# CS 237 Fall 2019 Homework Five Solution

### Due date: PDF file due Thursday October 10th @ 11:59PM in GradeScope with 6-hour grace period

### Late deadline: If submitted up to 24 hours late, you will receive a 10% penalty (with same 6 hours grace period)

## General Instructions

Please complete this notebook by filling in solutions where indicated. Be sure to "Run All" from the Cell menu before submitting.

There are two sections to the homework: problems 1 - 8 are analytical problems about last week's material, and the remaining problems are coding problems which will be discussed in lab next week.

In [44]:

```
# Here are some imports which will be used in code that we write for CS 237


# Imports potentially used for this lab


import matplotlib.pyplot as plt   # normal plotting
import numpy as np

from math import log, pi,log,floor       # import whatever you want from math
from random import seed, random
from scipy.special import comb
from collections import Counter

%matplotlib inline

# Calculating permutations and combinations efficiently

def P(N,K):
    res = 1
    for i in range(K):
        res *= N
        N = N - 1
    return res

def C(N,K):
    return comb(N,K,True)     # just a wrapper around the scipy function


# Useful code

def show_distribution(outcomes, title='Probability Distribution'):
    num_trials = len(outcomes)
    X = range( int(min(outcomes)), int(max(outcomes))+1 )
    freqs = Counter(outcomes)
    Y = [freqs[i]/num_trials for i in X]
    plt.bar(X,Y,width=1.0,edgecolor='black')
    if (X[-1] - X[0] < 30):
        ticks = range(X[0],X[-1]+1)
        plt.xticks(ticks, ticks)
    plt.xlabel("Outcomes")
    plt.ylabel("Probability")
    plt.title(title)
    plt.show()

# This function takes a list of outcomes and a list of probabilities and
# draws a chart of the probability distribution.

def draw_distribution(Rx, fx, title='Probability Distribution for X'):
    plt.bar(Rx,fx,width=1.0,edgecolor='black')
    plt.ylabel("Probability")
    plt.xlabel("Outcomes")
    if (Rx[-1] - Rx[0] < 30):
        ticks = range(Rx[0],Rx[-1]+1)
        plt.xticks(ticks, ticks)
    plt.title(title)
    plt.show()

def round4(x):
    return round(x+0.00000000001,4)

def round4_list(L):
    return [ round4(x) for x in L]
```

## Analytical Problem Instructions

The Problems 1, 2, and 4 ask you to "describe" a random variable, which means:

> (i) Give $R_X$ (you may schematize it if it is very complicated or infinite);
> (ii) List out the values of $f_X$ corresponding to each element of $R_X$;
> (iii) Draw a probability distribution, using the function `draw_distribution` provided in the previous cell.
> (iv) Give $E(X)$;
> (v) Give $Var(X)$ and $\sigma_X$ .

As always, round to 4 decimal places **at the last stage**, using the functions `round4(...)` and `round4_list(...)` given above.

A nice way to approach these is to do any complicated calculations in Python and then if you have to change something you won't have to redo all the calculations. Plus, you will make fewer mistakes in calculation. However, there is no need to do this for simpler problems.

I also **strongly** recommend creating new variables for each problem, for example Rx1, Rx2, etc. for the range of the random variable in problems 1, 2, etc. That way, you won't have problems if you forget and use the wrong variable! You can also refer to previous results without problems.

Following Problem One is an example of what I mean (it is a simple problem, but I am showing you how you could approach it).

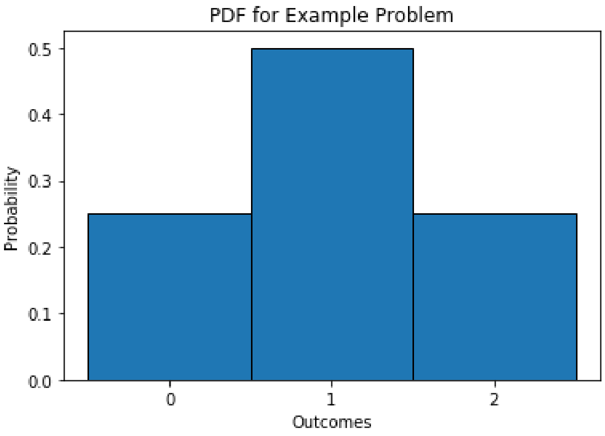
You are not **required** to do it this way, but I *encourage* you to do something similar.

## Example Problem

*Describe* the random variable X = "the number of heads showing on 2 flipped fair coins"

```
In [45]: Rx0 = list(range(3))

         def f0(k):                        # if you write f as a Python function, then you can create the l
             return  C(2,k)/4              # fx by using a list comprehension, as shown here:

         fx0 = [ f0(k) for k in Rx0 ]

         print("Solution:\n")
         print("(i)   Rx =",Rx0)
         print("(ii)  fx =",round4_list(fx0))             # in case you get complicated decimals, round to
         print("(iii)")

         draw_distribution( Rx0, fx0, title='PDF for Example Problem')

         (E0,V0,s0) = stats(Rx0,fx0)         # uses function you will write in Problem 1

         print("(b)   E(X) =",round4(E0))
         print("(c)   Var(X) = " + str(round4(V0)))
         print("      sigma_X = " + str(round4(s0)))
```

```
Solution:

(i)   Rx = [0, 1, 2]
(ii)  fx = [0.25, 0.5, 0.25]
(iii)
```



```
(b)   E(X) = 1.0
(c)   Var(X) = 0.5
      sigma_X = 0.7071
```

## Problem One

In order to understand how to calculate these statistical measures we have been learning this week, you will write a function which will return the most important measures, and since these involve very similar kinds of computations, to avoid extra work, we will return this as a triple (expected-value,variance,standard deviation).

Complete the following function stub, and demonstrate it on the random variable $X$ from the sample problem.

[Optional, but a great idea: Write a function PrintSolution(...) which prints out all the answers as shown in the Example Problem!]

In [46]:

```
# Function to compute most common one-number measures for X
def stats(Rx,fx):
    Ex = sum( [ Rx[k]*fx[k] for k in range(len(Rx))] )
    Ex2 = sum( [ Rx[k]*Rx[k]*fx[k] for k in range(len(Rx))])
    V = Ex2 - Ex**2
    return (Ex,V,V**0.5)

(E1,V1,s1) = stats(Rx0,fx0)

print("Solution:\n")
print("E(X) =",E1,"  Var(X) =",V1, "  sigma(X) =", round4(s1))
```

```
Solution:

E(X) = 1.0   Var(X) = 0.5   sigma(X) = 0.7071
```

In [47]:

```
def PrintSolution(Rx,fx,title="Probability Distribution Function"):
    print("(i)\t Rx =",Rx)
    print("(ii)\t fx =",round4_list(fx))
    print("(iii)")

    draw_distribution(Rx, fx,title)

    (E,V,s) = stats(Rx,fx)

    print("(b)   E(X) =",round4(E))
    print("(c)   Var(X) =",round4(V))
    print("      sigma_X =",round4(s))
```

## Problem Two

Suppose you deal a 5-card hand from a standard deck which has been shuffled well.

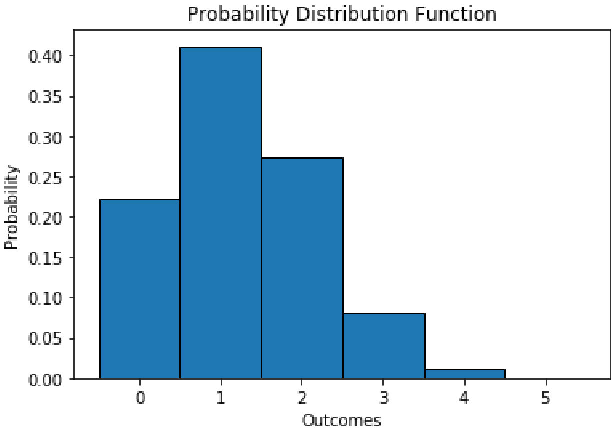
Let $X$ = "The number of Spades occurring in the hand."

*Describe* the random variable $X$.

In [48]:

```
Rx2 = [0,1,2,3,4,5]

def f2(k):
     return (C(13,k)*C(39,5-k)/C(52,5))

fx2 = [ f2(k)  for k in Rx2 ]

print("Solution:")
PrintSolution(Rx2,fx2)
```

```
Solution:
(i)      Rx = [0, 1, 2, 3, 4, 5]
(ii)     fx = [0.2215, 0.4114, 0.2743, 0.0815, 0.0107, 0.0005]
(iii)
```



```
(b)   E(X) = 1.25
(c)   Var(X) = 0.864
      sigma_X = 0.9295
```

**Problem Three**

We refer to the random variable $X$ from Problem Two.

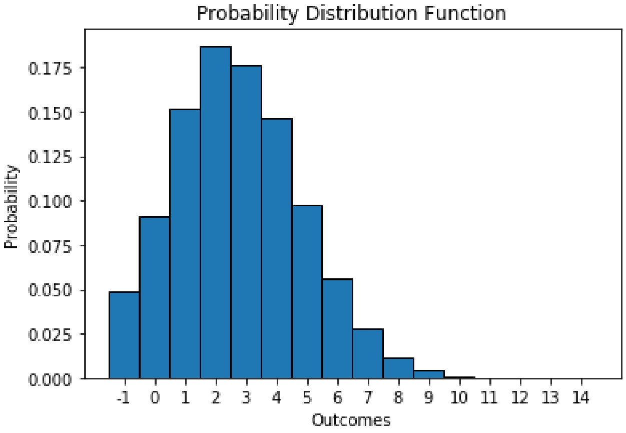
*Describe* the random variable $Y = 2X + X - 1$

Hint: when more than one instance of a random variable is involved, it is often useful to draw a matrix of all possibilities. Consider the two random variables $2X$ and $(X - 1)$ and draw a matrix of each of the two RVs and their sum; since these two RVs are independent, you can calculate the probabilities by multiplication, as shown in class. Or just put $X$ on each axis and calculate $2X + X - 1$ in the cells.

In [49]:

```
Rx3 = list(range(-1,15))
dict = {}
for x1 in Rx3:
    dict[x1] = 0

for r in range(len(Rx2)):
    for c in range(len(Rx2)):
        y = 2*Rx2[r] + Rx2[c] - 1
        dict[y] += fx2[r] * fx2[c]
fx3 = list(dict.values())

print("Solution:")
PrintSolution(Rx3,fx3)
```

```
Solution:
(i)      Rx = [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
(ii)     fx = [0.0491, 0.0911, 0.1519, 0.1873, 0.176, 0.1465, 0.0977, 0.0561, 0.0277, 0.0112,
0.0039, 0.0011, 0.0003, 0.0, 0.0, 0.0]
(iii)
```



```
(b)   E(X) = 2.75
(c)   Var(X) = 4.3199
      sigma_X = 2.0784
```

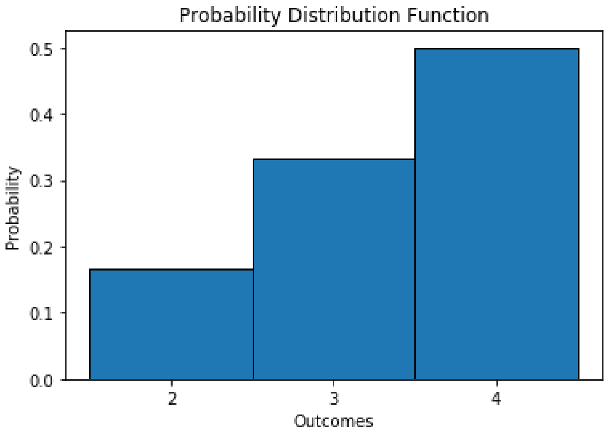
## Problem Four

Suppose we have a sack with $2$ red balls and $2$ black balls, and we draw balls *without replacement* until the **second red** ball is drawn.

*Describe* the random variable $X$ = "the number of balls drawn".

```
In [50]: Rx4 = [ 2, 3, 4 ]
         fx4 = [ 1/6, 1/3, 1/2 ]

         print("Solution:")
         PrintSolution(Rx4,fx4)
```

```
Solution:
(i)      Rx = [2, 3, 4]
(ii)     fx = [0.1667, 0.3333, 0.5]
(iii)
```



```
(b)   E(X) = 3.3333
(c)   Var(X) = 0.5556
      sigma_X = 0.7454
```

## Problem Five

A sack contains five balls, two of which are marked $\$1$ two $\$5$, and one $\$51$. One round of the game is played as follows: You pay me $\$10$ to select two balls at random (without replacement) from the urn, at which point I pay you the sum of the amounts marked on the two balls.

(a) *Describe* the random variable X = "the **net payout** in each round."

(b) Is this a fair game? Be precise and show all work.

(c) If your answer to (c) is "no," what should I charge for each turn to make it a fair game?

**Solution:**

Let X = "net payout on one play of game," which we analyze as follows:

$$S = \{(1,1),(1,5),(1,15),(5,1),(5,5),(5,15),(15,1),(15,5)\}$$

$$P = \left\{ \frac{2*1}{(5*4)}, \frac{2*2}{5*4}, \frac{2*1}{5*4}, \frac{2*2}{5*4}, \frac{2*1}{5*4}, \frac{2*1}{5*4}, \frac{1*2}{5*4}, \frac{1*2}{5*4} \right\}$$

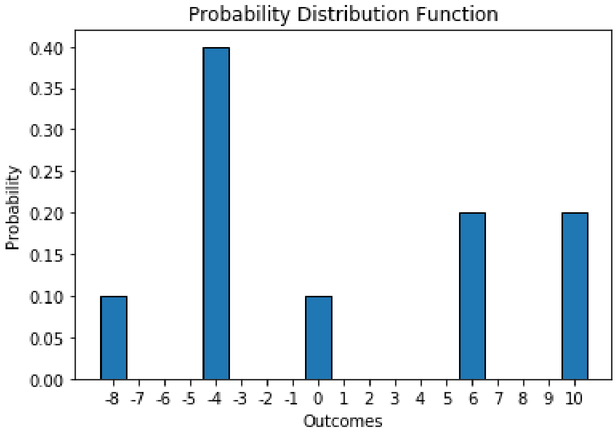
$$= \{0.1, 0.2, 0.1, 0.2, 0.1, 0.1, 0.1, 0.1\}$$

Net payout (sum of numbers on balls minus cost of 10 dollars):

$$= \{-8, -4, 6, -4, 0, 10, 6, 10\}$$

```
In [51]: Rx5    = [ -8,  -4,   0,   6,   10 ]
         fx5    = [ 0.1, 0.4, 0.1, 0.2, 0.2 ]

         print("(a)\n")
         PrintSolution(Rx5, fx5)
```

```
(a)

(i)      Rx = [-8, -4, 0, 6, 10]
(ii)     fx = [0.1, 0.4, 0.1, 0.2, 0.2]
(iii)
```



```
(b)   E(X) = 0.8
(c)   Var(X) = 39.36
      sigma_X = 6.2738
```

(b) Thus E(X) = -0.8 - 1.6 + 0 + 1.2 + 2 = 0.8 so on average you make 80¢ for each play, and it is NOT fair (to me!).

(c) I should charge $10.80 to make the game fair.

## Problem Six

Suppose you are playing a game with a friend in which you bet $n$ dollars on the flip of a fair coin: if the coin lands tails you lose your $n$ dollar bet, but if it lands heads, you get $2n$ dollars back (i.e., you get your $n$ dollars back plus you win $n$ dollars).

Let $X$ = "the amount you gain or lose."

(a) What is the expected return $E(X)$ on this game? (Give your answer in terms of $n$.)

Now, after losing a bunch of times, suppose you decide to improve your chances with the following strategy: you will start by betting $\$1$, and if you lose, you will double your bet the next time, and you will keep playing until you win (the coin has to land heads sometime!).

Let $Y$ = "the amount you gain or lose with this strategy".

(b) What is the expected return $E(Y)$ with this strategy? (Hint: think about what happens for each of the cases of $k = 1, 2, 3, \ldots$ flips).

(c) Hm ... do you see any problem with this strategy? How much money would you have to start with to guarantee that you always win?

(d) Suppose when you apply this strategy, you start with $\$20$ and you quit the game when you run out of money. Now what is $E(Y)$?

**Solution:**

(a) $R_X \;=\; \{-n, n\}, f_X = \{0.5, 0.5\}$, so clearly $E(X) = 0$.

(b) Here is a record of what happens for $k = 1, 2, 3, \ldots$ flips:

```
k = 1:   You win 1

k = 2:   You lose 1 on the first flip, bet 2 on the second flip and win 4;
              your net is 4 - (1+2) = 1
k = 3:   You lose 1 + 2 on the first two flips, bet 4 on the third, and win 8;
              you net is 8 - (1+2+4) = 1
```

Clearly, you win $\$1$ no matter what happens, so $E(Y) = 1$. The precise calculation is

$$1 * \frac{1}{2} + 1 * \frac{1}{4} + 1 * \frac{1}{8} + \ldots \;=\; \$1$$

(c) You would need an infinite amount of money to make sure you win with probability 1.0.

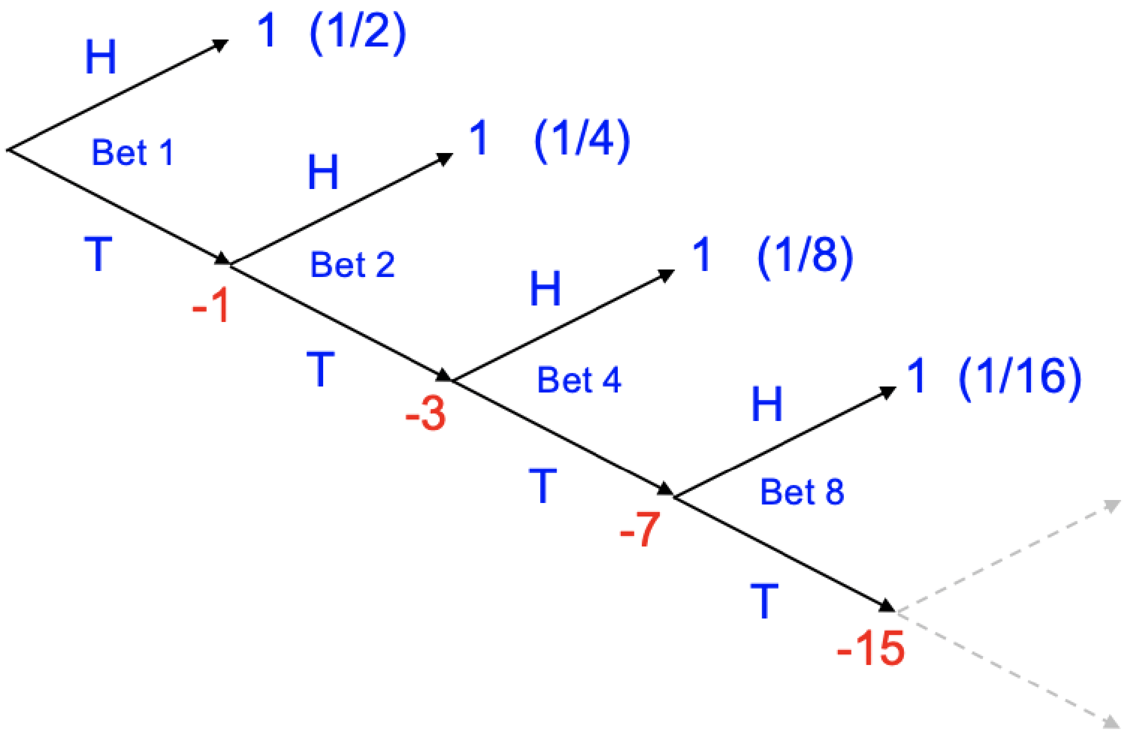
(d) The problem is that you may run out of money and can't double your bet! If you lose $K = 3$ times, you will have lost $1 + 2 + 4 = 7$ dollars and you can afford to double your bet to $8$ dollars, but if you lose $k = 4$ times, you will have lost $1 + 2 + 4 + 8 = 15$ dollars and can't double your bet! So you have

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} \;=\; \frac{15}{16}$$

probability of winning $\$1$, and $\frac{1}{16}$ probability of losing $15, so

$$1 * \frac{15}{16} - 15 * \frac{1}{16} \;=\; -\$0$$

It would turn out the same for any fixed limit (it is true after the first round, as we saw in part (a)), so, if you have an infinite amount of money, you can expect to win $1 on average, but otherwise, you can expect to break even on average. The following decision tree shows the situation:



This is just another way of saying "gambling is a tax on people who don't understand probability"!

## Problem Seven

Mr. Smith owns two appliance stores. In store A the number of TV sets sold by a salesperson is, on average, 13 per week with a standard deviation of 5. In store B the number of TV sets sold by a salesperson is, on average, 7 with a standard deviation of 4. Mr. Smith has a position open for a person to sell TV sets. There are two applicants. Mr. Smith asks one of them to work in store A and the other in store B, each for one week. The salesperson in store A sold 10 sets, and the salesperson in store B sold 6 sets.

Mr. Smith realizes that Store A typically has more customers and typically sells more TVs than Store B, so it would not be fair to simply compare the raw data. What he needs to do is "level the playing field" by comparing the two *after* adjusting for the differences between the two stores.

(a) Based on this information, which person should Mr. Smith hire?

(b) Suppose the person not hired asks Mr. Smith "Well, how many would I have had to sell to be exactly as good as the person you hired?" What should Mr. Smith say? (Not quite realistic, because the answer may not be an integer, but just give the floating point value.)

Hint: Use standardized random variables.

**Solution:**

Let $X_A$ be the number of TV sets the salesperson in store A sells and $X_B$ be the number of TV sets the salesperson in store B sells. Standardizing, we have that

$$X_A^* = \frac{10-13}{5} = -0.6$$

and

$$X_B^* = \frac{6-7}{4} = -0.25$$

Therefore, the number of TV sets the salesperson in store A sells is 0.6 standard deviations below the mean, whereas the number of TV sets the salesperson in store B sells is 0.25 standard deviations below the mean. So Mr. Norton should hire the salesperson who worked in store B.

(b) We want $k$ in the formula

$$\frac{k-13}{5} = -0.25$$

so

$$k = 13 - 5 * 0.25 = 13 - 1.2 = 11.8$$

## Problem Eight

Wayne is interested in two games, Keno and Bolita. To play Bolita, he buys a ticket for $1 marked with a number 1 .. 100, and one ball is drawn randomly from a collection marked with the numbers 1, ..., 100. If his ticket number matches the number on the drawn ball, he wins $75; otherwise he gets nothing and loses his $1 bet. To play Keno, he buys a ticket marked with the numbers 1 .. 4 and there are only 4 balls, marked 1, ..., 4; again he wins if the ticket matches the ball drawn; if he wins he gets $3; otherwise he again gets nothing and loses his bet.

(a) What is the expected payout (expected value of net profit after buying ticket and possibly winning something) for each of these games?

(b) What is the variance and standard deviation for each of these games?

(c) If he decides to play one (and only one) of these games for a very long time, which one should he choose? If he decides to try one of these games for a couple of times, just for fun, which one should he choose?

**Solution:** Let K = the net payout from one play in Keno, and B = the net payout from one play in Bolita. Then

```
    Rng(K) = { -1,   74 }            Rng(B) = { -1,    2 }
         p = { 0.99, 0.01 }               p = { 0.75, 0.25 }

      E(K) = -0.99 + 0.74 = -0.25      E(B) = -0.75 + 0.5 = -0.25


    Var(K) =  (-1-(-0.25))^2 0.99   Var(B) = (-1-(-0.25))^2 0.75


            + (74-(-0.25))^2 *0.01         + (2-(-0.25))^2 *0.25
           = 55.6875                       = 1.6875

   stdev(K) = 7.4624               stdev(B) = 1.2290
```

(a) The expected payout is -25¢ per play, i.e., on average they cost the same amount to play (and you should not expect to win in the long run!)

(b) Variance as shown above, obviously, Keno varies much more widely than Bolita.

(c) If Wayne has forgotten everything he ever knew about probability, and decides to play one of these for a long time, it doesn't actually matter which game he plays: he will lose an average of 25¢ per play. If Wayne has only temporarily forgotten his probability theory, and wants to play for a short time, then Bolita is a better game, since he is much more likely to have several payouts, compensating himself in part for his folly in playing this game.

## Lab Instructions

In homework 2, problem 11, we investigated how to generate random values (called "random variates" in the literature). In this lab, we will make this more explicit by considering three different ways for simulating a random variable:

> 1. By simulating the original physical experiment;
> 2. By inverting the CDF; and
> 3. By using an explicit formula.

In the last two, we will convert a random variate in the range $[0..1)$ created by the function `random()` into a random variate from a different distribution.

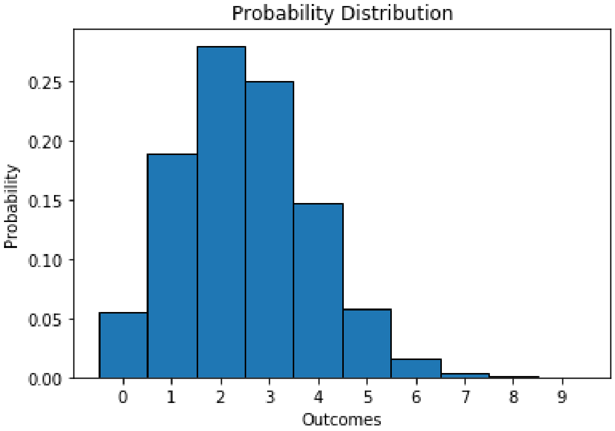
## Problem Nine: Generating a Binomial Distribution by Simulation

For this problem, complete the function stub to simulate the following random variable: Suppose we have a (possibly) unfair coin where the probability of Head is $p$ and we flip this coin $N$ times. Let $X$ = "the number of heads showing."

Demonstrate your solution by generating $10^5$ random variates for the parameters shown and displaying them using `show_distribution(...)`.

```
In [52]:
```

```
Demonstration:
```



## Problem Ten: Generating a Distribution by Inverting the CDF

In this problem we will investigate how to implement a random variable given by an arbitary probability distribution function. First, however, you need to know about the CDF, the Cumulative Distribution Function, which is a simple but crucially important idea, which you can understand by looking at the last slide on Lecture 8 (which I did not cover in lecture) or here (https://www.probabilitycourse.com/chapter3/3_2_1_cdf.php). Please look at this before continuing with the lab.

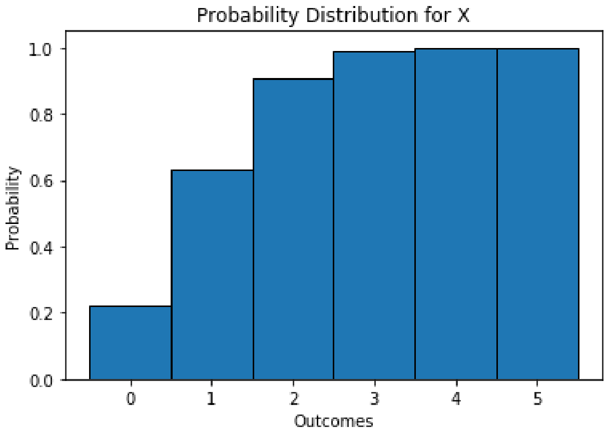
The basic idea in these problems is that we can essentially "invert" the CDF to obtain a function from a random variate in the range $[0..1)$ into a random variable from the given distribution.

### TODO, Part (a): Calculating the Cumulative Distribution Function

Complete the following to calculate $F_X$ for the CDF of the probability distribution $f_X$, and then demonstrate your code by calculating and displaying the CDF for the random variable of $X$ from the Example Problem.

```
In [53]:
```

```
Fx = [0.2215, 0.633, 0.9072, 0.9888, 0.9995, 1.0]
```



## Part (b): Generating random variates by inverting the CDF

The basic idea here is that the CDF is a function from outcomes to probabilities:

$$F_X \ : \ R_X \rightarrow [0..1)$$

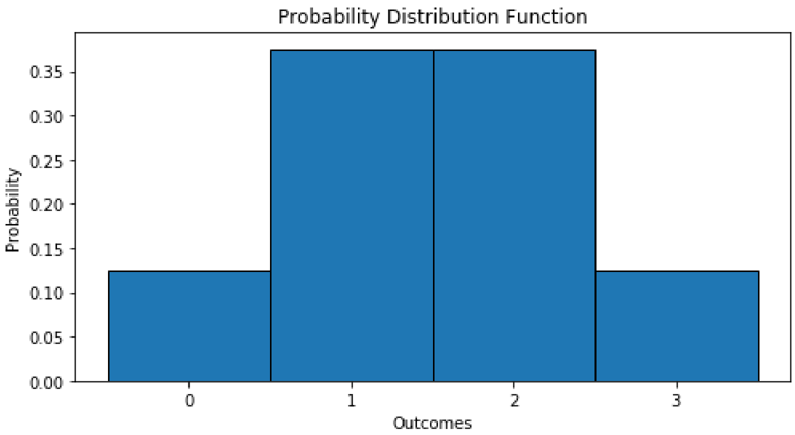
if we invert this function, we get a function from the interval $[0..1)$ into the outcomes:

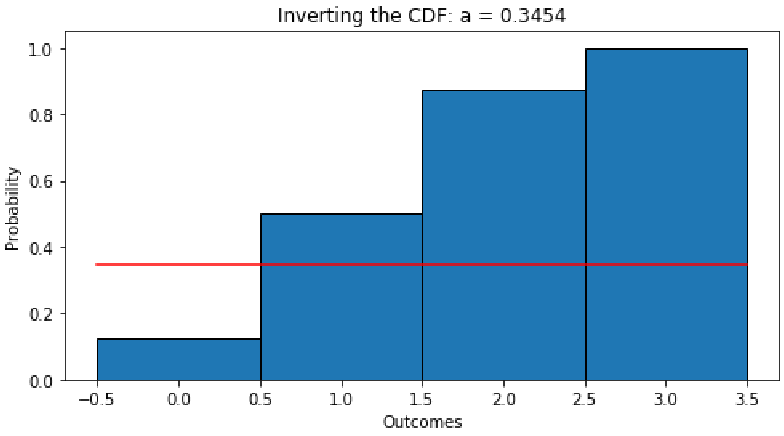
$$F_X^{-1} \ : \ [0..1) \rightarrow R_X$$

The algorithm for doing this is actually very simple: just generate a random value $a$ in the range $[0..1)$ and look for the first bin which is greater than $a$; output the corresponding outcome.

The next cell contains a demonstration of this idea: run the cell a few times to see where the random value ends up: the first bin that the red line intersects going left to right indicates the outcome that is output. Be sure you understand how this process works, and what number would be output, for each test.

In [54]:





### TODO for 10 (b):

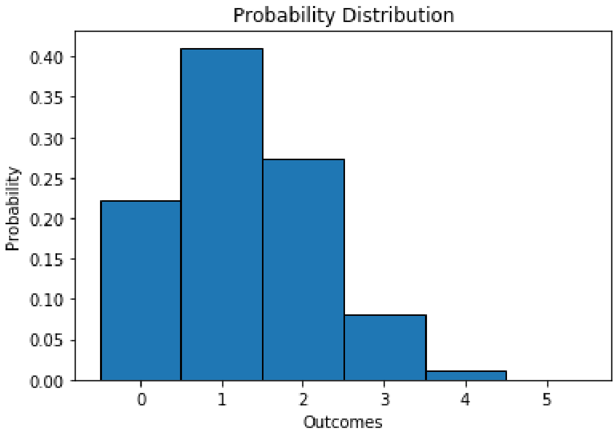
Complete the following code template to generate random variates for a given random variable (represented by Rx and fx), using the code from Part (a).

Demonstrate your code by generating $10^5$ variates from the random variable $X$ from Problem Two and display it using `show_distribution(...)`.

Hint: Compare with the theoretical distribution from Problem Two. They should be very close!

In [55]:

```
Demonstration:
```



## Problem Eleven: Creating Random Variates for Standard Distributions by

## Inverting the CDF

Now we will apply the technique from the last problem to generate random variates for two common distributions, which we will study in detail this week, however, we have already seen them many times; both of them involve a (possibly) unfair coin, where the probability of Head is p (and the probability of Tails is thus 1 - p):

> Binomial B(N,p): This is just the number of heads showing on N flipped coins, where the probability of heads is p (and the probability of Tails is thus 1-p).
>
> Geometric G(p): This is the number of flips you make until the first Head appears on a coin whose probability of Head is p.

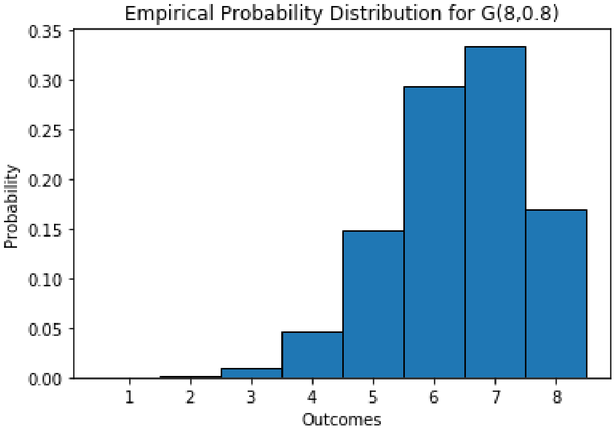
### Part (a) Binomial Variates

Display the result of generating $10^5$ random variates from the Binomial Distribution with $N = 8$ and $p = 0.8$ using `show_distribution`.

Hint: You may look at the Distributions Notebook on the class web site to see the formula for the PDF for the Binomial, or wait until lecture....

In [56]:

```
Solution:
```



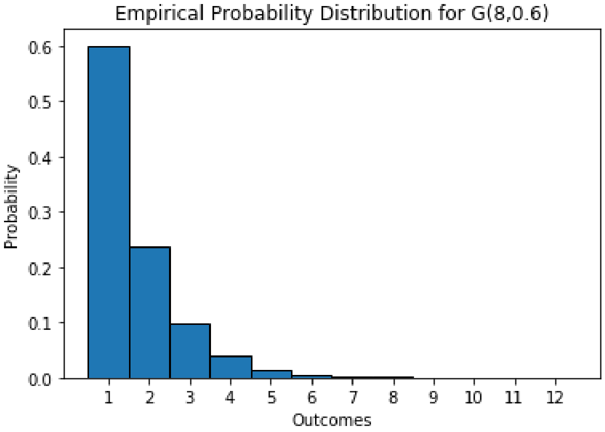
### Part (b) Geometric Variates

Although $R_x$ is infinite, we will only approximate this distribution by considering the first 20 outcomes. This will potentially create an error, since of course it is possible for the value produce to be larger than 20, however, the probability is so small we will not worry about it. Note that our code for the CDF did not calculate the last bin, but simply set it to 1.0, which will make sure our generation of random variates does not crash.

Display the experimental distribution for the Geometric Distribution with $p = 0.6$, using `show_distribution` from the beginning of the notebook, for $10^5$ trials and the given N and p.

Hint: You may look at the Distributions Notebook on the class web site to see the formula for the PDF for the Geometric, or wait until lecture....

```
In [57]:
```



## Problem Twelve: Generating the Geometric Distribution by Explicit Formula

Now we will explore using an explicit function for the inverse of the CDF. This is not possible for all distributions, but when it is, it the simplest (and most efficient) method.

The following formula is from Wikipedia (https://en.wikipedia.org/wiki/Geometric_distribution): if U is a random variable uniformly distributed in the range [0..1), then

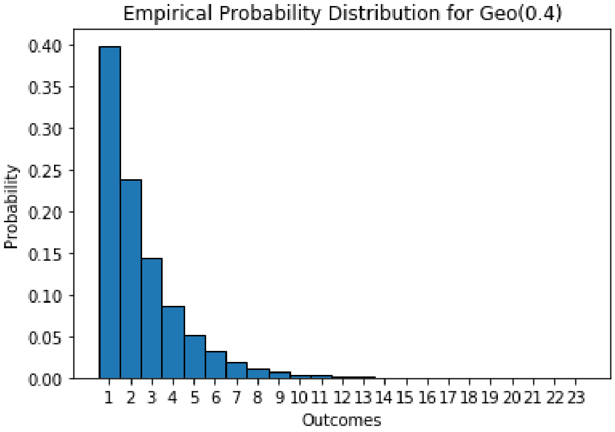
$$1 + \lfloor \ln(U) \ / \ \ln(1-p) \rfloor$$

is an integer which is distributed according to the Geometric Distribution with probability p.

Note: $\ln$ is log to the base $e$ (just `log(...)` in Python).

For this problem, simply complete the following function stub and demonstrate it as shown.

```
In [58]:
```


Empirical Probability Distribution for Geo(0.4)
Probability
Outcomes