use numbers to represent the month in the class (as in Sample Problem #2). You do not need to “translate” a month name into a number – just have the “user” provide the month number rather than its name when creating a new Date object. See example below:

```python
>>> mydate = Date(22, 11, 1976)
>>> mydate.printUK()
22 . 11 . 76
>>> mydate.addDays(5)
>>> mydate.printUK()
27 . 11 . 76
>>> mydate.addDays(21)
>>> mydate.printUK()
18 . 12 . 77
>>> mydate.addDays(49)
>>> mydate.printUK()
5 . 2 . 78
```