Frisbee Golf Equations

In this project, you will create a frisbee golf game. The drawing and animation code have already been written for you in the "Frisbee Golf Template.tns" file. You will write the code to generate various forms of equations. You will also write the code to display the equation and calculate the answer. When playing the game, each equation solved correctly will earn you a new frisbee. How skilled are you at Frisbee Golf?

Objectives:

Programming Objectives:

- Use variables to store values
- Use the randint() function to generate random numbers.
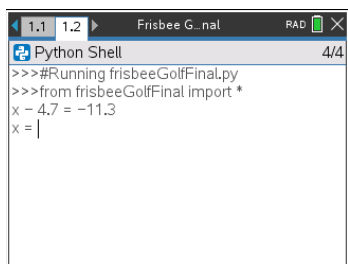- Use the print() function to display
- Use a while loop to repeat code.

Math Objectives:

- Solve one step equations with rational solutions. (May adjust for integers or positive numbers only. See teacher note on step 6)
- Solve multi-step equations with rational solutions. (May skip. See teacher note on step 20.)
- Use substitution to verify a value is a solution to an equation.

Math Course Connections: Middle School Mathematics

In this project, you will create a frisbee golf game. The drawing and animation code have already been written for you in the "Frisbee Golf Template.tns" file. You will write the code to generate various forms of equations. You will also write the code to display the equation and calculate the answer. When playing the game, each equation solved correctly will earn you a new frisbee. How skilled are you at Frisbee Golf?
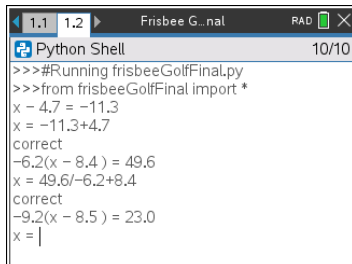
.

Sample Game:

The game asks for the value of x

For each question answered correctly, earn a frisbee.

Press the "t" key to aim and Throw the frisbee.

1. Obtain the "Frisbee Golf Template.tns" from your teacher. A large portion of the code has been prepared for you ahead of time.

2. Let's examine the code.

Libraries needed for the project

Generate two random integers, store in x and n

*You will write lines of code to generate random equations and their answers.

Get user answer. Check to see if it is correct.

Format the number display to ensure place value

Draw the hoop

```
from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

def getFrisbees():
    type = randint(1,2)
    x = randint(-100,100)/10
    n = randint(-100,100)/10
```

```
    a=float(eval(input("x = ")))
    if abs(a - x) <= 0.000001:
        print("correct")
        return 1
    else:
        print("sorry, x = ", x)
    return 0

def disp(val):
    if val > 0:
        return "+ " + p(val)
    return "- " + p(abs(val))

def p(x):
    if(floor((x+0.000001)*100)%10==0):
        x="{:.1f}".format(x)
    else:
        x="{:.2f}".format(x)
    return str(x)

def hoop():
    set_color(0,0,0)
    xc, yc = randint(100,180), randint(30,120)
    draw_arc(xc,yc-5,30,10,0,360)
    for i in range(2,5):
        draw_arc(xc,yc-i*5,30,i*10,180,180)
    fill_rect(xc+13,yc-40,5,20)
```

Draw the person

```
··return xc, yc

def person():
··set_color(0,0,0)
··draw_arc(0,40,20,20,0,360)
··fill_rect(9,40-25,1,25)
··draw_line(0,0,9,16)
··draw_line(9,16,18,0)
··draw_line(0,36,9,30)
··draw_line(9,30,20,36)
```

Draw the frisbee

Animate the frisbee

```
def frisbee(x,y):
··set_color(0,0,255)
··fill_arc(x,y,10,3,0,360)

def throw(x,y,angle,v,time):
···x = 20+v*cos(radians(angle))*time
··y = -16*(time**2) + v*sin(radians(angle))*time
··return x,y

def setAngle(angle):
··add = 5
```

Animate toss angle until the user presses "t"

```
··set_color(0,0,0)
··draw_line(20,36,30,36)
··while get_key() != "t":
····angle+=add
····if angle >80 or angle < 10:
······add=add*-1
····set_color(0,0,0)
····draw_line(20,36,25+10*cos(radians(angle))
····sleep(.2)
····set_color(255,255,255)
····draw_line(20,36,25+10*cos(radians(angle))
```

```
··set_color(0,0,0)
··draw_line(20,36,25+10*cos(radians(angle)),36
··return angle

def setVelocity(v):
··add = 10
··while get_key() != "t":
····v += add
····set_color(0,0,0)
····fill_rect(30,0,10,30)
····set_color(255,0,255)
```

Change and toss velocity until the user presses "t"

```
····fill_rect(30,0,10,30*v/130)
····sleep(.2)
····if v < 10 or v >= 130:
······add=add*-1
··return v

count = 1

#controls the number of questions
for i in range(0):
··count += getFrisbees()
```

*Repeatedly ask the equation questions. Add earned frisbees. You will eventually change range(0) to range(4).

```
··sleep(2)
print("You earned",count,"frisbees")
sleep(2)

set_window(0,200,0,130)

for i in range(count):
··clear()
··person()
··x,y,time,angle,v = 20,36,0,randint(5,70),60
··xc, yc = hoop()
```

Draw the objects, aim and power the frisbee, toss the frisbee, repeat for each frisbee earned

```
··angle = setAngle(angle)
··v = setVelocity(v)
··while y >=0 and x <= xc:
····x,y = throw(x,y,angle,v,time)
····frisbee(x,y)
····sleep(0.2)
····time+= 0.2
··sleep(2)
```

3. Obtain the file "Frisbee Golf Template.tns" if you don't already have it.

   Execute the code (ctrl r)

   The "aim" angle will automatically increase and decrease.   Press "t" when the desired angle is achieved.

   Now, determine how hard to throw the frisbee. When ready, press "t".

   The frisbee will fly along the given path.
   In this case, the goal was missed.
   If you had earned more frisbees, you would have more shots at the goal.
   .
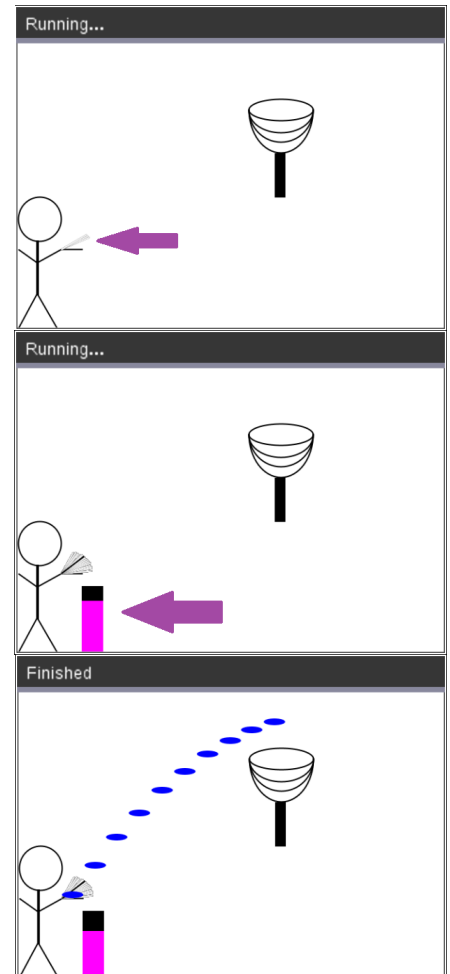
4. Now to earn more frisbees! To earn frisbees, the user must solve equations correctly.
   The first type of equations will look like: x + number = result   or   number + x = result

   Solve the following equations for x.
   a.) x + 8 = 13   b.) x + 1.8 = 5.3   c.) 8.3 + x = 17.5   d.) 2.5 + x = 6.8   e.) x – 5.3 = 7.8

5. A.) Check your answers from step 4. Did you know you can use the scratchpad and the such that key, |, to check your work. To get to the such that key, press [ctrl] [=].

   x + 8 = 13 | x = 7 is false because 7 + 8 = 15 not 13.
   x + 8 = 13 | x = 5 is true because 7 + 5 = 13

   B.) Look back over your work in step 4.   What patterns do you notice?

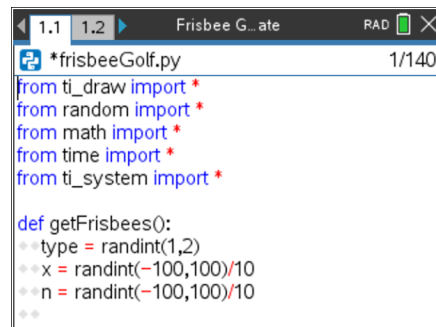# Math Explorations with Python

TI-NSPIRE™ CX II TECHNOLOGY
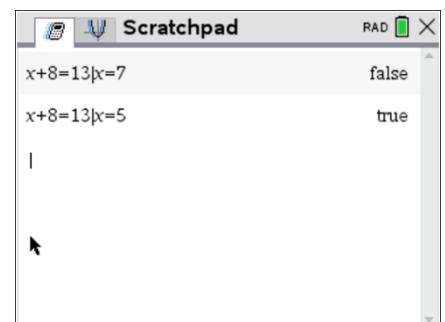
6. This first section of code will generate equations similar to the ones in step #4.

   type = randint(1,2)   will help determine if x comes first or second.

   x = randint(-100,100)/100   randomly creates a rational value for x
   x could be a positive or negative

   n = randint(-100,100)/100   randomly create a rational value to add to x.
   n could be positive or negative

```
1.1  1.2  ▶        Frisbee G…ate    RAD  ▣ ✕
*frisbeeGolf.py                          1/140
from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

def getFrisbees():
  type = randint(1,2)
  x = randint(−100,100)/10
  n = randint(−100,100)/10
```
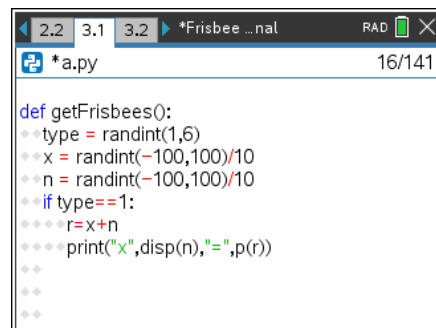
7. To create an equation such as "x + 3.5 = 9.2", add the following three lines of code.

   if type == 1:
     r = x + n
     print("x",disp(n),"=",p(r))

       **if   menu → built-ins → control -→ if
       **print   menu → built-ins →
   Make sure the two lines below the if are indented two spaces (diamonds).
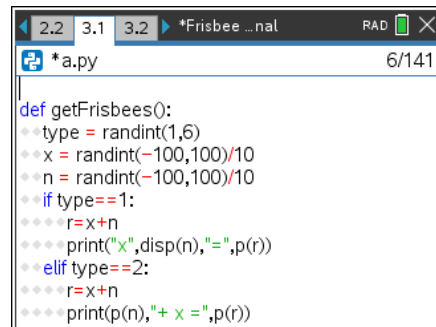
```
◀ 2.2  3.1  3.2  ▶   *Frisbee …nal   RAD ▣ ✕
*a.py                                  16/141

def getFrisbees():
  type = randint(1,6)
  x = randint(−100,100)/10
  n = randint(−100,100)/10
  if type==1:
    r=x+n
    print("x",disp(n),"=",p(r))
```

8. To create an equation such as "7.2 + x = 9.1", add the lines:

   elif type == 2:
     r = x + n
     print(p(n),"+ x =",p(r))

       **elif   menu → built-ins → control -→ elif

   The line print(p(n), "+ x=", p(r)) could be written without the function p(). It would look like print(n,"+ x = ",r). However, sometimes, python stores and prints rational numbers such as 8.2 as 8.19999999999. The function p() ensures this doesn't happen when printed.

```
◀ 2.2  3.1  3.2  ▶   *Frisbee …nal   RAD ▣ ✕
*a.py                                   6/141

def getFrisbees():
  type = randint(1,6)
  x = randint(−100,100)/10
  n = randint(−100,100)/10
  if type==1:
    r=x+n
    print("x",disp(n),"=",p(r))
  elif type==2:
    r=x+n
    print(p(n),"+ x =",p(r))
```

9. The code has already been written to check your answer.
   If you scroll down to about line 40, you'll see this code.

   The line **a=float(eval(input("x = ")))** asks the user enter the answer.
   The eval() around the input lets the user enter a number such as 3.5 or an expressions such as 7.2+5.1.

   The **if(abs(a-x) <= 0.000001):** then checks to see if it correct.

```
◀ 1.1  1.2  ▶        Frisbee G…ate    RAD ▣ ✕
*frisbeeGolf.py                          40/140




    a=float(eval(input("x = ")))
    if abs(a − x) <= 0.000001:
      print("correct")
      return 1
    else:
      print("sorry, x = ", x)
```

10. Currently, the program is set to ask 0 questions.
    If you run the code now, it should behave the same way it did in step 1.

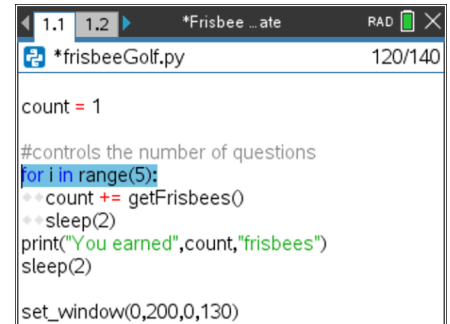   To ensure you haven't made an error, run the code.   [ctrl] [r]

    It shouldn't ask questions yet, but it also shouldn't have any errors.
    If there are errors, check your code with the code in steps 7-8.

11. Now to change the number of questions from 0 to 5.

    Scroll down to the section labeled "controls the number of questions"

    Change the line
        for i in range(0):   to   **for i in range(5):**

```
1.1  1.2 ▶        *Frisbee …ate    RAD 🔋 ✕
🐍 *frisbeeGolf.py                    120/140

count = 1

#controls the number of questions
for i in range(5):
··count += getFrisbees()
··sleep(2)
print("You earned",count,"frisbees")
sleep(2)

set_window(0,200,0,130)
```

12. Execute the code [ctrl] [r].

    The program should ask 5 questions.

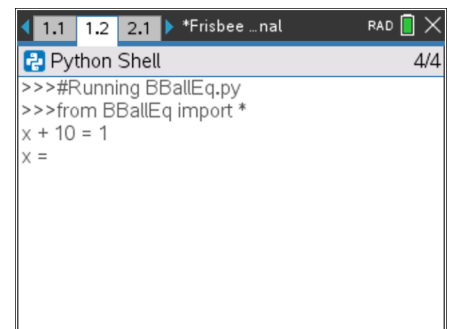   The sample question on the right is "x + 10 = 1"
    The user may either enter **-9** or **1 – 10.**
    The program will count either answer correct.

    Play the game a few times.

    If you answer 3 out of 5 questions correctly, you should get 4 frisbees.
    If you answer 5 out of 5 questions correctly, you should get 6 frisbees.

```
1.1  1.2  2.1 ▶  *Frisbee …nal    RAD 🔋 ✕
🐍 Python Shell                          4/4
>>>#Running BBallEq.py
>>>from BBallEq import *
x + 10 = 1
x =
```
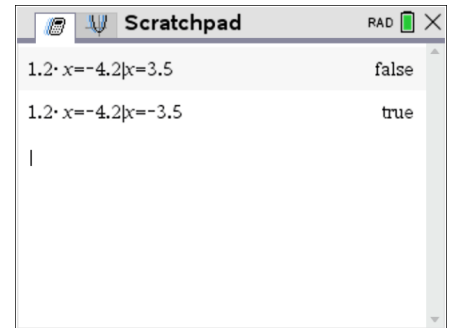
13. Now to program the third type of equation for the game. Solve the following equations:

    a.) $1.2x = -4.2$ b.) $-5.6x = 20.72$ c.) $7.8x = 0$ d.) $9.1x = 11.83$ e.) $-5.4x = 24.3$

14. A.) Check your answers using the scratchpad and the such that key.
To get to the such that key, press [ctrl] [=].

1.2*x = -4.2 | x = 3.5 is false because 1.2*3.5 is 4.2 not -4.2.
1.2*x = -4.2 | x = -3.5 is true because 1.2*-3.5 is -4.2.
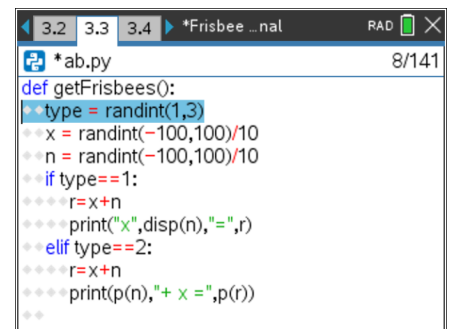
B.) Look back over your work in step 12.   What patterns do you notice?

Scratchpad	RAD ☒
$1.2 \cdot x = -4.2 \vert x = 3.5$	false
$1.2 \cdot x = -4.2 \vert x = -3.5$	true

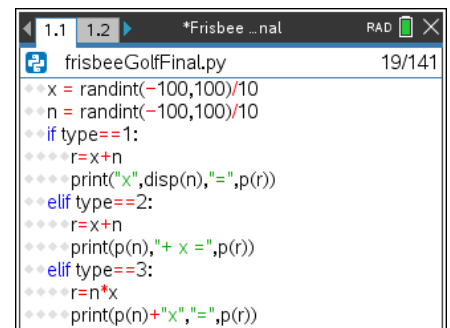15. Change the line that generates the **type** variable to randint(1,3).

type = randint(1,3)

```
def getFrisbees():
    type = randint(1,3)
    x = randint(-100,100)/10
    n = randint(-100,100)/10
    if type==1:
        r=x+n
        print("x",disp(n),"=",r)
    elif type==2:
        r=x+n
        print(p(n),"+ x =",p(r))
```

16. Add the following lines to create and display the third type of equation.

elif type == 3:
    r=x*n
    print(p(n)+"x","=",p(r))

```
    x = randint(-100,100)/10
    n = randint(-100,100)/10
    if type==1:
        r=x+n
        print("x",disp(n),"=",p(r))
    elif type==2:
        r=x+n
        print(p(n),"+ x =",p(r))
    elif type==3:
        r=n*x
        print(p(n)+"x","=",p(r))
```
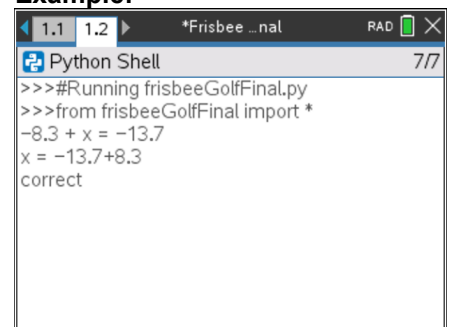
17. Execute your code. [ctrl] [r]
Play your game a few times. Ensure it displays all types of equations before continuing the code.

Remember: you can don't have to do the math in your head. The example on the right shows the user entered "-13.7 + 8.3" for the answer.

You should have:
addition problems where x comes first such as   x + 7.9 = 3.1
addition problems where x come second such as   4.5 + x = 9.4
multiplication problems such as   8.1x = 9.2

Example:

```
Python Shell                                7/7
>>>#Running frisbeeGolfFinal.py
>>>from frisbeeGolfFinal import *
-8.3 + x = -13.7
x = -13.7+8.3
correct
```
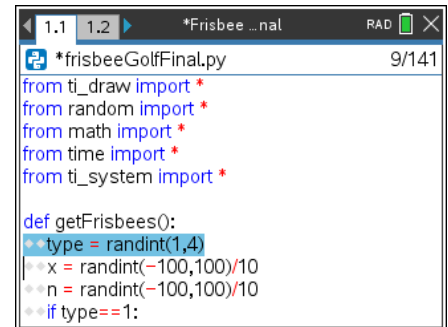
18. Now to program the fourth type of equation. Solve the following equations:

    a.) x/1.3 = 12.5   b.) x/3.8 = -28.1   c.) x/-2.4 = 17.2   d.) x/-9.7 = -90.1

    f.) Use the scratchpad and the such that key to check your work.
    g.) Look back over your work in step a.   What patterns do you notice?

19. Change the line that generates the **type** variable to randint(1,4).
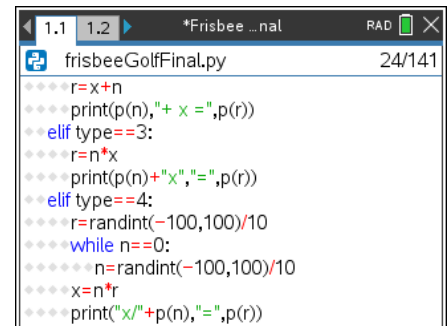
    type = randint(1,4)

20. Add the following lines to create and display the fourth type of equation.
    To keep the values "nicer", you'll generate r, then find x.
    You can't divide by 0. To avoid this possibility, you'll add a while loop that will
    regenerate n while it equals 0.

    elif type == 4:
      r = randint(-100,100)/10
      while n == 0:
        n=randint(-100,100)/10
      x=n*r
      print("x/"+p(n),"=",p(r))

    **randint   menu → random → randint

Execute your code. [ctrl] [r]

Play your game a few times.
Ensure it displays all 4 types of equations: x + n = r   n + x = r   n*x = r   x/n = r

21. Now to program the fifth type. Solve the following equations:
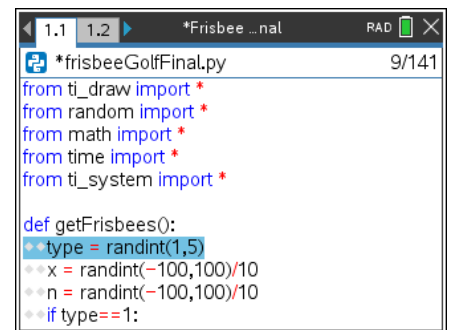
a.) 5.3x + 6.1 = 12.46   b.) -3.1x + 9.3 = 36.27   c.) 1.9x – 3.1 =4.69   d.) -9.2x – 3.1 = -70.26

f.) Use the scratchpad and the such that key to check your work.

g.) Look back over your work in step a.   What patterns do you notice?

22. Change the line that generates the **type** variable to randint(1,5).

type = randint(1,5)

23. Add the following lines to create and display the fifth type of equation.

```
elif type == 5:
    a = randint(-100,100)/10
    while a==0:
        a = randint(-100,100)/10
    r=a*x+n
    print(p(a)+"x",disp(n),"=",p(r))
```


24. Execute your code. [ctrl] [r]
    Play your game a few times.
    Ensure it displays all 5 types of equations: x + n = r   nx = r   ax + n = r
                                                  n + x = r   x/n = r

25. Now to program the sixth type. Solve the following equations:

 a.) 5.3(x + 6.1) = 12.72   b.) −2.6(x + 4.3) = 7.28   c.) 9.4x - 3.2 = -13.16   d.) −3.2(x - 1.4) = 20.16

 f.) Use the scratchpad and the such that key to check your work.

 g.) Look back over your work in step a.   What patterns do you notice?

26. Change the line that generates the **type** variable to randint(1,6).
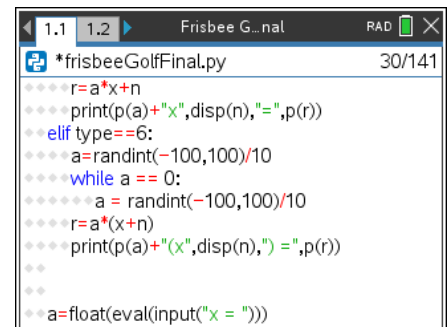
 type = randint(1,6)

```
1.1  1.2        Frisbee G…nal      RAD
*frisbeeGolfFinal.py                8/141
from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

def getFrisbees():
 type = randint(1,6)
 x = randint(−100,100)/10
 n = randint(−100,100)/10
 if type==1:
```

27. Add the following lines to create and display the sixth type of equation.

 elif type == 6:
     a = randint(-100,100)/10
     while a==0:
         a = randint(-100,100)/10
     r=r*(x+n)
     print(p(a)+"(x",disp(n),") =",p(r))

```
1.1  1.2        Frisbee G…nal      RAD
*frisbeeGolfFinal.py               30/141
    r=a*x+n
    print(p(a)+"x",disp(n),"=",p(r))
   elif type==6:
    a=randint(−100,100)/10
    while a == 0:
     a = randint(−100,100)/10
    r=a*(x+n)
    print(p(a)+"(x",disp(n),") =",p(r))


 a=float(eval(input("x = ")))
```

28. Execute your code. [ctrl] [r]

 Play your game a few times.

 Ensure it displays all 6 types of equations: x + n = r   nx = r   ax + n = r
                                               n + x = r   x/n = r   a*(x + n) = r

29. Congratulations! Your Frisbee Golf game is complete. The only thing left to do is practice!

30. Optional Challenges:

 *Can you generate equations in the form (x + n)/a =r.   For example, (x + 2.8)/7.1 = 3.4   or (x – 1.9)/2.5 = -3.1

         *Hint: a should not be 0. Therefore, use code similar to step 27 to generate a.

             r = (x + n)/a

             To print, write **print("(x", disp(n), ")/", p(a), "=", p(r))**

 What about an equation with four variables such as a(x + n) + b = r? An example would be 3.1(x + 2.2) – 8.1 = 1.3.

*Hint: generate a value for b

r = a*(x + n) + b

To print, write   **print(p(a) + "(x", disp(n), ")", disp(b), "=", p(r))**