Wharf width 7: Probability 0.2455855988604828 TOTAL

Wharf width 9: 0.4429678848283499 TOTAL

Wharf length 17: 0.5230399079449762 TOTAL

Each time it is tested, the first two decimal places are always constant.

ended up doing the anchor head challenge first, here is my code:

from random import randint  
location = 4  
distance = 0  
reached = 0  
fall = 0  
list =[]  
while 1 <= location <= 7 and distance < 21:  
 n = randint(0,1)  
 distance += 1  
 list.append(distance)  
 if n == 0:  
 list.append('Left')  
 location -= 1  
 list.append(location)  
 if n == 1:  
 list.append('Right')  
 location += 1  
 list.append(location)  
 if location < 1 or location > 7 and distance < 21:  
 location = 4  
 distance = 0  
 list.append("FALL")  
 fall += 1  
reached += 1  
print (reached/(reached+fall), "is the probability of success.")  
print(list)  
print (list.count("FALL"))

And here is my output for the probability:  
0.08333333333333333 is the probability of success.  
[1, 'Right', 5, 2, 'Left', 4, 3, 'Left', 3, 4, 'Right', 4, 5, 'Left', 3, 6, 'Right', 4, 7, 'Left', 3, 8, 'Right', 4, 9, 'Right', 5, 10, 'Right', 6, 11, 'Right', 7, 12, 'Left', 6, 13, 'Left', 5, 14, 'Left', 4, 15, 'Left', 3, 16, 'Left', 2, 17, 'Left', 1, 18, 'Left', 0, 'FALL', 1, 'Left', 3, 2, 'Right', 4, 3, 'Left', 3, 4, 'Left', 2, 5, 'Left', 1, 6, 'Left', 0, 'FALL', 1, 'Right', 5, 2, 'Left', 4, 3, 'Left', 3, 4, 'Left', 2, 5, 'Left', 1, 6, 'Left', 0, 'FALL', 1, 'Left', 3, 2, 'Right', 4, 3, 'Left', 3, 4, 'Left', 2, 5, 'Right', 3, 6, 'Left', 2, 7, 'Left', 1, 8, 'Left', 0, 'FALL', 1, 'Left', 3, 2, 'Left', 2, 3, 'Left', 1, 4, 'Right', 2, 5, 'Right', 3, 6, 'Left', 2, 7, 'Right', 3, 8, 'Left', 2, 9, 'Left', 1, 10, 'Left', 0, 'FALL', 1, 'Right', 5, 2, 'Right', 6, 3, 'Left', 5, 4, 'Right', 6, 5, 'Right', 7, 6, 'Right', 8, 'FALL', 1, 'Right', 5, 2, 'Right', 6, 3, 'Right', 7, 4, 'Right', 8, 'FALL', 1, 'Left', 3, 2, 'Left', 2, 3, 'Right', 3, 4, 'Left', 2, 5, 'Left', 1, 6, 'Right', 2, 7, 'Right', 3, 8, 'Left', 2, 9, 'Left', 1, 10, 'Left', 0, 'FALL', 1, 'Left', 3, 2, 'Left', 2, 3, 'Right', 3, 4, 'Left', 2, 5, 'Left', 1, 6, 'Left', 0, 'FALL', 1, 'Left', 3, 2, 'Left', 2, 3, 'Left', 1, 4, 'Right', 2, 5, 'Left', 1, 6, 'Left', 0, 'FALL', 1, 'Right', 5, 2, 'Right', 6, 3, 'Left', 5, 4, 'Left', 4, 5, 'Left', 3, 6, 'Left', 2, 7, 'Right', 3, 8, 'Left', 2, 9, 'Left', 1, 10, 'Right', 2, 11, 'Left', 1, 12, 'Right', 2, 13, 'Right', 3, 14, 'Left', 2, 15, 'Left', 1, 16, 'Left', 0, 'FALL', 1, 'Right', 5, 2, 'Right', 6, 3, 'Left', 5, 4, 'Right', 6, 5, 'Left', 5, 6, 'Left', 4, 7, 'Right', 5, 8, 'Left', 4, 9, 'Right', 5, 10, 'Right', 6, 11, 'Left', 5, 12, 'Right', 6, 13, 'Left', 5, 14, 'Left', 4, 15, 'Left', 3, 16, 'Right', 4, 17, 'Right', 5, 18, 'Left', 4, 19, 'Right', 5, 20, 'Right', 6, 21, 'Left', 5]  
11I am currently simulating this program using turtle graphics... do you think that is a good idea?

**Saturday, February 6th**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [11:17 AM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612639076001300)

great idea! I would be interested in seeing that!

**Sunday, February 7th**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [7:08 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612753694003500)

Hi, I just finished using pygame, and here it is:

import pygame,time  
from random import randint  
from pygame.locals import \*  
import sys  
pygame.init()  
screen = pygame.display.set\_mode((600,600))  
pygame.display.set\_caption("Drawing Rectangles")  
pos\_x = 237  
pos\_y = 250pos\_y2 = 610  
pos\_x2 = pos\_x + 63.5  
color = 255, 255, 0  
color1= 255,100,10  
width = 0 # solid fill  
pos = pos\_x, pos\_y, 137, 441  
pos1 = pos\_x2,pos\_y2,10,10  
t=0  
distance =0  
fail=0  
pygame.draw.rect(screen, color, pos, width)  
pygame.draw.rect(screen, color1, pos1, width)  
pygame.display.update()while (pos\_y2 >= 250):  
 print(pos1)  
 pygame.draw.rect(screen, color1, pos1, width)  
 pygame.display.update()  
 pos\_y2 = pos\_y2 - 21  
 pos1= pos\_x2,pos\_y2,10,10  
 print("Moving forward",pos1)  
 for event in pygame.event.get():  
 if event.type in (QUIT, KEYDOWN):  
 sys.exit()  
 screen.fill((0,0,200))  
 pygame.draw.rect(screen, color, pos, width)  
 pygame.draw.rect(screen,color1,pos1,width)  
 pygame.display.update()  
 n = randint(0, 1)  
 print("The random generated number is ", n)  
 if(n == 0):  
 pos\_x2 -= 21  
 pos1 = pos\_x2, pos\_y2, 10, 10  
 pygame.draw.rect(screen, color1, pos1, width)  
 time.sleep(3)  
 else:  
 print("moving right")  
 pos\_x2 += 21  
 pos1 = pos\_x2, pos\_y2, 10, 10  
 pygame.draw.rect(screen, color1, pos1, width)  
 time.sleep(3) if(pos\_x2 <= 216.5 or pos\_x2 >= 384.5):  
 print("FAILURE")  
 pos\_y2 = 610  
 pos\_x2 = 300.5  
 fail+=1print("SUCCESS ")  
print("odds are ",1/(1+(fail)))

Currently, I am using the case that the x coordinate for the "human" (small rectangle) has to be less than or equal to 216.5 or greater than or equal to 384.5, but I was about to use just greater than and less than without the equal to s, so that it could be demonstrated that she actually fell into the water, but when doing that I had trouble resetting the positions, because the y coordinates were always changing.

**Monday, February 8th**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [12:25 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612815915009700)

ok i think you have the opportunity here to "re-factor" your code (which just means write it a bit differently so it works the way you want). You don't **have** to but it wouldn't hurt. The value you get out of it is seeing how refining code has value.As an example there is the arithmetic of the problem (wharf is 7 x 21) and the arithmetic of the pygame screen. You have written this so that your sailor is walking in pygame arithmetic rather than steps arithmetic. If you re-did this I would suggest having the sailor walk in natural wharf coordinates so x is a number from 0 to 8 and y is a number from 0 to 21; and then convert that when you actually need to draw the animation. You would add a constant offset and then scale by pixels per unit. I think that one sailor step is 21 pixels for example. I also think that instead of writing "FAILURE" you should write "SPLASH! Fell off the side of the wharf!"I changed 3 seconds to a variable "sleeptime" so I could set that to zero and watch the trials go faster.Finally I like how you calculated the odds as (1 - in - number - of - tries). However each time I run the program I get a different result. Once I got odds of 0.5 because I got lucky and made it across on the second try. Once I got 0.05 or so... lots of falling in the water. You might think about how you could run the program to get an average probability of the sailor making it back to her boat.

**Wednesday, February 10th**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [7:46 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613015180000800)

I am working on refactoring the code, and I will have it done by Saturday...



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [8:28 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613017681002700)

ok. I have been working on the bugs problem (you may think it is all solved but there are still questions!) and I know you are working on AnchorHead. That means you still have Egon left; that will be I hope a very fun one for you.

**Yesterday**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [7:25 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613273141001500)

Here is the code for accurately calculating the average (without pygame):

from random import randint  
denominator = 0  
numerator = 0  
for i in range(10000):  
 location = 4  
 distance = 0  
 reached = 0  
 fall = 0  
 list =[]  
 while distance < 17:  
 n = randint(0,1)  
 distance += 1  
 list.append(distance)  
 if n == 0:  
 list.append('Left')  
 location -= 1  
 list.append(location)  
 if n == 1:  
 list.append('Right')  
 location += 1  
 list.append(location)  
 if location < 1 or location > 9 and distance < 17:  
 location = 4  
 distance = 0  
 list.append("FALL")  
 fall += 1  
 reached += 1  
 print(reached/(reached+fall), "is the probability of success.")  
 #print(list)  
 #print(list.count("FALL"))  
 numerator += reached  
 denominator += (reached+fall)  
average = numerator/denominator  
print (average, "TOTAL")

And these are the outputs:  
Wharf width 7: Probability 0.2455855988604828 TOTAL  
Wharf width 9: 0.4429678848283499 TOTAL  
Wharf length 17: 0.5230399079449762 TOTALEach time it is tested, the first two decimal places are always constant.Here is the pygame code, refactored, not in a for loop, because it would take too much time:

import pygame  
from random import randint  
import time  
from pygame.locals import \*  
for i in range (1000):  
 location = 4  
 distance = 0  
 reached = 0  
 fall = 0  
 pygame.init()  
 screen = pygame.display.set\_mode((600,600))  
 pygame.display.set\_caption("Drawing Rectangles")  
 pos\_x = 237  
 pos\_y = 159  
 pos\_y2 = 590  
 pos\_x2 = 300.5  
 color = 255, 255, 0  
 color1 = 115, 0, 10  
 width = 0 # solid fill  
 pos = pos\_x, pos\_y, 137, 441  
 pos1 = pos\_x2, pos\_y2, 10, 10 def move\_on\_screen(coordinate, direction = 0):  
 global pos1  
 global pos\_y2  
 global pos\_x2  
 pygame.draw.rect(screen, color, pos, width)  
 pygame.draw.rect(screen, color1, pos1, width)  
 pygame.display.update ()  
 time.sleep(1)  
 if coordinate == "y":  
 pos\_y2 -= 21  
 pos1 = pos\_x2, pos\_y2, 10, 10  
 else:  
 if direction == 0:  
 pos\_x2 -= 21  
 pos1 = pos\_x2, pos\_y2, 10, 10  
 else:  
 pos\_x2 += 21  
 pos1 = pos\_x2, pos\_y2, 10, 10  
 pygame.draw.rect(screen, color1, pos1, width)  
 pygame.display.update() while distance < 21:  
 for event in pygame.event.get():  
 if event.type in (QUIT, KEYDOWN):  
 sys.exit()  
 screen.fill((0, 0, 200))  
 distance += 1  
 print(distance)  
 move\_on\_screen("y")  
 pygame.display.update()  
 n = randint(0, 1)  
 print("Random generated number is ",n)  
 if n == 0:  
 location -= 1  
 elif n == 1:  
 location += 1  
 move\_on\_screen("x", direction = n)  
 if location < 1 or location > 7:  
 location = 4  
 distance = 0  
 fall += 1  
 pos\_x2= 300.5  
 pos\_y2= 590 reached += 1  
 print (reached/(reached+fall), "is the probability of success.")



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [7:36 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613273808002600)

Also, for the Egon problem, Egon can choose any random point, including inside the triangle, and any of the points that make up the triangle, right?

**Today**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [10:26 AM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613327181001700)

nope! you are choosing a paint can location at random, not a point in the triangle.

[10:27](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613327258002800)

so only three possibilities for each time you choose.

[10:31](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1613327468004900)

nice job on the anchor head problem. how wide must the wharf be to reach probability 1.0?