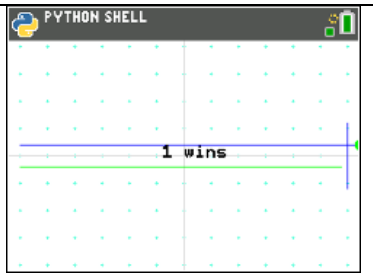
Turtle Graphics **Races**

This activity extends your turtle skills and concepts and demonstrates the use of multiple turtles competing in a 'race' across the screen.

0. Introduction:

   Two turtles are in a race to the finish line! Let's write a program that simulates this race.



1. Start a new Python program (**TD**) and add the turtle module to the code by using **<Fns> Modul <Add-Ons>** :

from turtle import *

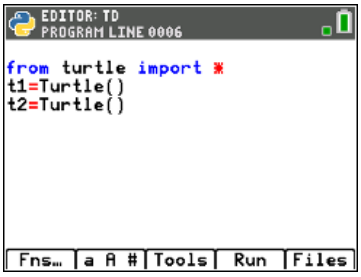
Recall that selecting this statement from the menu actually pastes *two* statements into your Editor as seen here.

But this program requires *two* turtles. The second line only creates one turtle object, **t**.

2. Change the variable name of the turtle from **t** to **t1**. Copy and paste this line and name the second turtle **t2**.

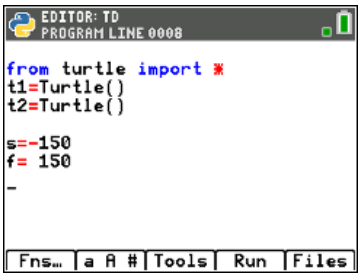
t1 = Turtle()
t2 = Turtle()

When you select any turtle function from the menu you now have to modify the names of the turtles to suit this program.

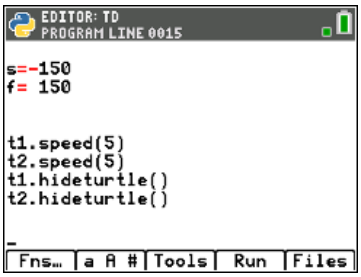


3. Since the turtle window domain is -159 to 159, set the **s**tartline and **f**inishline to be near the left and right edge of the screen:

s = -150
f = 150

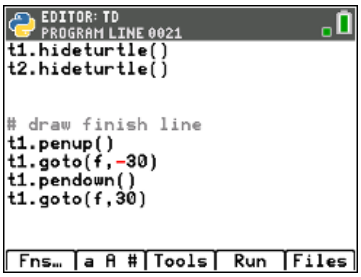


4. Now do some turtle housekeeping: set the speed of each turtle to the same value (you can adjust these later) and hide both turtles.



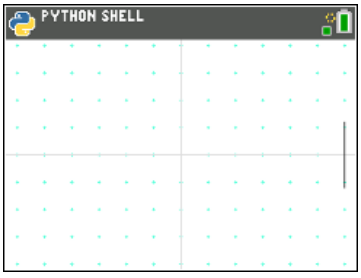
5. Use one of the turtles to make the finish line:

```
# draw finish line
t1.penup()
t1.goto(f, -30)
t1.pendown()
t1.goto(f, 30)
t1.penup()
```



Remember that throughout this program, after selecting a turtle function from the menus, you must edit the turtle name by adding a 1 or 2 to the turtle name **t.** *as necessary.*

6. You can test the program now to see that the finish line is showing on the right side of the screen as seen here.



7. Place the two turtles at the starting line and get them ready to race. Place one turtle above the x-axis and one below the x-axis Our two 'lanes' are y = 10 and y = -10. You can choose other lanes.

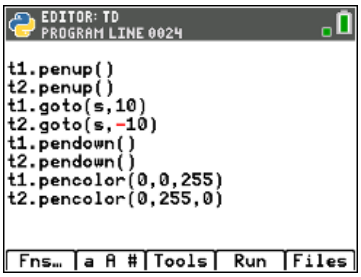
   Make sure both turtle pens are up before moving them.
   Use **t.goto()** to place the turtles at the start line.

   t1.goto(s, 10)
   t2.goto(s, -10)

   t.goto *is on the* **turtle… Move** *menu.*
   After the turtles have been moved, put the pens down again.

   You can also set the pen color of each turtle. **t.pencolor()** is on **Turtle Graphics > Pen Control**

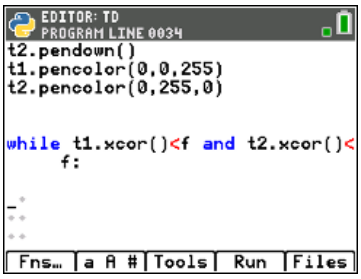
   Think: in what direction are the (hidden) turtles now facing?
   t.goto() *does not affect the heading.*



8. We are ready to start the race!

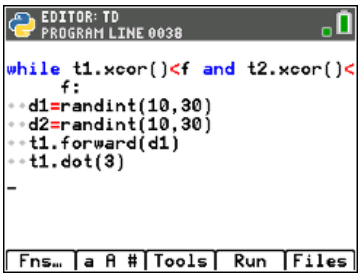
   while t1.xcor() < f and t2.xcor() < f:

   This **while** loop (the race) ends when one of the turtles crosses the finish line (**f**). The function **t.xcor()** gives a turtle's x-coordinate and is found on the **turtle… > State** menu:



9. Generate two *random** values: **d1** is the distance turtle 1 will move and **d2** is the distance turtle 2 will move (in pixels). But they cannot both move at the same time! We let turtle 1 move first. (*more about this later*). Make a small dot at the turtle's new position.

   while t1.xcor() < f and t2.xcor() < f:
   d1=randint(10, 30)
   d2=randint(10, 20)
   t1.forward(d1)
   t1.dot(3)

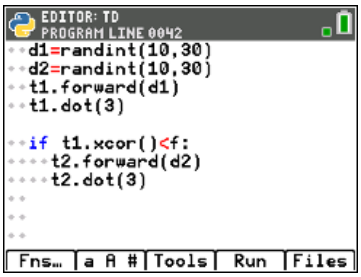


   Remember to add* **from random import randint *at the top of your program.*

10. Down the track, we make sure that turtle 1 has not crossed the finish line before letting turtle 2 make a move.

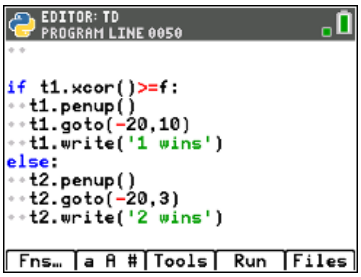
```
if t1.xcor() < f:
    t2.forward(d2)
    t2.dot(3)
```

This is the end of the **while** loop. Test the program again to see the turtles race across the screen.



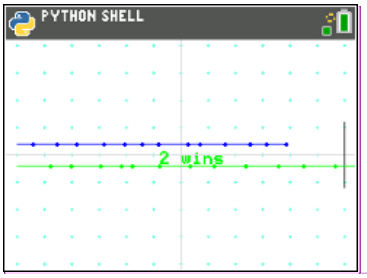
11. After the **whle** loop ends, report the winner using an **if…else** structure:

```
if t1.xcor() >= f:
  t1.penup()
  t1.goto(-20,10)
  t1.write('1 wins')
else:
  t2.penup()
  t2.goto(-20,3)
  t2.write('2 wins')
```



12. Test your program now to see that each turtle can win sometimes and that the code is working properly.

Note: If you do not see the grid, install the appvar **GRID.8xv** *supplied with the* **TURTLE.8xv** *module onto your calculator.*



Comment [KC1]: CX screenshot?

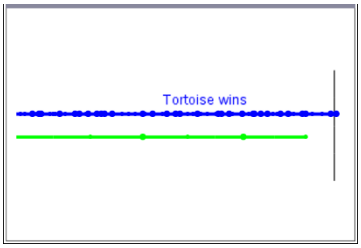
13. Challenges:

- Is it 'fair' that turtle 1 always goes first? Build a routine that chooses a turtle to go first at random. Does it make a difference in the outcome? Should it be just the first move or every move that is random?
- **Analyze the program:** Embed the setup functions and the race in another

  while get_key() != "esc":

  loop, change the turtle speeds to 0, keep track of the number of wins for each turtle and display the them on the screen. Let the program run for awhile to see if the race is 'fair'.
- **Tortoise v. Hare**

  Tortoise and Hare are having a race. The slow but steady Tortoise takes small steps and the lazy Hare takes giant leaps. For every ten small (*random*) steps Tortoise takes, Hare makes one giant (*random*) leap. Create a race program between Tortoise and Hare in which each wins some of the races. Analyze your code as above to see how 'fair' your code is.



Teacher Notes: Sample code *(note longer variable names)*

```
from random import randint
from turtle import *
t1=Turtle()
t2=Turtle()

startline = -150
finishline=   150

t1.hideturtle()
t2.hideturtle()
t1.hidescale()
t1.speed(9)
t2.speed(9)

# draw the finish line
t1.penup()
t1.goto(finishline,-50)
t1.pendown()
t1.goto(finishline,50)
t1.penup()
t2.penup()

# go to starting line
t1.goto(startline,10)
t2.goto(startline,-10)
t1.pendown()
t2.pendown()
t1.pencolor(0,0,255)
t2.pencolor(0,255,0)

# race here...
while t1.xcor()<finishline and t2.xcor()<finishline:
  d1=randint(10,30)
  d2=randint(10,30)
  t1.forward(d1)
  t1.dot(1)
  if t1.xcor()<finishline:
    t2.forward(d2)
    t2.dot(1)

#end of race
if t1.xcor()>=finishline:
  t1.penup()
  t1.goto(-20,20)
  t1.write('1 wins')
else:
  t2.penup()
  t2.goto(-20,-30)
  t2.write('2 wins')
```