



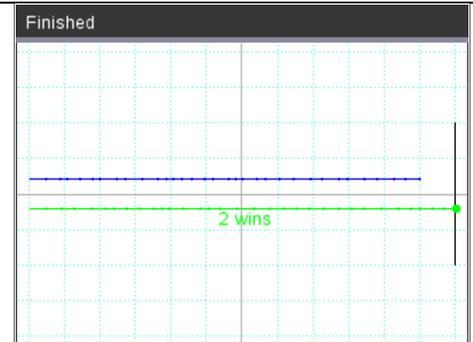
Turtle Graphics

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 (**turt_race.py**) and add the turtle module to the code by using [menu] > **More Modules > Turtle Graphics:**

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 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.





10 MOC: Python Modules

TI-NSPIRE™ CX II PYTHON

TURTLE: RACES

- Since the turtle window domain is -159 to 159, set the **startline** and **finishline** to be near the left and right edge of the screen:

```
startline = -150
finishline = 150
```

- Now do some turtle housekeeping: hide both turtles and set the speed of each turtle to the same value. Optionally, hide the scale in the lower left corner of the screen. Only one turtle needs to hide the scale.

- Use one of the turtles to make the finish line:

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

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

```
*turtle race
RAD
7/63
*turt_race.py
from turtle import *
t1=Turtle()
t2=Turtle()

startline = -150
finishline= 150
```

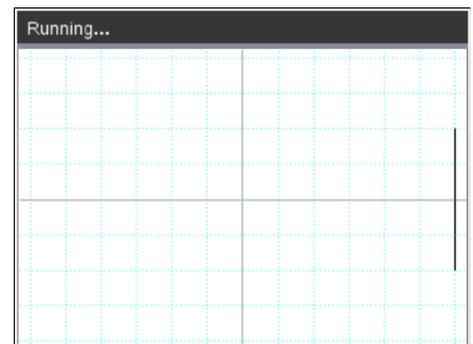
```
*turtle race
RAD
11/59
*turt_race.py

startline = -150
finishline= 150

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

```
*turtle race
RAD
18/61
*turt_race.py

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





10 MOC: Python Modules

TI-NSPIRE™ CX II PYTHON

TURTLE: RACES

- 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.

You can also set the pen color of each turtle. **t.pencolor()** is on

Turtle Graphics > Pen Control

After the turtles have been moved, put the pens down again.

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

- 8. We are ready to start the race...

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

This **while** loop ends when one of the turtles crosses the finish line.

The function **t.xcor()** gives a turtle's x-coordinate and is found on the **Turtle Graphics > Tell Turtle's state** menu.

Remember to add the digits to the turtle names.

- 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() < finishline and t2.xcor() < finishline:

d1=randint(5, 12)

d2=randint(5, 12)

t1.forward(d1)

t1.dot(1)

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

```

1.1 1.2 1.3 *turtle race RAD 27/60
*turt_race.py
t1.goto(finishline,50)
t1.penup()

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

```

```

1.1 1.2 1.3 *turtle race RAD 34/65
*turt_race.py
t2.pendown()
t1.pencolor(0,0,255)
t2.pencolor(0,255,0)

# race here...
while t1.xcor()<finishline and t2.xcor()<finishline:
++
|

```

```

1.1 1.2 1.3 *turtle race RAD 35/61
*turt_race.py
t1.pencolor(0,0,255)
t2.pencolor(0,255,0)

# race here...
while t1.xcor()<finishline and t2.xcor()<finishline:
++ d1=randint(5,12)
++ d2=randint(5,12)
++ t1.forward(d1)
++ t1.dot(1)
++

```



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() < finishline:
    t2.forward(d2)
    t2.dot(1)

```

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

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

```

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')

```

12. Test your program now to see that both turtles can win and that the code is working properly.

```

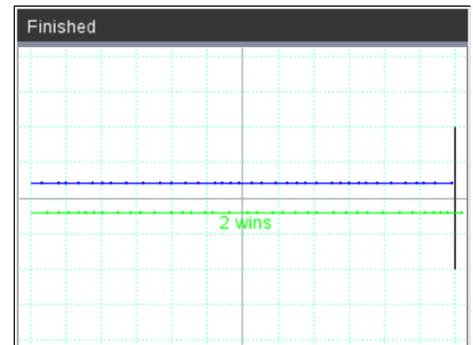
1.1 1.2 1.3 *turtle race RAD 41/63
*turt_race.py
while t1.xcor()<finishline and t2.xcor()<finishline:
  d1=randint(5,12)
  d2=randint(5,12)
  t1.forward(d1)
  t1.dot(1)
  if t1.xcor()<finishline:
    t2.forward(d2)
    t2.dot(1)
  |
#end of race

```

```

1.1 1.2 1.3 *turtle race RAD 44/59
#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')

```





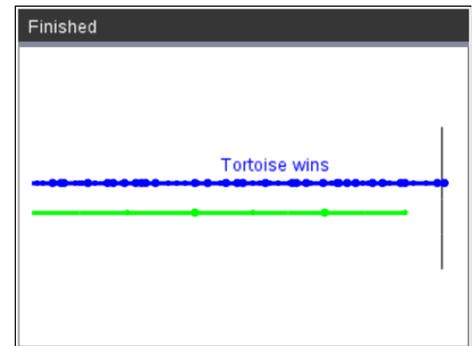
10 MOC: Python Modules

TI-NSPIRE™ CX II PYTHON

TURTLE: RACES

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?
- 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

```
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(5,12)
    d2=randint(5,12)
    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')
```