Fraction Tic Tac Toe

In this project, you will create a fraction tic tac toe game. The tic tac toe board and scoring code have already been written for you in the "Tic Tac Toe Template.tns" file. You will write the code to generate fraction addition, subtraction, multiplication and division problems. You will also write the code to request the user's answer. The user must answer the question correctly to earn a new spot on the board. The computer always earns a new spot, so math carefully!

Objectives:

Programming Objectives:

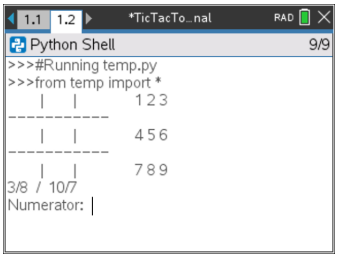
- Use variables to store values
- Use the randint() function to generate integers
- Use the print() function to display
- Use a while loop to repeat code.

Math Objectives:

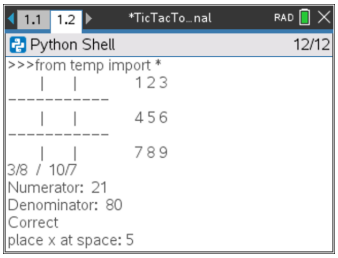
- Add and subtract fractions with unlike denominators.
- Multiply and divide fractions.
- Add, subtract, multiple, and divide fractions with positive and negative values.

Math Course Connections: Middle School Mathematics

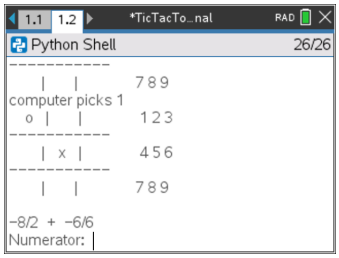
In this project, you will create a fraction tic tac toe game. The tic tac toe board and scoring code have already been written for you in the "Tic Tac Toe Template.tns" file. You will write the code to generate fraction addition, subtraction, multiplication and division problems. You will also write the code to request the user's answer. The user must answer the question correctly to earn a new spot on the board. The computer always earns a new spot, so math carefully!



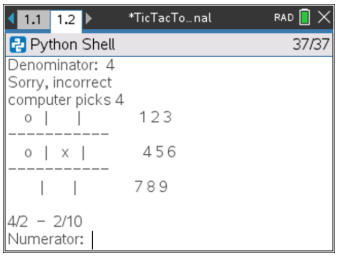
User given fraction problem



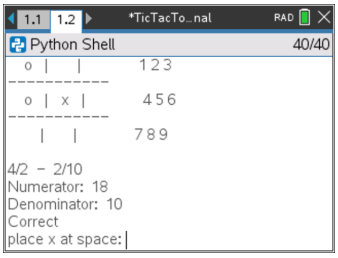
If answered correctly, earns an "x"



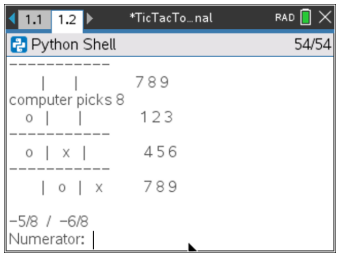
Computer plays "o". Next question.



User enters incorrect answer. Lose a turn. Computer gets to place "o".



User answers correctly, gets to choose a location on the board.



Game continues until someone wins or spaces are filled.

1. Obtain the "Tic Tac Toe Template.tns" from your teacher. Part of the programming code has been coded for you.

2. Let's examine the code template.

Libraries needed for the project

*You write code to create one fraction

*You write code to create 2 new fractions using the function above.
Generate a question, show the question, get the user's answer

Code checks if answer is correct

Determine and display current board

Checks tic tac toe board for a winner

Create the initial tic tac toe board

```
1.1  Tic Tac To...ate  RAD
*FractionTicTacToe.py  1/82
from time import *
from random import *

def fraction():




def problem():




    #check answer
    if abs( answer - un/ud) < 0.000001:
        print("Correct")
        sleep(2)
        return True
    else:
        print("Sorry, incorrect ")
        sleep(2)
        return False

def board(lst,player):
    str = "   " + lst[1] + "  |  " + list[2] + "  |  " + ls
    str += "\n" + "-"*11 + "\n"
    str += "   " + lst[4] + "  |  " + list[5] + "  |  " +
    str += "\n" + "-"*11 + "\n"
    str += "   " + lst[7] + "  |  " + list[8] + "  |  " +
    print(str)

def check(lst):
    for a in range(0,3):
        if lst[1+a*3]!=" "and lst[1+a*3]==lst[2+a*3]=
            return True
        if lst[1+a]!=" " and lst[1+a]==lst[4+a]==lst[7
            return True
    if lst[1]==lst[5]==lst[9]!=" " or lst[3]==lst[5]==
        return True
    return False

list=[" "]*10
computer=[1,2,3,4,5,6,7,8,9]
win=False
board(list,"x")
```

(Continues on the next page)

Loop generates multiple rounds of player and computer play

```
while win==False and len(computer)>0:
    if problem()==True:
        player="x"
        spot=int(input("place "+ player+" at space: "
        while spot < 1 or spot >9 or list[spot] != " ":
            spot=int(input("place "+ player+" at spac
        board(list,player)
        list[spot]=player
        computer.remove(spot)
        win=check(list)
    sleep(1)
    if win==False:
        player="o"
        c=choice(computer)
        computer.remove(c)
        print("computer picks",c)
        list[c]=player
        board(list,player)
        win=check(list)
    print(" ")

board(list,player)
if win==True and player=="x":
    print("You win!!")
elif win==True:
    print("Sorry, the computer wins")
else:
    print("No one wins!!")
```

Display the winner

3. The first step to create your fraction tic tac toe game is to create a fraction. To create positive numerator values from 1 to 10, you would write the code n = randint(1,10). To create the numbers 3 to 7 it would be n = randint(3, 7). You will create integer values from -10 to 10.

   Write the following line of code to generate a number from -10 to 10 and store it in a variable **n**. Write this line under the *def fraction()* function.

   n = randint(-10,10)

   *randint is in the menu menu → random → randint()
   *Make sure n is on the left side of the equation. When programming, the variable goes on the left, the math goes on the right.

```
1.1  *Tic Tac T...ess  RAD
*FractionTicTacToe.py  5/80
from time import *
from random import *

def fraction():
    n=randint(-10,10)


def problem():
```

4. For this game, we do not want to allow the numerator to be a zero. You'll write a loop that generates a new value for n while it equals zero.

   Add the lines: **while n==0:**
   n=randint(-10,10)

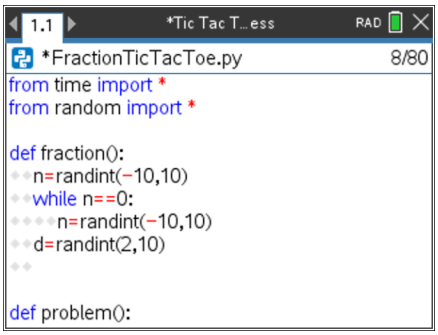
   **while is in the menu menu → built-ins → control → while

```
1.1  *Tic Tac T...ess  RAD
*FractionTicTacToe.py  7/8
from time import *
from random import *

def fraction():
    n=randint(-10,10)
    while n==0:
        n=randint(-10,10)


def problem():
```

5. The denominator should be an integer between 2 and 10. Because we are not making zero an option, we won't need a loop to make sure it doesn't happen. Add a line of code that creates a variable d with values between 2 and 10.

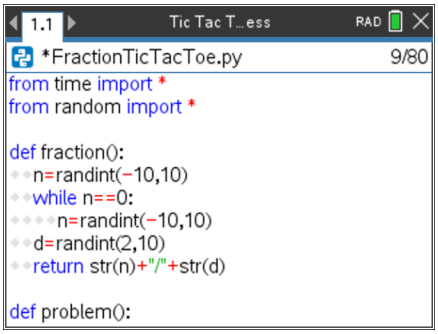
6. Did you add the line:

   d = randint(2,10)



7. The last step to creating a fraction is to put the numerator and denominator together. Because this is code is written inside a *definition* named fraction, we will *return* the fraction so it can be used in other parts of the code.

   You will write **return str(n) + "/" + str(d)**. The numerator and denominator are both integers while the sign "/" is a string. To put these two data types together, you must write str(n) and str(d) to treat the numbers as characters.



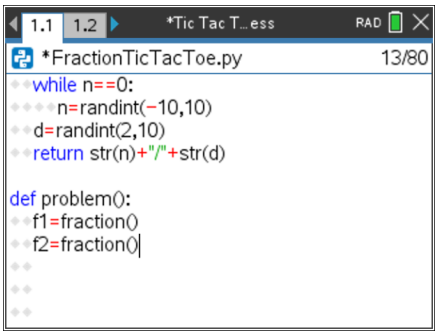
   Add the line:

   return str(n) + "/" + str(d)

   **return menu → built-ins → functions → return
   **str() menu → built-ins → type → string
   ** " " [ctrl] [*]

8. Now to create a question.

   Go to the second function *def problem()*. Create two fractions, f1 and f2.



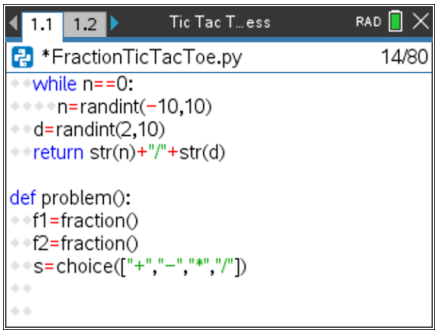
   Add the lines:

   f1 = fraction()
   f2 = fraction()

   **You can type both lines, or you can copy and paste to get the second line. To copy, put the cursor at the beginning of the first line. Press [shift] and use the right arrow key to highlight the line. To copy, use [doc] → edit → copy

9. Now to determine the type of question. The function *choice()* will select a random item from items within the parenthesis. To randomly select an operation and store the sign choice, type:

   s = choice(["+","-","*","/"])

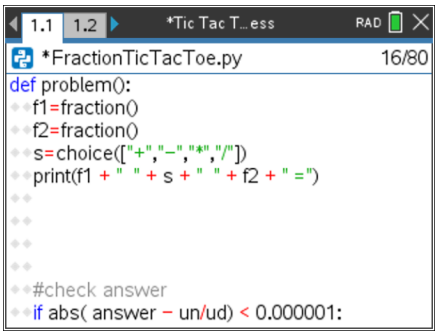


   *choice menu → random → choice

10. You have the two fractions, *f1* and *f2*, and an operation *s*.
    Now to print them to the screen.

    print(f1 + " " + s + " " + f2 + " =")

    **print menu → built-ins → i/o → print

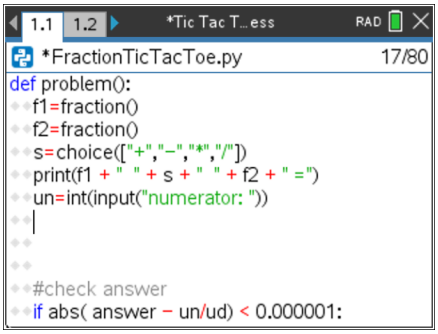


11. After printing the fraction question, the user needs to enter an answer.
    Ask the user to input the numerator. The result in a variable named **un,** short for user's numerator.

    un=int(input("numerator"))

    **int menu → built-ins → type → int
    **input menu → built-ins → i/o → input

    **If you don't put int() around the input() Python will treat your answer like characters instead of integers!



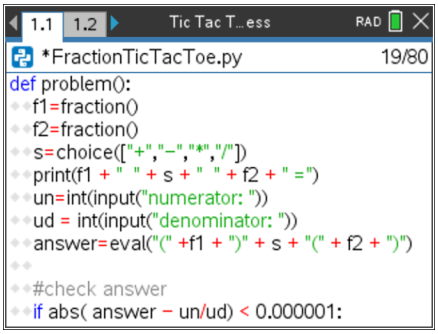
12. Add another line to ask for the denominator. Store the result in a variable named **ud**, short for user's denominator. *(This should look similar to the line from step 11.)*

13. The last step is to calculate the real answer to the question. The code has already been written to check the real answer against the user's input. We just need to calculate the correct answer.
    Add:

    answer=eval("(" +f1 + ")" + s + "(" + f2 + ")")

    **eval menu → built-ins → i/o → eval

    If typed correctly,
    All the plus signs should appear in red when you are done.
    All the "(" and ")" should appear in green.



14. You're done! Unless you have errors, when you press [ctrl] [r]

    Play the game several times. Can you beat the computer each time?