CS 237 Fall 2019 Homework Five

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 [1]:

```
# 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       # 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

Some of the following problems 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 σ_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.

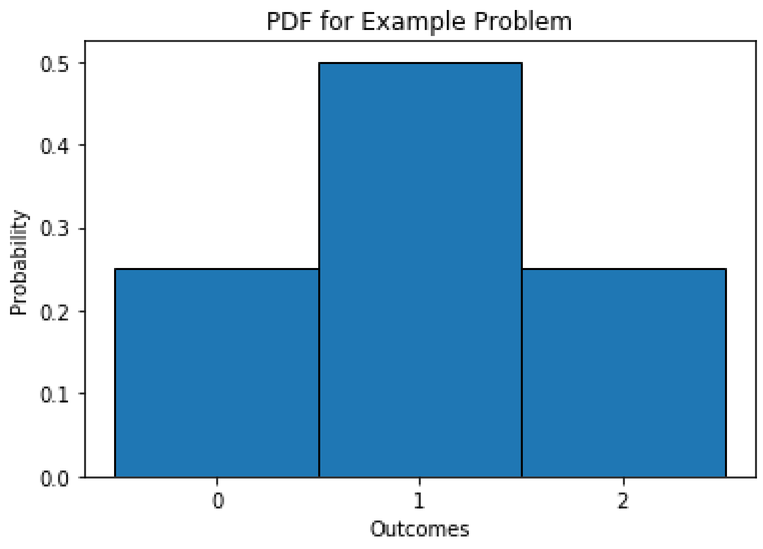
Example Problem

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

In [2]:

```
Rx0 = list(range(3))

def f0(k):                          # if you write f as a Python function, then you ca
    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 deci
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)
```



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 [3]:

```
# Function to compute most common one-number measures for X
def stats(Rx,fx):
    expectValue = 0
    var = 0
    for x in range(len(Rx)):
        expectValue = expectValue + Rx[x] * fx[x]
    for x in range(len(Rx)):
        var = var + Rx[x] * Rx[x] * fx[x]
    var = var - expectValue*expectValue
    standardD = var ** (0.5)
    return (expectValue,var,standardD)
(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
```

Problem Two

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

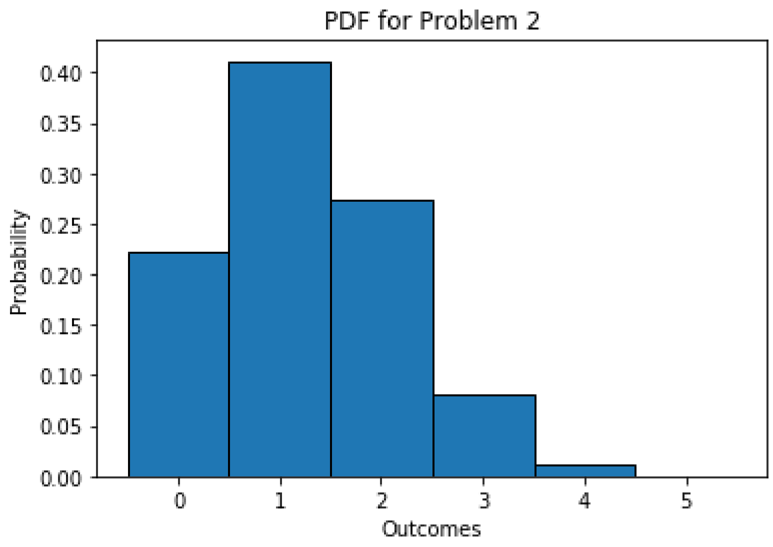
Let X = "The number of Spades occurring in the hand."

Describe the random variable X.

In [4]:

```
Rx2 = list(range(6))            # your code here

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

fx2 = [f2(k) for k in Rx2]            # your code here

print("Solution:\n")
print("(i)   Rx =",Rx2)
print("(ii)  fx =",round4_list(fx2))            # in case you get complicated deci
print("(iii)")
draw_distribution( Rx2, fx2, title='PDF for Problem 2')

(E2,V2,s2) = stats(Rx2,fx2)           # uses function you will write in Problem 1

print("(iv)   E(X) =",round4(E2))
print("(v)   Var(X) = " + str(round4(V2)))
print("       sigma_X = " + str(round4(s2)))
```

```
Solution:

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



```
(iv)   E(X) = 1.25
(v)   Var(X) = 0.864
      sigma_X = 0.9295
```

Problem Three

We refer to the random variable X from Problem Two.

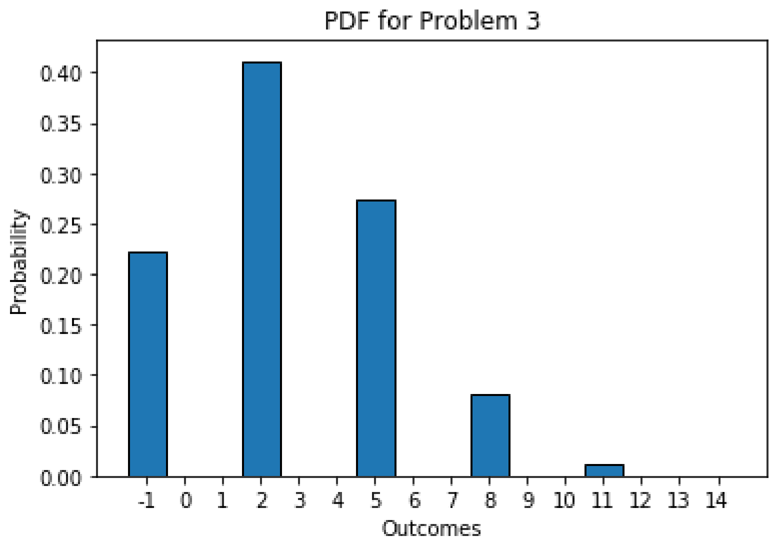
Describe the random variable $Y = 2X + X - 1$

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 [5]:

```
Rx30 = [2*Rx2[x] for x in list(range(6))]
Rx31 = [Rx2[x] -1 for x in list(range(6))]

Rx3 = [Rx30[x] + Rx31[x] for x in list(range(6))]

fx3 = fx2

print("Solution:\n")
print("(i)   Rx =",Rx3)
print("(ii)  fx =",round4_list(fx3))           # in case you get complicated deci
print("(iii)")
draw_distribution( Rx3, fx3, title='PDF for Problem 3')

(E3,V3,s3) = stats(Rx3,fx3)          # uses function you will write in Problem 1

print("(iv)   E(X) =",round4(E3))
print("(v)   Var(X) = " + str(round4(V3)))
print("      sigma_X = " + str(round4(s3)))
```

```
Solution:

(i)   Rx = [-1, 2, 5, 8, 11, 14]
(ii)  fx = [0.2215, 0.4114, 0.2743, 0.0815, 0.0107, 0.0005]
(iii)
```



```
(iv)   E(X) = 2.75
(v)   Var(X) = 7.7757
      sigma_X = 2.7885
```

Problem Four

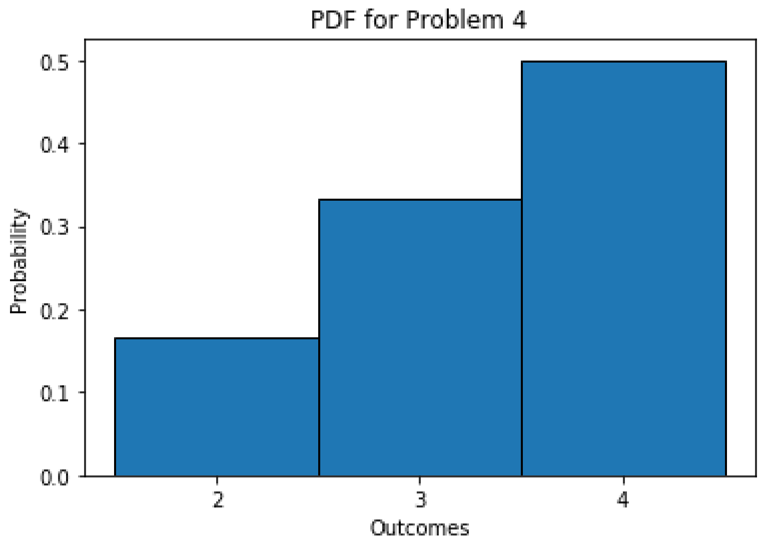
Suppose we have a sack with 2 red balls and 2 black balls, and we draw balls ***without replacement*** until the **second red** ball is drawn.

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

In [6]:

```
Rx4 = [2,3,4]
fx4 = [0.1667, 0.3333, 0.5000]

print("Solution:\n")
print("(i)   Rx =",Rx4)
print("(ii)  fx =",round4_list(fx4))           # in case you get complicated deci
print("(iii)")
draw_distribution( Rx4, fx4, title='PDF for Problem 4')

(E4,V4,s4) = stats(Rx4,fx4)          # uses function you will write in Problem 1

print("(iv)   E(X) =",round4(E4))
print("(v)   Var(X) = " + str(round4(V4)))
print("      sigma_X = " + str(round4(s4)))
```

```
Solution:

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



```
(iv)   E(X) = 3.3333
(v)   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 $15. 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.

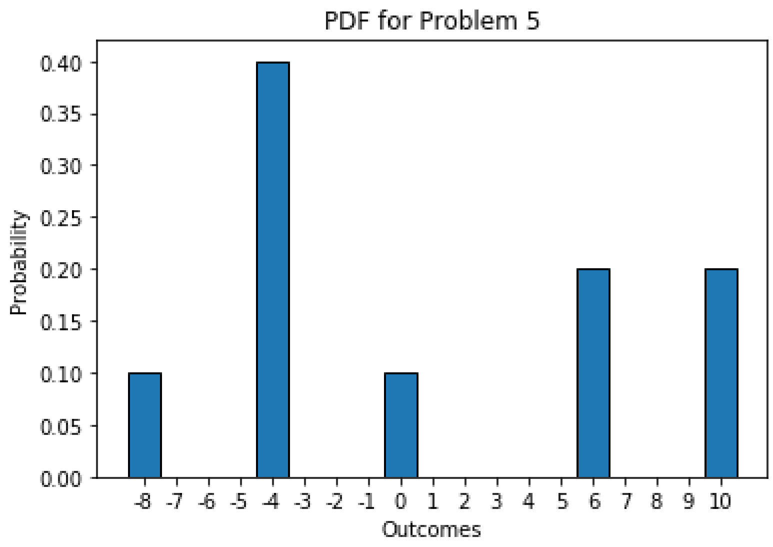
(c) If your answer to (b) is "no," what should I charge for each turn to make it a fair game?

Solution:

In [7]:

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

print("Solution:\n")
print("(i)   Rx =",Rx5)
print("(ii)  fx =",round4_list(fx5))            # in case you get complicated deci
print("(iii)")
draw_distribution( Rx5, fx5, title='PDF for Problem 5')

(E5,V5,s5) = stats(Rx5,fx5)          # uses function you will write in Problem 1

print("(iv)   E(X) =",round4(E5))
print("(v)   Var(X) = " + str(round4(V5)))
print("      sigma_X = " + str(round4(s5)))
```

```
Solution:

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



```
(iv)   E(X) = 0.8
(v)   Var(X) = 39.36
      sigma_X = 6.2738
```

(b) This is not a fair game because the expected value, or the mean of the game, is positive. Expected value is calculated by adding all the sums of each random variable with the corresponding probability of occurring. Here, the expected value is -0.8(0.1) + -4(0.4) + 0(0.1) + 6(0.2) + 10(0.2) = 0.8. In order to have a fair game, the expected value should be 0. However, since the expected value is 0.8, this game is not fair.

(c) In order to make this game fair, we need to change the price that gets paid. Since we need to have an expected value of 0, the sum of each random variable with the corresponding probability of occurring should be equal to 0. If we denote the amount charged as X, the equation becomes (2-x)(0.1)+(6-x)(0.4)+(10-x)(0.1)+(16-x)(0.2)+(20-x)(0.2). If we set it equal to 0, x is 10.8.

Therefore, we need to pay 10.8 dollars 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 $k = 1, 2, 3, \dots$ 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) Rx, the random variable, is [-n, n] because if you lose, you lose n doll
ars, but if you win, you get 2n dollars. Fx, the probability for each random
variable occurring, is [0.5,0.5] because there is 50 percent chance that hea
ds will occur and 50 percent that tails will occur when flipping a fair coi
n. If we calculate the expected value, we get n * 0.5 - n * 0.5 = 0
The expected value is 0

(b) In order to calculate the expected return, we need to know when the game
stops, with how much profit.
P(game stops at nth time) = 1/(2^(n-1)) * 1/2 = 1/(2^n)
The game stops when you win, which means that the game stops when the coin l
ands on heads. Because the coin is fair, the probability of getting heads is
0.5 and the probability of getting tails is 0.5. If the game stops at nth tr
ial, this means that prior to the nth trial, it all came out tails, which gi
ves the probability of 1/(2^(n-1)) and heads at the nth trial, which gives u
s 1/2. If we multiply them, we get 1/(2^n).
Amount earned after nth trial = (-1) + (-2) + (-4) + ...(-2^(n-1)) + 2^n = 1
Since you double up the money paid if you lose until you win, the will lose
 that double until you eventually win. Using this pattern, if you calculuate
the nth trial, you get a net positive 1 dollar.
E(Y) = 1

(c) There is a problem with this strategy. Since you do not stop until you w
in and there is a possibility of tails is 0.5 everytime you flip a coin, the
game can go forever. As n, the trial, goes to infinity, the amount of money
 guaranteed to win the 1 dollar is also infinity since you always need doubl
e the amount of money compared to the previous round.
Money needed: Infinity dollars to win one dollar

(d) With 20 dollars, the following is the maximum number of bets we can perf
orm
1st: -1 dollar (19 dollars left)
2nd: -2 dollar (17 dollars left)
3rd: -4 dollar (13 dollars left)
4th: -8 dollar (5 dollars left)
5th: -5 dollar- even though it is not double the previous amount (0 dollar l
eft)

Because you have the possibility of winning at any stage and the possibility
of losing and winning at the final 5th stage, when calculating the expected
 value, we need to consider all the possibilities. When you win at any stag
e, we earn a net profit of 1 dollar no matter which stage you are at, accord
ing to part b.

E(Y) = 1(1/2) + 1(1/4) + 1(1/8) + 1(1/16) + (5-15)(1/32) + (-20)(1/32)
     = 1/2 + 1/4 + 1/8 + 1/16 - 5/16 - 5/8
     = 1/2 + 1/4 + 1/8 - 1/4 - 5/8
     = 1/2 + 1/8 - 5/8
     = 1/2 - 4/8
     = 0
E(Y) = 0
```

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 varibles.

Solution:

```
(a) We need calculate the standardized random variable value.
    For store A:
    z = (10 - 13)/5 = -0.6

    For store B:
    z = (6-7)/4 = -0.25


   Since the salesperson in store B has higher standardized random variable
 value, the salesperson in store B should be hired by Mr. Smith.


(b) The person not hired is the salesperson in store A since the value of -
0.6 is less than -0.25. In order for him to be exactly as good as the hired
 salesperson, he needs to have standardized random variable value that is gr
eater than -0.25.
    (x - 13)/5 > -0.25
    (x - 13) > -1.25
    x > 11.25.

    In order for salesperson in store A to be exactly as good as the salespe
rson hired, he should sell 11.25 TV sets.
```

Problem Eight

Wayne is interested in two games, Keno and Bolita. To play Bolita, he buys a ticket for
1*markedwithanumber*1..100, *andoneballisdrawnrandomlyfromacollectionmarkedwiththenumbers*1,.
100.*Ifhisticketnumbermatchesthenumberonthedrawnball*, *hewins*
otherwise he gets nothing and loses his
1*bet*. *ToplayKeno*, *hebuysaticketmarkedwiththenumbers*1..4*andthereareonly*4*balls*, *marked*1, . . . , 4;3
againhewinsiftheticketmatchestheballdrawn; *ifhewinshegets*
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:

```
(a)For Bolita:
    Rb = [-1, 74]
    Fb = [0.99, 0.01]
    Eb = -1(0.99) + 74(0.01) = -0.25

    The expected payout for Bolita is -0.25 dollars

   For Keno:
    Rk = [-1, 2]
    Fk = [0.75, 0.25]
    Ek = -1(0.75) + 2(0.25) = -0.25

    The expected payout for Keno is 0.25 dollars

(b) For Bolita:
    Variance = -1*-1*0.99 + 74*74*0.01 - (-0.25)^2 = 55.6875
    Standard Deviation = 7.4624

    For Keno:
    Variance = -1*-1*0.75 + 2*2*0.25 - (-0.25)^2 = 1.6875
    Standard Deviation = 1.2990

(c) If he decides to play one of these games for a very long time, he should
choose Keno since it has a lower variance and standard deviation value than
 the values for Bolita. When playing for a long time, he should consider cho
osing the value that has lower variance even though the expected values for
 two games are the same because he has lower risk of losing tons of money.
    If he decides to play one of these games for a couple of times, just for
fun, he should play Bolita because when playing for a few games just for fu
n, Bolita has a high return value of 74 dollars of profit, which is benefici
al even if he only wins once.
```

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

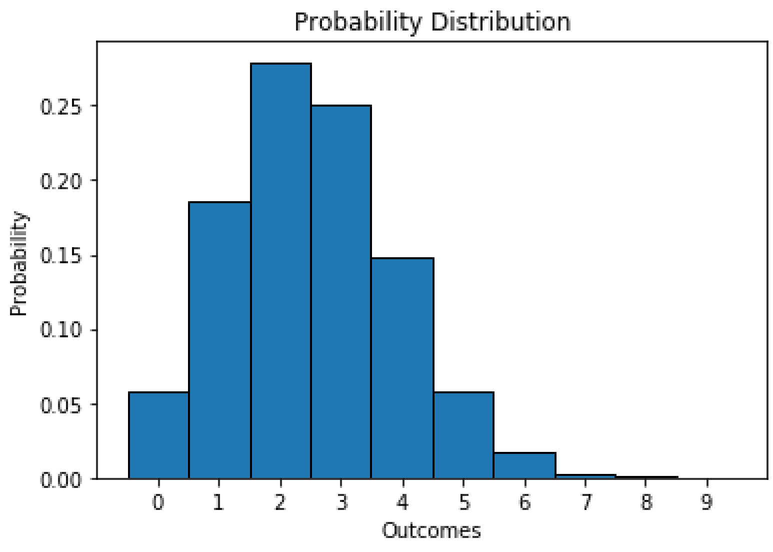
For this problem, complete the function stub to simulate the following random variable: Suppose we have a (possibly) unfair coin where the probability of Head is p and we flip this coin N times. Let X = "the number of heads showing."

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

In [8]:

```
# First, create a function which simulates the coin, where
# you return 1 with probability p and 0 with probability 1-p.

N = 10
p = 0.25

num_trials = 10**5


def coinFlip(p):
    if random() <= p:
        return 1
    else:
        return 0

def coinFlips(N,p):
    head = 0
    for x in range(N):
        if coinFlip(p) == 1:
            head = head + 1
    return head
lst = []
for x in range(num_trials):
    answer = coinFlips(N,p)
    lst.append(answer)


print("Demonstration:")
show_distribution(lst)
seed(0)
```

```
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.

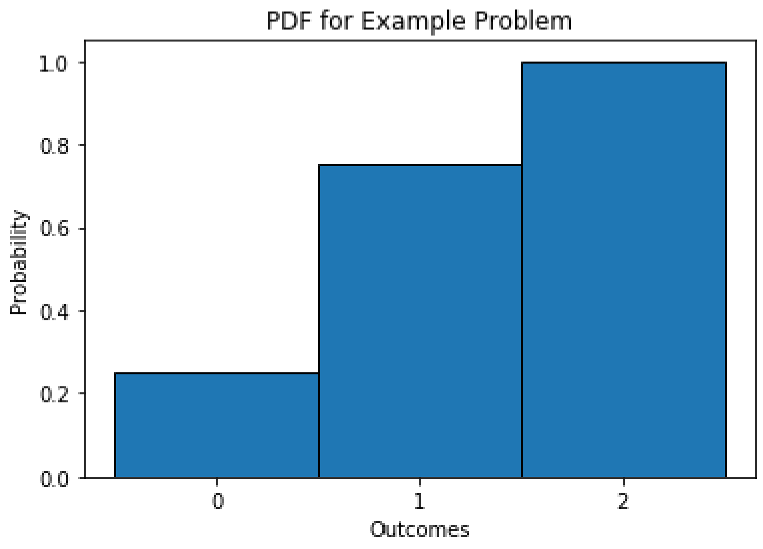
The basic idea in these problems is that we can essentially "invert" the CDF to obtain a function from a random variate in the range $[0..1)$ into a random variable from the given distribution.

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 [9]:

```
def CDF(fx):
    C = []
    sum = 0
    for x in fx:
        sum = sum + x
        C.append(sum)
    # just in case
    C[-1] = 1.0
    return C

newfx = CDF(fx0)

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



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:

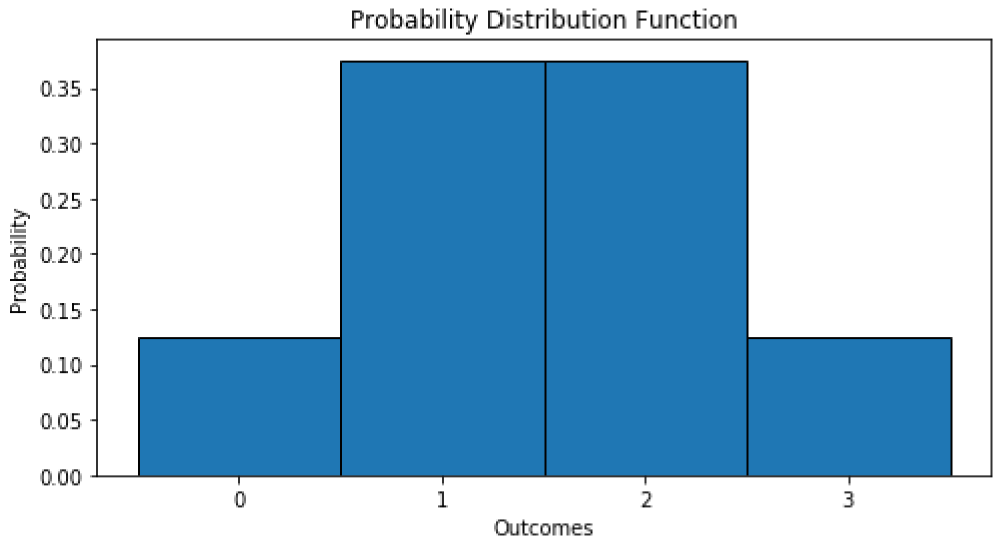
$$F_X^{-1} \ : \ [0..1) \rightarrow R_X$$

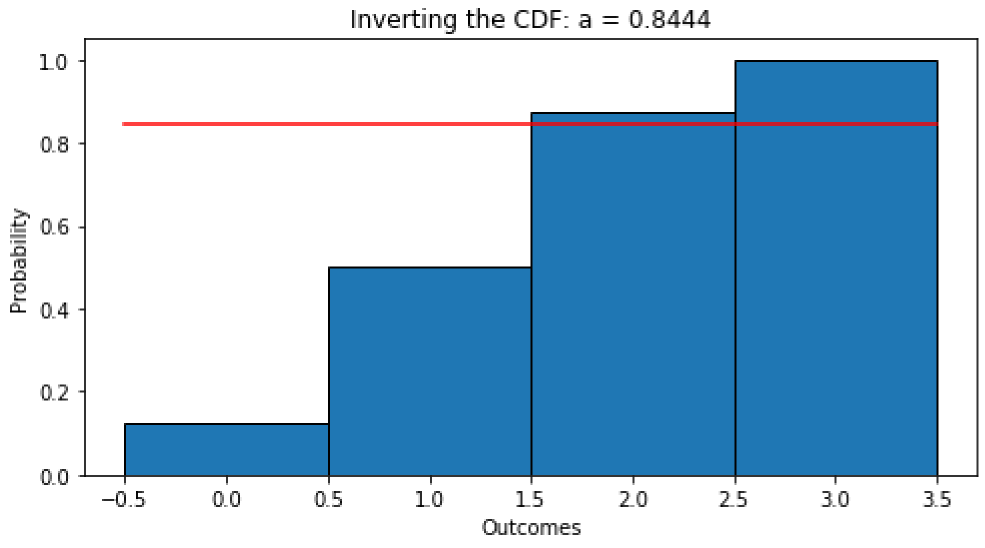
The algorithm for doing this is actually very simple: just generate a random value a in the range $[0..1)$ and look for the first bin which is greater than a; output the corresponding outcome.

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 [10]:

```
Rx10b = [0,1,2,3]
fx10b = [1/8,3/8,3/8,1/8]
Fx10b = [1/8,4/8,7/8,1.0]
plt.figure(figsize=(8, 4))
plt.bar(Rx10b,fx10b,width=1.0,edgecolor='black')
plt.ylabel("Probability")
plt.xlabel("Outcomes")
if (Rx10b[-1] - Rx10b[0] < 30):
    ticks = range(Rx10b[0],Rx10b[-1]+1)
    plt.xticks(ticks, ticks)
plt.title("Probability Distribution Function")
plt.show()

plt.figure(figsize=(8, 4))
plt.bar(Rx10b,Fx10b,width=1.0,edgecolor='black')
a = random()
plt.plot([-0.5,3.5],[a,a],color="red")
plt.ylabel("Probability")
plt.xlabel("Outcomes")
plt.title("Inverting the CDF: a = " + str(round4(a)))
plt.show()
```





TODO for 10 (b):

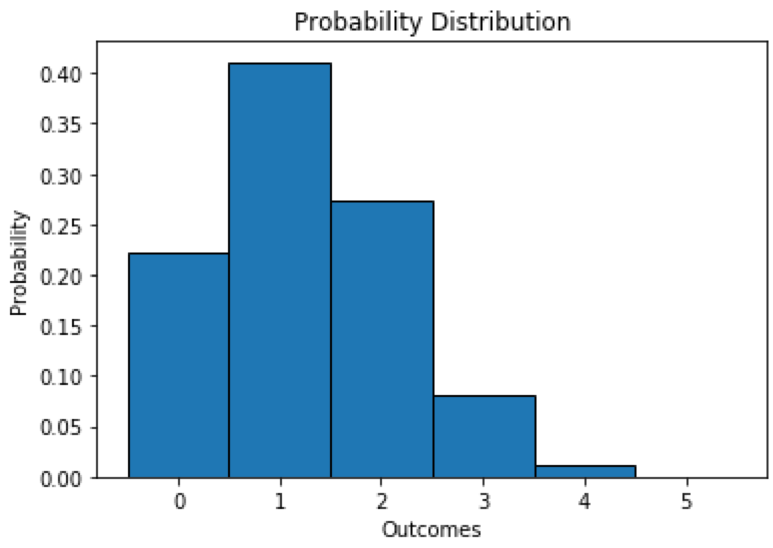
Complete the following code template to generate random variates for a given random variable (represented by Rx and fx), using the code from Part (a).

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

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

In [11]:

```
def rvs(Rx,fx):
    newFx = CDF(fx)
    ranNumber = random()
    for x in range(len(newFx)):
        if ranNumber < newFx[x]:
            return Rx[x]

seed(0)

num_trials = 10**5
new_fx = []
for x in range(num_trials):
    number = rvs(Rx2,fx2)
    new_fx.append(number)
show_distribution(new_fx)
```



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

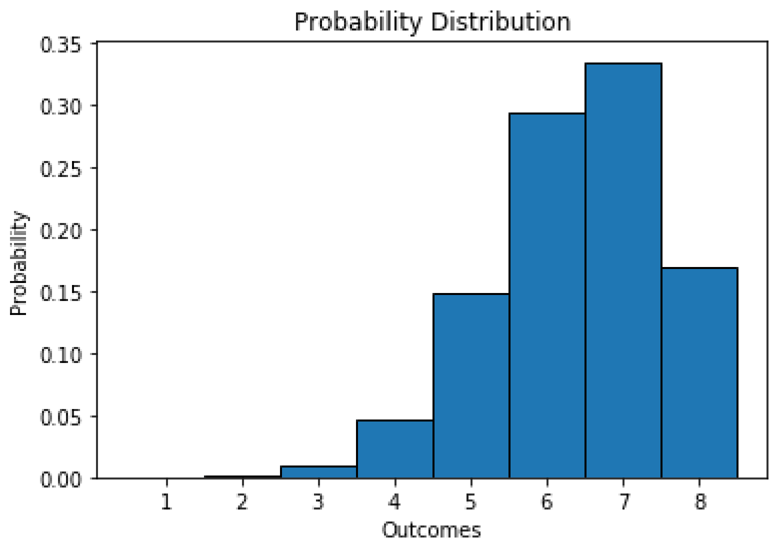
Display the result of generating 10^5 random variates from the Binomial Distribution with $N = 8$ and $p = 0.8$ using `show_distribution`.

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 [12]:

```
# Now generate the empirical distribution

N = 8
p = 0.8

num_trials = 10**5

Rx11a = [0,1,2,3,4,5,6,7,8]

fx11a = [0.00000256, 0.00008192, 0.0011, 0.0092, 0.0459, 0.1468, 0.2936, 0.3355, 0.1

new_fx = []
for x in range(num_trials):
    number = rvs(Rx11a,fx11a)
    new_fx.append(number)

print("Solution:")
show_distribution(new_fx)
```

```
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.

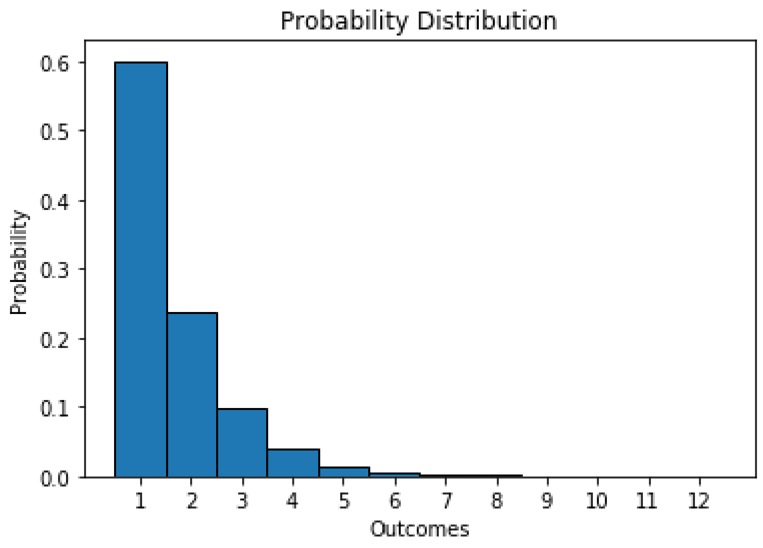
Display the experimental distribution for the Geometric Distribution with $p = 0.6$, using `show_distribution` from the beginning of the notebook, for 10^5 trials and the given N and p.

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 [13]:

```
# Solution

limit = 20
p = 0.6

Rx11b = list(range(limit+1))

fx11b = [0]
value = 0.6
for x in range(limit):
    fx11b.append(value)
    value = value * 0.4

num_trials = 10**5

seed(0)

new_fx = []
for x in range(num_trials):
    number = rvs(Rx11b,fx11b)
    new_fx.append(number)

print("Solution:")
show_distribution(new_fx)
```

```
Solution:
```



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

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

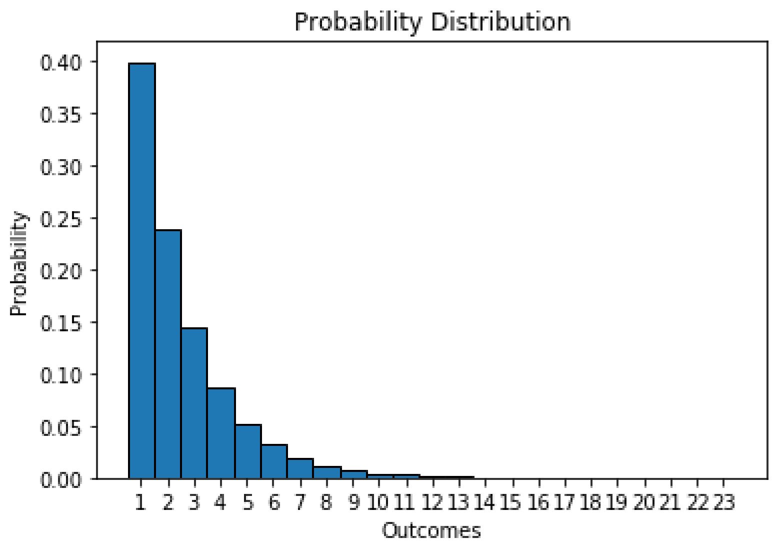
is an integer which is distributed according to the Geometric Distribution with probability p.

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 [14]:

```
p = 0.4

num_trials = 10**5

seed(0)
lst = []
for x in range(num_trials):
    value = random()
    value = 1+ log(value)/log(1-p)
    value = int(value)
    lst.append(value)

show_distribution(lst)
```


Probability Distribution
Probability
Outcomes