

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:

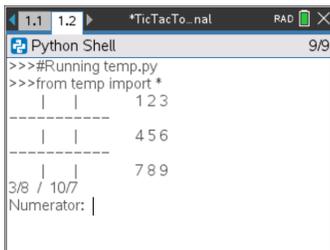
- Use variables to store values
- Use the randint() function to generate integers
- Use the print() function to display
- Use a while loop to repeat code.

Math Objectives:

- Add and subtract fractions with unlike denominators.
- Multiply and divide fractions. (*Teacher Note step 9 tells how to restrict to add and subtract*)
- Add, subtract, multiple, and divide fractions with positive and negative values. (*Teacher note step 3 to restrict to positive values*).

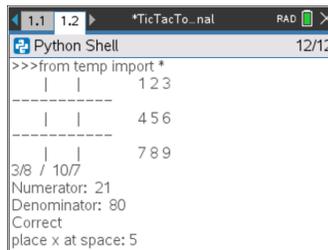
Math Course Connections: Middle School Mathematics

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!



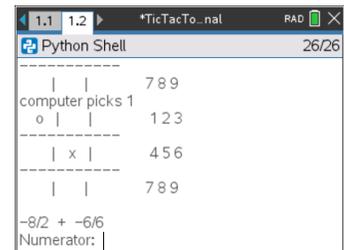
```
Python Shell 9/9
>>>#Running temp.py
>>>from temp import *
| | | 1 2 3
-----
| | | 4 5 6
-----
| | | 7 8 9
3/8 / 10/7
Numerator: |
```

User given fraction problem



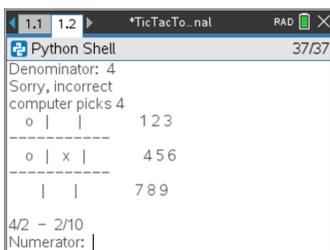
```
Python Shell 12/12
>>>from temp import *
| | | 1 2 3
-----
| | | 4 5 6
-----
| x | 7 8 9
3/8 / 10/7
Numerator: 21
Denominator: 80
Correct
place x at space: 5
```

If answered correctly, earns an “x”



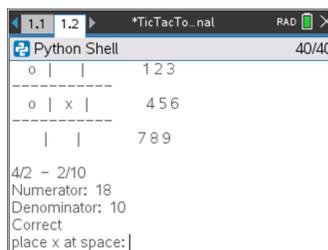
```
Python Shell 26/26
| | | 7 8 9
computer picks 1
o | | 1 2 3
-----
| x | 4 5 6
-----
| | | 7 8 9
-8/2 + -6/6
Numerator: |
```

Computer plays “o”. Next question.



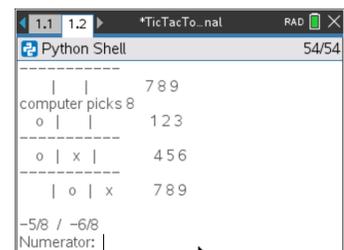
```
Python Shell 37/37
Denominator: 4
Sorry, incorrect
computer picks 4
o | | 1 2 3
-----
o | x | 4 5 6
-----
| | | 7 8 9
4/2 - 2/10
Numerator: |
```

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



```
Python Shell 40/40
o | | 1 2 3
-----
o | x | 4 5 6
-----
| | | 7 8 9
4/2 - 2/10
Numerator: 18
Denominator: 10
Correct
place x at space: |
```

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

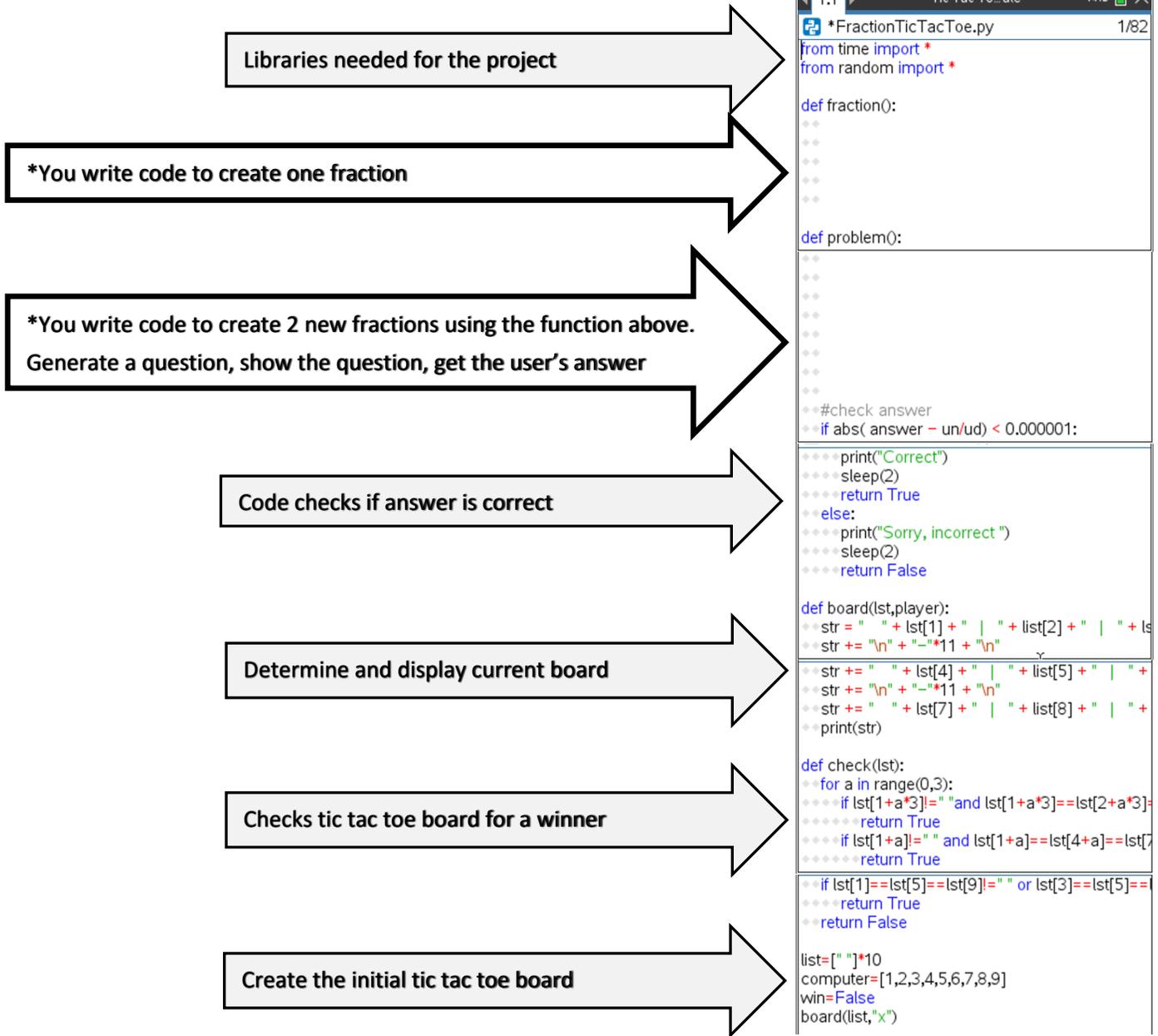


```
Python Shell 54/54
| | | 7 8 9
computer picks 8
o | | 1 2 3
-----
o | x | 4 5 6
-----
| o | x 7 8 9
-5/8 / -6/8
Numerator: |
```

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.



The diagram consists of seven text boxes on the left, each with an arrow pointing to a specific section of Python code in a screenshot on the right. The code is for a Fraction Tic Tac Toe game and includes imports, function definitions for creating fractions, checking answers, displaying the board, checking for a winner, and initializing the game state.

Libraries needed for the project → `from time import *`
***You write code to create one fraction** → `def fraction():`
***You write code to create 2 new fractions using the function above. Generate a question, show the question, get the user's answer** → `def problem():`
Code checks if answer is correct → `#check answer`
Determine and display current board → `def board(lst,player):`
Checks tic tac toe board for a winner → `def check(lst):`
Create the initial tic tac toe board → `list=[" "]*10`

(Continues on the next page)

Loop generates multiple rounds of player and computer play

Display the winner

```
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 space: "))
        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!!")
```

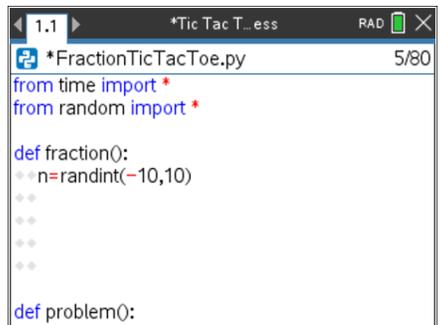
- The first step to create your fraction tic tac toe game is to create a fraction. To create positive numerator values from 1 to 10, you would write the code `n = randint(1,10)`. To create the numbers 3 to 7 it would be `n = randint(3, 7)`. You will create integer values from -10 to 10.

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.



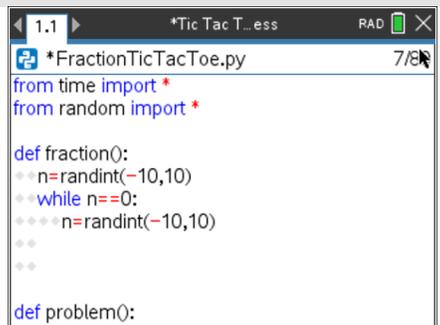
Teacher Notes:
If only positive fractional values are desired, students should put `n = randint(1,10)`. Students do not need step 4 if using only values 1 through 10.

- 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)
```

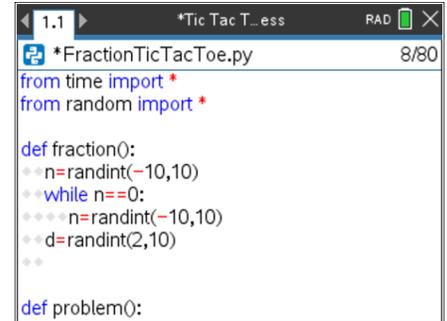
**while is in the menu menu → built-ins → control → while



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)
```



```

1.1 *FractionTicTacToe.py 8/80
from time import *
from random import *

def fraction():
    n=randint(-10,10)
    while n==0:
        n=randint(-10,10)
    d=randint(2,10)

def problem():

```

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.

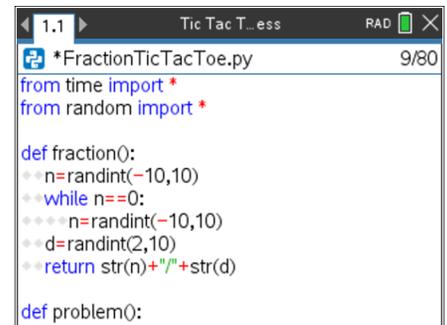
Add the line:

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

****return** menu → built-ins → functions → return

****str()** menu → built-ins → type → string

**** " "** [ctrl] [*]



```

1.1 Tic Tac T...ess 9/80
*FractionTicTacToe.py
from time import *
from random import *

def fraction():
    n=randint(-10,10)
    while n==0:
        n=randint(-10,10)
    d=randint(2,10)
    return str(n)+"/"+str(d)

def problem():

```

8. Now to create a question.

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

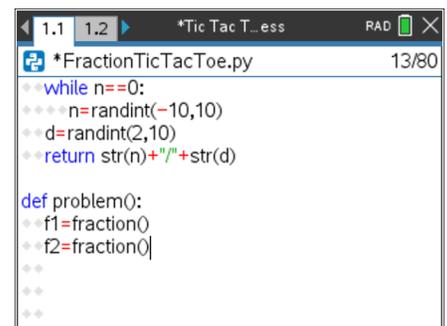
Add the lines:

```
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



```

1.1 1.2 *Tic Tac T...ess 13/80
*FractionTicTacToe.py
while n==0:
    n=randint(-10,10)
    d=randint(2,10)
    return str(n)+"/"+str(d)

def problem():
    f1=fraction()
    f2=fraction()

```

Teacher Notes:

If only positive fractions for questions and answers are desired, students need to write **`n=randint(1,10)`** in step 3. They also need to add the following lines here:



```
while eval(f1) < eval(f2):
    f2=fraction()
```

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

```
*FractionTicTacToe.py 14/80
while n==0:
    n=randint(-10,10)
    d=randint(2,10)
    return str(n)+"-"+str(d)

def problem():
    f1=fraction()
    f2=fraction()
    s=choice(["+", "-", "*", "/"])
    ...
```

Teacher Notes:

If only addition and subtraction questions are desired, write **s = choice(["+", "-"])** instead.

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

```
*FractionTicTacToe.py 16/80
def problem():
    f1=fraction()
    f2=fraction()
    s=choice(["+", "-", "*", "/"])
    print(f1 + " " + s + " " + f2 + " =")
    ...

#check answer
if abs( answer - un/ud) < 0.000001:
```

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

```
*FractionTicTacToe.py 17/80
def problem():
    f1=fraction()
    f2=fraction()
    s=choice(["+", "-", "*", "/"])
    print(f1 + " " + s + " " + f2 + " =")
    un=int(input("numerator: "))
    |
    ...

#check answer
if abs( answer - un/ud) < 0.000001:
```

**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

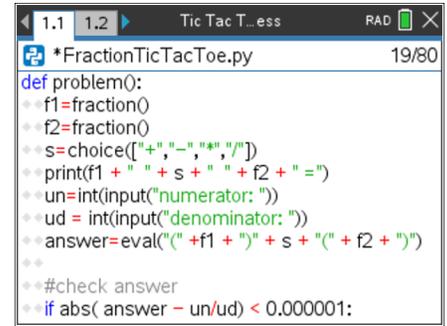
If typed correctly,

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?



```
*FractionTicTacToe.py 19/80
def problem():
    f1=fraction()
    f2=fraction()
    s=choice(["+", "-", "*", "/"])
    print(f1 + " " + s + " " + f2 + " =")
    un=int(input("numerator: "))
    ud = int(input("denominator: "))
    answer=eval("(" + f1 + ")" + s + "(" + f2 + ")")
    #check answer
    if abs( answer - un/ud) < 0.000001:
```

Teacher Notes:

#Original code written for positive and negative values

```
from time import *
from random import *

def fraction():
    n=randint(-10,10)
    while n==0:
        n=randint(-10,10)
    d=randint(2,10)
    return str(n)+"/"+str(d)

def problem():
    f1=fraction()
    f2=fraction()
    s=choice(["+", "-", "*", "/"])
    print(f1 + " " + s + " " + f2 + " =")
    un=int(input("numerator: "))
    ud = int(input("denominator: "))
    answer=eval("(" + f1 + ")" + s + "(" + f2 + ")")

#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] + " | " + lst[3] + "      1 2 3"
    str += "\n" + "-"*11 + "\n"
    str += "  " + lst[4] + " | " + list[5] + " | " + lst[6] + "      4 5 6"
    str += "\n" + "-"*11 + "\n"
    str += "  " + lst[7] + " | " + list[8] + " | " + lst[9] + "      7 8 9"
    print(str)

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

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

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 space: "))
        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!!")
```

Teacher Notes:

#Only Positive values.

```
from time import *
from random import *
```

```
def fraction():
    n=randint(1,10)
    d=randint(2,10)
    return str(n)+"/"+str(d)
```

```
def problem():
    f1=fraction()
    f2=fraction()
    while eval(f1) < eval(f2):
        f2=fraction()
    s=choice(["+", "-", "*", "/"])
    print(f1 + " " + s + " " + f2 + " =")
    un=int(input("numerator: "))
    ud = int(input("denominator: "))
    answer=eval("(" + f1 + ")" + s + "(" + f2 + ")")
```

```
#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] + " | " + lst[3] + "      1 2 3"
    str += "\n" + "-"*11 + "\n"
    str += "  " + lst[4] + " | " + list[5] + " | " + lst[6] + "      4 5 6"
    str += "\n" + "-"*11 + "\n"
    str += "  " + lst[7] + " | " + list[8] + " | " + lst[9] + "      7 8 9"
    print(str)
```

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

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

```
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 space: "))
        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!!")
```