

**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 “GOLFTEMP.8xv” 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 “GOLFTEMP.8xv” 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 Overview:**

```

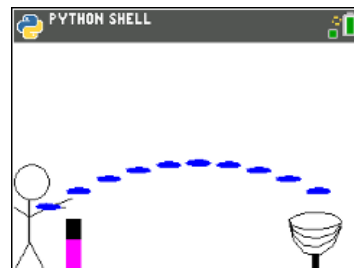
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running FRGLFFIN
>>> from FRGLFFIN import *
x/-6.6 = -9.2
x = |
    
```

The game asks for the value of x

```

PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running FRGLFFIN
>>> from FRGLFFIN import *
x/-6.6 = -9.2
x = -9.2*-6.6
correct
1.3 + x = 1.9
x = 1.9+1.3
sorry, x = 0.6
x - 4.7 = 2.6
x = |
    
```

For each question answered correctly, earn a frisbee.



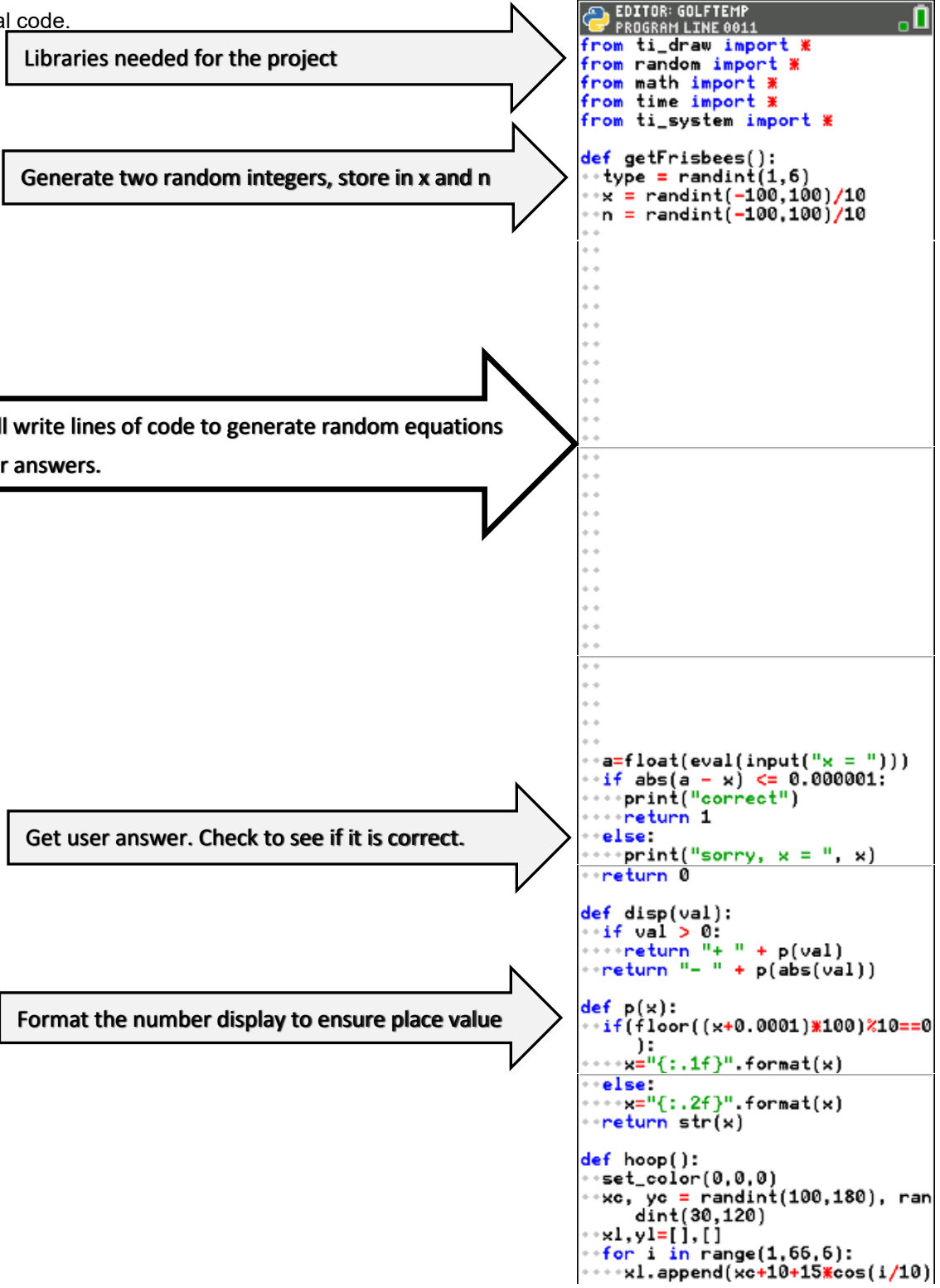
Press the “clear” key to aim and throw the frisbee.

**Teacher Notes:**

Note: Student calculator needs the additional ti\_draw.8xv AppVar which is available in the OS and Apps bundle version 5.7 for the handheld. If you are using SmartView, you will need to make sure you have updated the version 5.7 or above.

1. Obtain the “GOLFTEMP.8xv” from your teacher. A large portion of the code has been prepared for you ahead of time.

2. Let's examine the initial code.



The diagram shows a TI-84 CE editor window titled 'EDITOR: GOLFTEMP' with 'PROGRAM LINE 0011'. The code is as follows:

```

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
    ...
    ...
    ...
    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+0.0001)*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), ran
    dint(30,120)
    x1,y1=[],[]
    for i in range(1,66,6):
        x1.append(xc+10+15*cos(i/10)

```

Callouts and their corresponding code sections:

- Libraries needed for the project** points to the import statements at the top of the code.
- Generate two random integers, store in x and n** points to the initialization of `x` and `n` in the `getFrisbees()` function.
- \*You will write lines of code to generate random equations and their answers.** points to the ellipsis (...) lines in the `getFrisbees()` function.
- Get user answer. Check to see if it is correct.** points to the input and conditional logic in the `getFrisbees()` function.
- Format the number display to ensure place value** points to the `p(x)` function.



Draw the hoop continued

```

    )
    *... yl.append(ye-10+5*sin(i/10))
    *xl.append(xl[0])
    *yl.append(yl[0])
    *draw_poly(xl,yl)
    *
    *for a in range(2,5):
    *... xl=[]
    *... yl=[]
    *... for i in range(31,66,6):
    *... *xl.append(xc+10+15*cos(i/10))
    *... *yl.append(ye-10+5*a*sin(i/10))
    *... draw_poly(xl,yl)
    *
    *fill_rect(xc+8,ye-48,5,20)
    *return xc, ye

```

Draw the person

```

def person():
*set_color(0,0,0)
*draw_circle(10,50,10)
*fill_rect(8,15,2,25)
*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

```

def frisbee(x,y):
*set_color(0,0,255)
*xl=[]
*yl=[]
*for i in range(0,66,6):
*... xl.append(x+7*cos(i/10))
*... yl.append(y+2*sin(i/10))
*fill_poly(xl,yl)

```

Animate the frisbee

```

def throw(x,y,angle,v,time):
*x = 20+v*cos(radians(angle))*time
*y = -16*(time**2) + v*sin(radians(angle))*time + 36
*return x,y

```

Animate toss angle until the user presses "clear"

```

def setAngle(angle):
*add = 5
*set_color(0,0,0)
*draw_line(20,36,30,36)
*while not escape():
*... 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)),36+10*sin(rad

```



Change and toss velocity until the user presses "clear"

\*Repeatedly ask the equation questions. Add earned frisbees. You will eventually change range(0) to range(5).

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

```

    ians(angle))
    *sleep(.2)
    *set_color(255,255,255)
    *draw_line(20,36,25+10*cos(radians(angle)),36+10*sin(radians(angle)))
    *set_color(0,0,0)
    *draw_line(20,36,25+10*cos(radians(angle)),36+10*sin(radians(angle)))
    *return angle
def setVelocity(v):
    *add = 10
    *while not escape():
        *v += add
        *set_color(0,0,0)
        *fill_rect(30,0,10,30)
        *set_color(255,0,255)
        *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 num. of questions
for i in range(0):
    *count += getFrisbees()
    *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()
    *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+= 0.25
    *sleep(2)

```

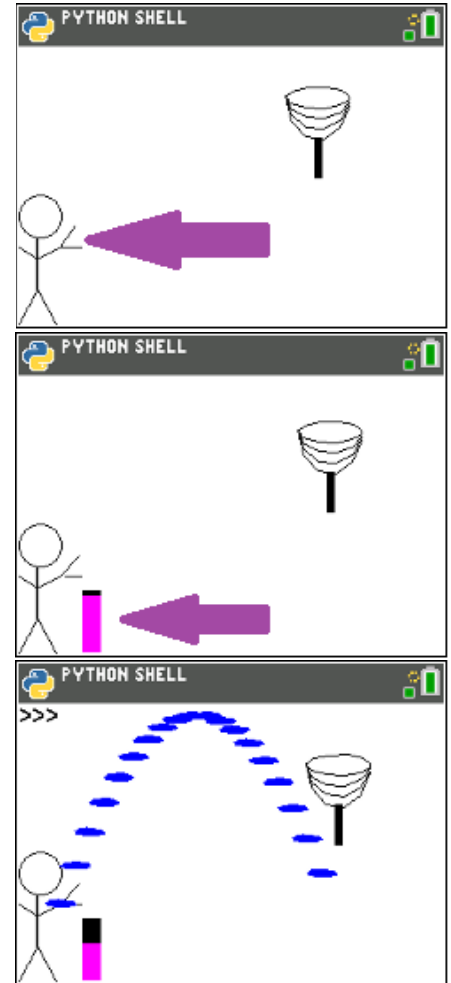
3. Obtain the file “GOLFTEMP.8xv” if you don’t already have it.

Execute the initial code ( [Trace] ).

The “aim” angle will automatically increase and decrease. Press “clear” when the desired angle is achieved.

Now, determine how hard to throw the frisbee. When ready, press “clear”.

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:  $x + \text{number} = \text{result}$  or  $\text{number} + x = \text{result}$


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.)  $x - 5.3 = 7.8$

**Teacher Notes:**

- a.) 5    b.) 3.5    c.) 9.2    d.) 4.3    e.) 13.1

5. A.) Check your answers from step 4.

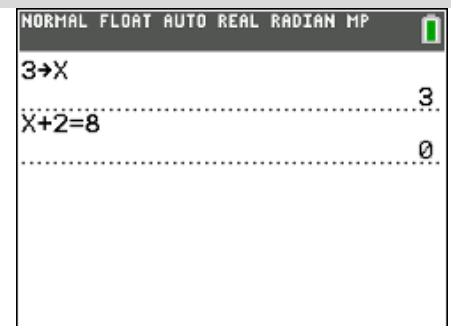
Did you now you can use the store key  to store and check values.

Type the following.

$3 \rightarrow x$

$x + 2 = 8$

Notice the screen on the right shows a 0 for False. When x is a 3,  $x + 2$



evaluates to 6.

Type the following.

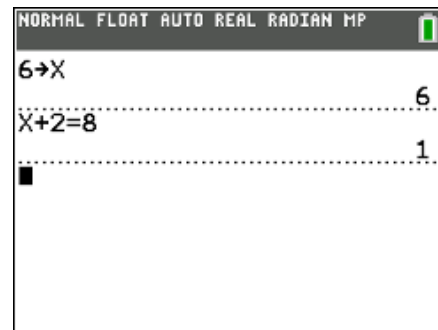
$6 \rightarrow x$

$x + 2 = 8$

Notice the screen on the right shows a 1 for True. When  $x$  is a 6,  $x + 2$  evaluates to 8.

## FRISBEE GOLF EQUATIONS

### TEACHER DOCUMENT



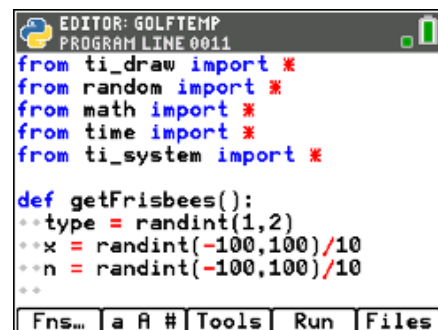
B.) Look back over your work in step 4. What patterns do you notice?

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



#### Teacher Notes:

Changing the last two lines above to `x = randint(-10,10)` and `n = randint(-10,10)` will result in integer constants between -10 and 10. If only positive integers for  $x$  and are, use code such as `x = randint(1,20)`.

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`

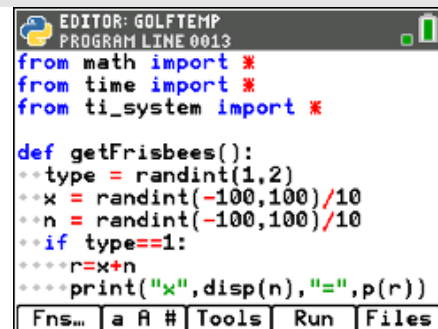
`print("x",disp(n),"=",p(r))`

**\*\*if** Fns  $\rightarrow$  Ctl  $\rightarrow$  if

**\*\*print** Fns  $\rightarrow$  I/O  $\rightarrow$  print

**\*\*The** [a A #] menu might be helpful for typing

*Make sure the two lines below the if are indented two spaces (diamonds).*



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 Fns → Ctl → elif

\*\*The tools menu will allow you to copy and paste lines.

*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 8.1999999999. The function `p()` ensures this doesn't happen when printed.*

9. The code has already been included 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.

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. Run ([Trace])

*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.

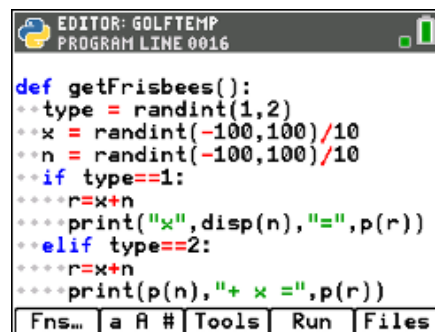
Scroll down to the section labeled “controls the number of questions”

Change the line

`for i in range(0):` to `for i in range(5):`

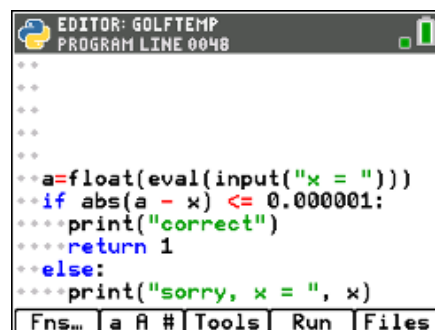
## FRISBEE GOLF EQUATIONS

### TEACHER DOCUMENT



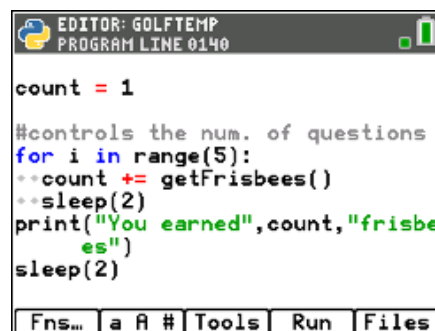
```
EDITOR: GOLFTEMP
PROGRAM LINE 0016

def getFrisbees():
    *type = randint(1,2)
    *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))
```



```
EDITOR: GOLFTEMP
PROGRAM LINE 0048

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



```
EDITOR: GOLFTEMP
PROGRAM LINE 0140

count = 1

#controls the num. of questions
for i in range(5):
    *count += getFrisbees()
    *sleep(2)
    print("You earned",count,"frisbee
    es")
    sleep(2)
```

#### Teacher Notes:

Students may put any positive integer inside the `range()` function.

For example, “for i in range(3):” would result in three questions.

12. Execute the code Run ([Trace]).

The program should ask 5 questions.

The sample question on the right is “ $-6.9 + x = -5.3$ ”

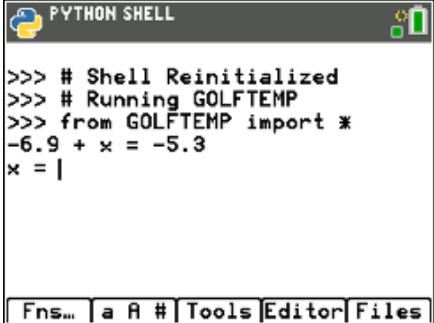
The user may either enter **1.6** or **-5.3 + 6.9**.

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.



```


PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running GOLFTEMP
>>> from GOLFTEMP import *
-6.9 + x = -5.3
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.)  $-5.4x = 24.3$

**Teacher Notes:**

- a.) -3.5      b.) -3.7      c.) 0      d.) 1.3      e.) -4.5

14. A.) Check your answers. You could use the store key  like you did in step 5. Remember, if the value for x is incorrect, it will evaluate to 0. If the x value is correct, it will evaluate to 1.

Example:

- $1 \rightarrow x$   
 $1.2 * x = -4.2$       Evaluates to 0 because it is False
- $3.5 \rightarrow x$   
 $1.2 * x = -4.2$       Evaluates to 1 because it is True

B.) Look back over your work in step 13. What patterns do you notice?



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

```
type = randint(1,3)
```



```

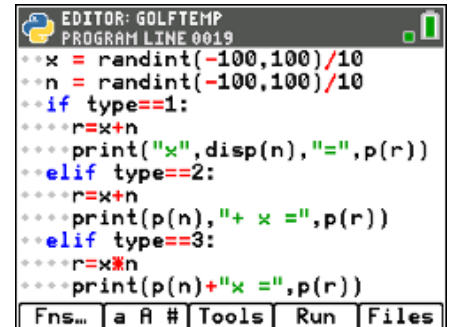
EDITOR: GOLFTMP
PROGRAM LINE 0008
from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

def getFrisbees():
    type = randint(1,3)
    x = randint(-100,100)/10
    n = randint(-100,100)/10
    if type==1:
    
```

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))
    
```



```

EDITOR: GOLFTMP
PROGRAM LINE 0019
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=x*n
    print(p(n)+"x =",p(r))
    
```

17. Execute your code. Run ([Trace])

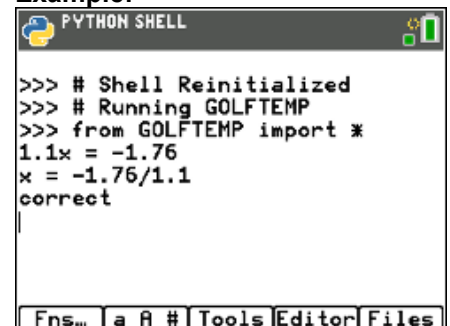
Play your game a few times. Ensure it displays all types of equations before continuing the code.

**Remember:** you can don't have to do the math in your head. The example on the right shows the user entered "-1.76/1.1" 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
>>> # Shell Reinitialized
>>> # Running GOLFTMP
>>> from GOLFTMP import *
1.1x = -1.76
x = -1.76/1.1
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.)  $x/-9.7 = -90.1$

- e.) Check your work. Make sure your answers are correct.  
 f.) Look back over your work in step a. What patterns do you notice?

**Teacher Notes:**

- a.) 16.25      b.) 107.16      c.) -41.28      d.) 873.97

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

```
type = randint(1,4)
```



```
EDITOR: GOLFTEMP
PROGRAM LINE 0008
from ti_draw import *
from random import *
from math import *
from time import *
from ti_system import *

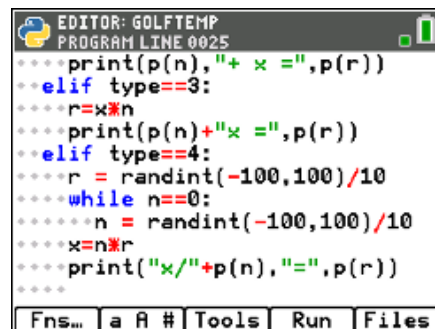
def getFrisbees():
    type = randint(1,4)
    x = randint(-100,100)/10
    n = randint(-100,100)/10
    if type==1:
```

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))
```



```
EDITOR: GOLFTEMP
PROGRAM LINE 0025
print(p(n),"+ x =",p(r))
elif type==3:
    r=x*n
    print(p(n)+"x =",p(r))
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))
```

\*\*while      Fns → Ctl → while

\*\*randint    Fns → Modl → random → randint

\*\*[Tools] has copy and paste options that might make things easier to modify.

Execute your code. Run ([Trace])

Play your game a few times.

Ensure it displays all 4 types of equations:  $x + n = r$        $n + x = r$        $n*x = r$        $x/n = r$

**Teacher Notes:**

Teachers may choose to stop after step 20 if all that is desired is one step equations.

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.)  $-9.2x - 3.1 = -70.26$

- e.) Check your work.
- f.) Look back over your work in step a. What patterns do you notice?

**Teacher Notes:**

- a.) 1.2      b.) -8.7      c.) -4.1      d.) 7.3

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

```
type = randint(1,5)
```

```

EDITOR: GOLFTMP
PROGRAM LINE 0008
from time import *
from ti_system import *

def getFrisbees():
    type = randint(1,5)
    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:

```

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))

```

```

EDITOR: GOLFTMP
PROGRAM LINE 0022
while n==0:
    n = randint(-100,100)/10
x=n*r
print("x/"+p(n),"=",p(r))
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. Run ([Trace])

Play your game a few times.

Ensure it displays all 5 types of equations:  $x + n = r$        $nx = r$        $ax + n = r$   
 $n + x = r$        $x/n = r$

**Teacher Notes:**

Teachers may choose to stop after step 24, or continue on to equations in the form  $a(x + b) = c$ .

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.)  $-3.2(x - 1.4) = 20.16$

- e.) Check your work.
- f.) 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)
```

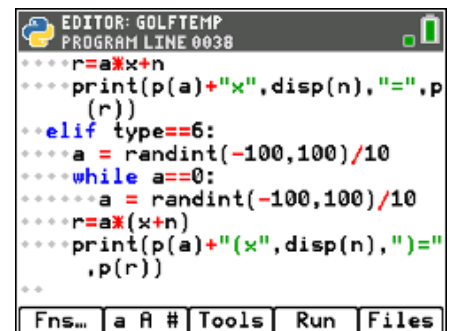


```
EDITOR: GOLFTMP
PROGRAM LINE 0008
from random import *
from math import *
from time import *
from ti_system import *

def getFrisbees():
    type = randint(1,6)
    x = randint(-100,100)/10
    n = randint(-100,100)/10
    if type==1:
        r=x+n
    .....
```

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=a*(x+n)
    print(p(a)+"(x",disp(n),") =",p(r))
```



```
EDITOR: GOLFTMP
PROGRAM LINE 0038
    .....
```

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  $(x - 1.9)/2.5 = -3.1$

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

**Teacher Notes:**

```
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:
    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=x*n
    print(p(n)+"x =",p(r))
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))
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))
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 = ")))
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+0.0001)*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)
    xl,yl=[],[]
    for i in range(1,66,6):
        xl.append(xc+10+15*cos(i/10))
        yl.append(yc-10+5*sin(i/10))
    xl.append(xl[0])
    yl.append(yl[0])
    draw_poly(xl,yl)

for a in range(2,5):
    xl=[]
    yl=[]
    for i in range(31,66,6):
        xl.append(xc+10+15*cos(i/10))
        yl.append(yc-10+5*a*sin(i/10))
    draw_poly(xl,yl)

fill_rect(xc+8,yc-48,5,20)
```

```
return xc, yc

def person():
    set_color(0,0,0)
    draw_circle(10,50,10)
    fill_rect(8,15,2,25)
    draw_line(0,0,9,16)
    draw_line(9,16,18,0)
    draw_line(0,36,9,30)
    draw_line(9,30,20,36)

def frisbee(x,y):
    set_color(0,0,255)
    xl=[]
    yl=[]
    for i in range(0,66,6):
        xl.append(x+7*cos(i/10))
        yl.append(y+2*sin(i/10))
    fill_poly(xl,yl)

def throw(x,y,angle,v,time):
    x = 20+v*cos(radians(angle))*time
    y = -16*(time**2) + v*sin(radians(angle))*time + 36
    return x,y

def setAngle(angle):
    add = 5
    set_color(0,0,0)
    draw_line(20,36,30,36)
    while not escape():
        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)),36+10*sin(radians(angle)))
        sleep(.2)
        set_color(255,255,255)
        draw_line(20,36,25+10*cos(radians(angle)),36+10*sin(radians(angle)))
    set_color(0,0,0)
    draw_line(20,36,25+10*cos(radians(angle)),36+10*sin(radians(angle)))
    return angle

def setVelocity(v):
    add = 10
```

```
while not escape():
    v += add
    set_color(0,0,0)
    fill_rect(30,0,10,30)
    set_color(255,0,255)
    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 num. of questions
for i in range(5):
    count += getFrisbees()
    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()
    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+= 0.25
    sleep(2)
```