V. Rego

Th, March 24

Three examples that use function “random()” from random.py to simulate some situation.

1. simple coin-tossing

2. How a drunken moves on the x-y plane, one random step at a time

3. How Larry takes $n to play a slot machine and loses all his money

```python
# 1.py
# Lets review the function "random()" from the module random.py

# When you call the function random(), it returns a number that lies in the interval [0,1), i.e.,
# it will be 0 or positive, but always less than 1. This number is randomly
# chosen, and every
# number in this interval has the *same* chance of being chosen. That is why
# this function is
# also called a "uniformly-random " random number generator.

# if you want to write a program that will toss a coin for you as many times
# as you want,
# random() is a good function to use.

# Suppose your coin has a probability p of landing heads when you toss it,
# for 0 < p < 1.
# [Usually p is close to 0.5, since most coins are approximately fair; but
# by letting p be any value
# we can ask the program to toss any kind of coin].

from random import *  # in this way we don't have to type "random.random()
 "and can
                # simply call it by typing "random()"

def toss(p):  # this function tosses this coin once and returns True if
               # heads and False if tails
    if (random() < p):
        return True
    else:
        return False

def Geometric(p):  # This tosses repeatedly until we get heads for the first
time. This random number # of tosses until the first heads is called a Geometric random variable # This function will return one instance of this random variable, as if you sat and # did the coin tossing experiment yourself

\[ k = 0 \]

\[ \text{done} = \text{False} \]

\[ \text{while } (\text{done} == \text{False}): \]

\[ k = k + 1 \] # update the toss count

\[ \text{done} = \text{toss}(p) \] # toss the coin

\[ \text{return}(k) \] # when we exit the loop, return the toss count \( k \)

# Now supposing we want to toss the coin \( n \) times and list all \( n \) toss-counts

\[ \text{def main}(): \]

\[ \text{seed}(1234567) \] # don't remove this line. It will set things up for the function \text{random}() # so that every time the program runs it will generate the same stream of random numbers. In this way, different people using this same seed numbers, and thus the same result # will get the same stream of random numbers. # if we remove this line, then different people/runs will give different results # and we will not have control of the experiment

\[ n = 10 \] # suppose we want to get 10 realizations of this geometric random variable

\[ p = 0.1 \] # suppose probability of heads is 0.1; that means probability of tails is 0.9

\[ \text{count} = 0 \]

\[ \text{while} (\text{count} < n): \] # we use " < n " because we start counting at 0
print("Tossing experiment ",count," # of tosses to get heads:",Geometric(p))

count = count + 1

#___________________________________________________________________________
#2.py

# Let's now use random() to decide how something moves in some space.

# To keep things simple, let the space be the x-y plane.

# Now let's make a small story around this problem. Larry has too much to
drink at a party,
# and his host's house is at point (a,b) on the x-y plane. He always moves
# from one such point
# to a point one-unit (one step) away: North, South, East or West.

# He wanders around late at night looking for his home, and cannot remember
where he lives.

# The police will pick him up the instant he steps outside a square defined
by the four
# points: (n,n), (-n,-n), (-n,n),(n,-n). So if his position at any time is
(x,y), he gets
# nabbed if x <= -n, x >= n, y <= -n, y>= n.

# How many steps does he take before the police get him?

# How does he move? Each time he moves he takes one step. With equal
probability he moves
# 1 step N, or 1 step S, or 1 step E or 1 step W. We'll make a rule that
when we consult probabilities
# we will do it in the N, S, E, W order so that all of us do the same thing
and thus will get the same
# result.

# Rules: if random() falls in the interval

# [0,0.25) then N
# [0.25,0.5) then S
# [0.5,0.75) then E
# [0.75,1) then S

# The probability of moving in any direction is 0.25 (i.e., equal probability)
from random import *

def checkifnabbed(x, y, n): # tells if the police nab Larry
    if (x <= -n or x >= n or y <= -n or y >= n): return True
    else: return False

    # in your sample question set, we had to check if a frog landed on the
    # boundary of or outside
    # a circle of radius r in order to stop the frog jumps; for Larry it's a
    # square

def direction(): # return N, S, E or W with probability 0.25 each; the order
    # in which we check
    # is important
    if (0 <= random() < 0.25): return "N"
    elif (0.25 <= random() < 0.5): return "S"
    elif (0.5 <= random() < 0.75): return "E"
    else: return "W"

def main():
    seed(1234567) # do not remove; initializes the random number
    stream seed
    # if you remove/change the seed you'll get different results; try
    # another number to see for yourself

    a, b = eval(input("Enter address of Larry's host: a, b: "))

    n = eval(input("Enter size of square, i.e., n: "))

    x = a
    y = b

    steps = 0

    while (checkifnabbed(x, y, n) == False):
        which = direction() # note that we call direction just once to get
        # one move
        if (which == "N"): y = y + 1
        elif (which == "S"): y = y - 1
        elif (which == "E"): x = x - 1
        else: x = x + 1
steps = steps + 1

print("Number of steps before Larry is nabbed: ", steps)  # print as soon as we exit the loop

#_________________________________________________________________________
#3.py
# Let's try using random() to see how Larry does at gambling.
# To play a slot machine Larry has to pay $2 for each try.
# At this slot machine Larry wins with probability p (and thus loses with probability 1 - p), 0 < p < 1
# If Larry wins, the machine pays him $4 (which means he wins $2).
# If Larry loses, the machine pays him nothing (which means he loses $2).
# How many tries does it take Larry to lose all his money? He WILL go broke for sure.
# We'll assume that Larry starts with $n, for $n >= 2.
from random import *

def winorlose(p):  # just like tossing a coin, only this time it's the slot machine
    if (random() < p): return ("win")  # call random() to decide win or lose, just like heads or tails
    else: return ("lose")

def number_of_steps_to_ruin(n,p):  # start out with #n >= 2 dollars, return the number of steps until Larry is ruined (i.e., he goes broke)
    # p is the probability of a win on any given try
    steps = 0
    while (n > 0):  # n > 0 means Larry is not yet broke; initially n >= 2
n = n - 2  # Larry pays the slot machine $2 so he can have one try

steps = steps + 1  # increment the number of times he plays by 1

result = winorlose(p)  # Larry pulls the arm on the slot machine; he wins or loses

if (result == "win"): n = n + 4  # if he wins he gets $4 (really he paid $2 to play; the machine and also gives him a $2 prize.

                   # returns his $2

# if he loses he gets $0, so there's no need of an "else" clause

# after some number of tries Larry will go broke because n will reach 0; loop is exited

return(steps)  # return the number of steps to the caller

print(" Number of tries until Larry is ruined = ",steps)

def main():

    seed(7654321)  # do not remove this line; it initializes the random number seed

    n = eval(input("Enter the amount of money in dollars that Larry takes to the slot machine: "))

    # Use values of p less than or equal to 0.5

    p = eval(input("Enter the probability that the slot machine allows a win on any try: "))

    count = number_of_steps_to_ruin(n, p)

    print(" Number of tries until Larry is ruined = ",count)

main()