**[Rob Fatland](https://app.slack.com/team/UN804U79N" \t "_blank)**  [8:15 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1610770500001800)

Hi [@Ridhi Gundapuneni](https://pythonbyte.slack.com/team/U01CWCNQ0AC). The snake game has some good logic tasks involved. I never coded it up myself but I'm sure it will be fun to do. Are you all set for challenges or are you interested in some other ideas? Let me know; I have plenty of suggestions.



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [9:21 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1610774473004400)

Please let me know if you have any other ideas; I haven’t decided on it yet. Are there any new routes in the mocean game?



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [9:42 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1610775721005800)

I'll let you know: There should be more routes this weekend if everything goes well. Let me know what you think of the challenge [here](https://github.com/robfatland/pythonbytes/blob/master/projects/bugs/README.md). Also you might take a look at the **Egon** problem on [this page](https://github.com/robfatland/pythonbytes/blob/master/projects/bugs/README.md). You have to scroll down a bit.

[**projects/bugs/README.md**](https://github.com/robfatland/pythonbytes/blob/master/projects/bugs/README.md)

# PythonBytes Project In-Depth  
  
## Four Bugs  
  
You do not have to use graphics for this project; but you certainly can! I used turtles to represent the bugs.

Show more

[robfatland/pythonbytes](https://github.com/robfatland/pythonbytes) | Added by [GitHub](https://pythonbyte.slack.com/services/B01CPLC8925)

**Tuesday, January 26th**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [1:55 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611698107001500)

Hi, I finished the bug program for the square, and I am working on the other polygon shapes. Here is the code, for calculating the distance from the starting point to the point where all the bugs meet:

import turtle  
a = turtle.Turtle()  
b = turtle.Turtle()  
c = turtle.Turtle()  
d = turtle.Turtle()a.penup()  
a.goto (200, 200)  
a.pendown()b.penup()  
b.goto(200, -200)  
b.pendown()c.penup()  
c.goto(-200, -200)  
c.pendown()d.penup()  
d.goto(-200, 200)  
d.pendown()screen = turtle.getscreen()  
screen.bgcolor ("black")#a.shape ('turtle')  
a.pencolor ("red")  
a.color ("red")  
a.write("Alpher", False, 'left', ('Times new roman', 8, 'normal'))#b.shape ('turtle')  
b.pencolor ("orange")  
b.color ("orange")  
b.write("Bethe", False, 'left', ('Times new roman', 8, 'normal'))#c.shape ('turtle')  
c.pencolor ("yellow")  
c.color ("yellow")  
c.write("Gamow", False, 'left', ('Times new roman', 8, 'normal'))#d.shape ('turtle')  
d.pencolor ("green")  
d.color ("green")  
d.write("Dyson", False, 'left', ('Times new roman', 8, 'normal'))a.setheading (a.towards (b))  
b.setheading (b.towards (c))  
c.setheading (c.towards (d))  
d.setheading (d.towards (a))print (a.towards (b))  
print (b.towards (c))  
print (c.towards (d))  
print (d.towards (a))distance = a.distance(b)  
da = 0  
db = 0  
dc = 0  
dd = 0  
while distance > 0.01:  
 step = 10  
 if distance <= 10:  
 print ("WE HAVE ENTERED THE IF")  
 step = 0.01  
 #a.write(distance, False, 'left', ('Times new roman', 8, 'normal'))  
 a.forward (step)  
 b.forward (step)  
 c.forward(step)  
 d.forward(step)  
 da += step  
 '''db += 10  
 dc += 10  
 dd += 10'''  
 #print(da) a.setheading(a.towards(b))  
 b.setheading(b.towards(c))  
 c.setheading(c.towards(d))  
 d.setheading(d.towards(a))  
 distance = a.distance(b)  
 print(distance)print("The final distance is",da)



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [2:02 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611698531002000)

This is the final output for the distance:  
Final distance is  460.07999999999083



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [8:08 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611720483008400)

Hi Ridhi!**WE HAVE ENTERED THE IF** is such a beautiful expression. Thank you!I have not had time to improve ***Mocean*** but I have been thinking about it.Your program worked perfectly, nice job.Meanwhile you arrived at the answer '460' so my question for you is: Is 460 the correct answer to the problem? Your step size is "10". You could make the steps smaller or larger if you like.Since it is  very time-consuming for the "IF" part to finish: I suggest inside the if statement setting step = 0.1 as faster than step = 0.01.Challenge for Ridhi: Establish whether or not the answer depends on step ; and if so: Why? Bonus question: Does the program get more accurate or less accurate if you make step smaller? Bonus bonus question: Is there a correct final answer; and if so what is it?Also one more thing: I modified your program a little bit so I could read it and run it more easily. If you like I'll paste it here.



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [8:44 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611722643008600)

**STEP**  
 **DISTANCE**  
   **0.5**  
 401  
   **1**  
 402  
   **3**  
 407  
   **5**  
 413  
   **7**  
 421  
   **9**  
 432  
   **10**  
 460So, I got these results when testing the different step lengths and the distances. I know that the speed is the one that is changing here… and it is equal to distance divided by time, but in this case, instead of the distance changing with speed, the time should be changing, and the distance should be constant, shouldn’t it? I am also guessing that the correct distance is 400, or very close to that, based on the tests above.



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [8:54 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611723273015700)

Your guess is correct; now the question is: What is the mathematical reasoning for **why** it is correct?I agree that "speed" versus "step size" can be confusing. However if you just focus on step size (and don't worry about speed) then you can stay focused on the question: How far does each bug travel. You could slow them down or speed them up if you like, but it is the step size that changes the total path length, not speed. The path length is a distance and it does not have a "time" in it. Of course fast bugs with step size 1 will arrive at the center sooner than slow bugs with the same step size of 1; but they will travel the same path and therefore cover the same distance.So back to the question of distance: It is worth considering why smaller steps give smaller total distance. I think it has something to do with the bugs all moving at the same time. If you take a gigantic step towards a bug while it is moving: You started out pointing in the correct direction but by the time you reach the end of that straight step the bug has moved a bit to your right. Does this make sense?

**Wednesday, January 27th**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [6:31 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611801083001200)

[@Ridhi Gundapuneni](https://pythonbyte.slack.com/team/U01CWCNQ0AC) you are so far ahead you may be somewhat bored by today's Zoom topic. If so: Feel free to work on your own stuff, even if you choose to stay on the call -- which is fine of course.

**Thursday, January 28th**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [2:48 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611874125000500)

[@Ridhi Gundapuneni](https://pythonbyte.slack.com/team/U01CWCNQ0AC) do you have an explanation that you like: For why smaller step sizes change the answer? (edited)



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [3:11 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611875461003800)

Yes. For example, if an adult and his child were taking a walk, the adult's steps would be larger compared to that of the child's. So in a given time the distance traveled in the same amount of time would be different for the adult, and different for the child. The adult would have traveled more in the same time period and the child would have traveled less. I think this is what is happening with the different bug step sizes.



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [5:52 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611885135002100)

ok but consider an adult who takes very large steps but (so the child can keep up) waits for a bit after each one. Now they are both going at the same speed... but the child takes more, smaller steps. So it is still a question: Who measures the bug's distance more accurately, Big Steps or Small Steps?



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [9:56 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611899777003600)

I think that small steps will measure the bugs’ distance more accurately because due to the sample size being smaller, that will give a more accurate average.

[9:57](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611899823004300)

Please let me know if I am missing anything.



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [10:07 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1611900471014800)

You are on the right path. (That's a little joke for you.) Here is one way to think of it that may help. I describe how I see it and you see if you like the explanation. It will take a bit of thinking and you may wish to discuss it with others in your household.Let us suppose that you are A and you are looking at B. And you are about to take a step towards B. You very carefully set the direction of your next step: Towards B where B is right now.Now at the same moment B takes a step towards C. So when your foot lands you notice that B has moved off of your step direction, just a little bit. That is: You went in approximately the right direction but because B was moving -- to be perfectly accurate -- your step should have curved ever so slightly to the right. But it did not. It was straight.No worries. You keep going!You fix your direction to be towards B for your next step... and the same thing happens. B moves a little bit so you wind up going in the direction that B used to be in. In fact every time you take a step: B moves so your step direction was correct at the beginning but a little bit off by the time the step was done.So you think "Aha I will take tiny steps!" Now the same thing happens: You take your tiny step but B moves a tiny bit also. So your tiny step was more accurate than before... because B (who is also now taking tiny steps) did not move as far. So now your tiny steps are just just a little bit inaccurate.The smaller you make your steps, the more closely you will follow the perfect path, of a vast number of super small steps.Now let's put you and the other bugs on unicycles! As you travel forward you can keep smoothly changing the direction of your motion to always keep bug B directly in your sights. As a result: You path will be a smooth curve, just as if you had taken an enormous number of teensy weensy steps.In Python what you did was simulate this process with straight line segments for each of your steps. So your answer is always a little bit too large; because your step path only approximates the smooth curve of the unicycle.This idea, by the way, is at the heart of a subject in mathematics called calculus. So if some day you find yourself learning calculus: Remember the little bugs!By the way: I have described the effect of your step size on your answer but I have not yet explained why the answer is the same distance as the edge of the table. That is still an unsolved problem in our conversation. (edited)

**Saturday, January 30th**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [5:28 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612056510002100)

Thank you for your explanation. I have completed the program to work for any polygon, the code is below:

import turtle  
from random import \*  
alphabet = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]  
l = []def set\_direction(n):  
 for i in range(n):  
 if i == n:  
 l[i].setheading(l[i].towards(l[0]))  
 else:  
 l[i-1].setheading(l[i - 1].towards(l[i]))def forward(n,step):  
 for i in range (n):  
 l[i].forward(step)def setturt(n):  
 for i in range(n):  
 R = random()  
 G = random()  
 B = random()  
 l.append(turtle.Turtle())  
 l[i].pencolor(R, G, B)  
 l[i].write(alphabet[i], False, 'left', ('Times new roman', 8, 'normal'))def drawpolygon(n):  
 for i in range (n):  
 l[i].penup()  
 for j in range(i):  
 l[i].forward(100)  
 l[i].left(360/n)  
 l[i].pendown()def main():  
 totaldistance = 0  
 n2 = eval(input("Please enter the number of sides you want your polygon to have: "))  
 setturt(n2)  
 drawpolygon(n2) distance = l[0].distance(l[1])  
 '''print("distance between l[2] and l[1]",l[2].distance(l[1]))  
 print("distance between l[0] and l[1]",distance)  
 print("distance between l[0] and l[2]",l[2].distance(l[0]))''' while distance > 0.1:  
 step = 0.5  
 if distance <= 10:  
 print ("WE HAVE ENTERED THE IF")  
 step = 0.1  
 forward(n2, step)  
 totaldistance += step  
 set\_direction(n2)  
 distance = l[0].distance(l[1])  
 print(distance)  
 print (totaldistance, "is the total distance.")  
main()  
screen = turtle.Screen()  
screen.exitonclick()

One thing that I am not sure about, is that the distance when measured, from the vertex to the middle, is about 67, but I know the ideal distance in this case is 100, the side length. I am not sure why this is the case.

**Sunday, January 31st**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [10:15 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612160135018500)

Here is a formula I calculated. We can test it. Distance walked = table edge length divided by (1 + F). I will tell you what F is in just a moment.First: you let me input a number of bugs when the program starts. Let us call that n. I use this number n to calculate F as follows. First I make a fraction (n-2)/n and I multiply this by pi which is about 3.14159.... I take the cosine of this new number. If you are not familiar with cosine: No problem. It is a special function. The main thing to know for the moment is that you feed a number into cosine and it gives back a result between -1 and +1. This number is F. So now we can look at the entire formula.Let's abbreviate the edge of the table as s. In your program you use s = 100.  Let's call the distance one bug walks in an n bug problem: DMy formula is:D = s / (1 + cos(pi \* (n-2)/(n)))I wrote some Python code as follows:

from math import cos, piprint('n-bugs table edge distance bug travels')for n in range(2, 10):  
 s = 100.  
 f = cos(pi\*(n-2)/n)  
 d = s / (1 + f)  
 print(n, ' ', s, ' ', round(d, 2))

When I run this program using your table edge of 100 I get this table of results:

n-bugs table edge distance bug travels  
2 100.0 50.0  
3 100.0 66.67  
4 100.0 100.0  
5 100.0 144.72  
6 100.0 200.0  
7 100.0 265.6  
8 100.0 341.42  
9 100.0 427.43

Notice that when I only have two bugs: They start out 100 units apart and they walk towards one another. So they travel half of 100 or 50. This is what the formula predicts so that's good.When I look at the result for 4 bugs: I get 100. So this means each bug travels exactly the length of the edge of the table. As you have shown this is a spiral path for each bug and they all meet at the center. If they walked along a straight diagonal from corner to center: You are right they would travel 67... but they take the spiral path and this is longer. In fact it is equal to the length of the edge of the table.I ran your program with 7 bugs and got a result of 275. The formula predicts it should be 265.6. I ran your program a second time using a smaller step size to see if it would get closer. I got 273.7 with a step size of 0.2. This is smaller but still not very close to 265.5. I made a fast version of your program by turning off the graphics: screen.tracer(0) and this runs in just a few seconds. I reduced the step size to 0.02 and I got a total distance of... well I will let you try it out. Here is the "Fast Ridhi Code" for n bugs:

import turtle  
from math import pi, coss = 100.  
print()  
print('For this simulation the table edge length is', s)  
print()  
n = int(input('How many bugs (n): '))  
step = float(input('How tiny of a step size: '))  
print()formuladistance = s / (1. + cos(pi\*(n-2)/n))screen = turtle.Screen()  
screen.tracer(0)totaldistance, buglist = 0., []  
modcounter, modblip = 0, int(10/step)for i in range(n):  
 buglist.append(turtle.Turtle())  
 for j in range(i):  
 buglist[i].forward(100)  
 buglist[i].left(360/n)while buglist[0].distance(buglist[1]) > step\*1.5:  
 for i in range(n): buglist[i].forward(step)  
 totaldistance += step  
 for i in range(-1,n-1):  
 buglist[i].setheading(buglist[i].towards(buglist[i+1]))  
 if not modcounter % modblip:  
 print('at', modcounter, 'so far', round(totaldistance, 2))  
 modcounter += 1print()  
print(round(totaldistance, 2), "is the simulation total distance.")  
print()  
print(round(formuladistance, 2), "is the predicted total distance")  
print()

**Wednesday, February 3rd**



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [6:50 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612407059001900)

My computer was glitching, I had to leave the call... when you get a chance, please let me know the other challenges that you were explaining are.



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [9:45 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612417526002400)

will do. I hope your computer feels better soon. Try feeding it some honey, that always cheers me up.

:laughing:**1**

**Thursday, February 4th**



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [12:59 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612472378003100)

Here are the two that are not bugs; let me know if you decide to try one or the other.[This is the link to the Anchor Head coding challenge](https://github.com/robfatland/pythonbytes/tree/master/projects/egon).[This is also the link to the Egon painting coding challenge](https://github.com/robfatland/pythonbytes/tree/master/projects/egon). (You have to scroll down a bit.)



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [2:13 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612476833004500)

Thank you so much! I will start to work on the Egon challenge.



[**Rob Fatland**](https://app.slack.com/team/UN804U79N)  [3:54 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612482842005200)

oh don't forget: you can use screen.tracer(0) to turn off plotting; it will run 100 times faster. Then at the end put screen.tracer(1) to see the results.



[**Ridhi Gundapuneni**](https://app.slack.com/team/U01CWCNQ0AC)  [7:13 PM](https://pythonbyte.slack.com/archives/D01CM31H4S0/p1612494812005500)

Thank you!

**Friday, February 5th**