

Unit 5: The TI Modules

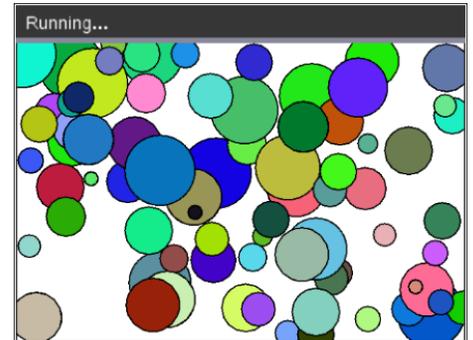
Skill Builder 3: Circles Everywhere!

In this lesson, you will make use of several separate included modules to create a 'screen saver' style animation.

**Objectives:**

- Use several imports
- Use random numbers
- Stop a program with a keypress
- Draw filled circles

Your next project makes random circles on the screen until a key is pressed. This is a sort of 'screen saver' (a type of program that was used in the old days to prevent 'burning in' of a CRT display by fixed images).



1. Start a new Python file using the Type: *Geometry Graphics*

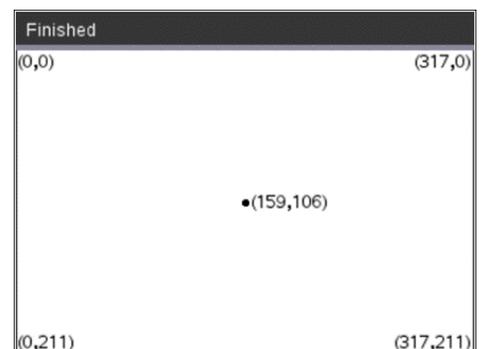
This template provides the **ti\_draw** module. You will also need two more modules in this project: one for random numbers and one for a 'press **esc** to end' feature. Random number functions are found in the **random** module and **get\_key** is in the **ti\_system** module, so import both modules. They are found on different **menus**.



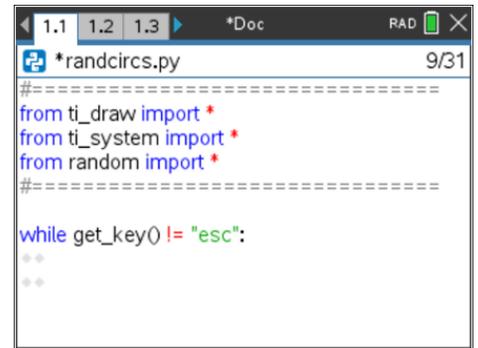
2. In this project, do *not* use **set\_window**. Use the 'default' graphics window where (0,0) is in the upper left corner. The screen is 318 x 212 pixels. The coordinates of the four corners and the center of the canvas are shown in this image.

An important difference between this screen and one designed using **set\_window** is:

In the statement **draw\_rect(x, y, width, height)**, the (x,y) pair is now the *upper-left* corner and the *height* value measures from *top-to-bottom* because the y-values increase going *down* the screen.



- The main program consists of a loop that stops when **esc** is pressed. This **while** statement is on **menu > More Modules > TI System**.



```

1.1 1.2 1.3 *Doc RAD X
*randcircs.py 9/31
#=====
from ti_draw import *
from ti_system import *
from random import *
#=====

while get_key() != "esc":
  **
  **
  **
  
```

- To draw random circles in random colors, we need to set up several variables with random values. For the circles we need values for *x*, *y*, *radius* and for the colors we need *red*, *green*, and *blue* values.

Each of these six variables is assigned like this one:

**x = randint(0,317)**

Assign each variable a random integer in an *appropriate* range. Keep in mind the screen dimensions and color value restrictions.



```

1.1 1.2 1.3 *Doc RAD X
randcircs.py 14/42
from random import *
#=====

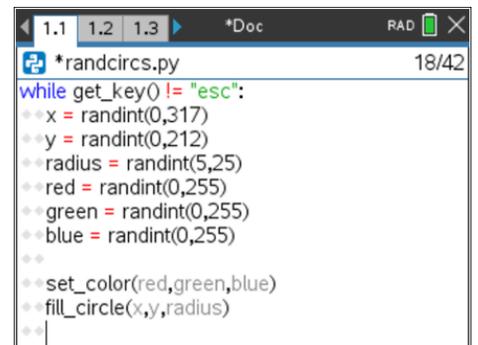
while get_key() != "esc":
  ** x = randint(0,317)
  ** y = randint(0,212)
  ** radius = randint(5,25)
  ** red = randint(0,255)
  ** green = randint(0,255)
  ** blue = randint(0,255)
  **
  **
  **
  
```

- Add the **set\_color()** and **fill\_circle()** functions from:

**menu > More Modules > TI Draw**

Do you see that the gray inline prompts for the arguments are the same words as the variable names? Unfortunately, you cannot leave them like that. The inline prompt *red* is *not* the variable *red*. You must type the variable names in place of the inline prompts.

Replace the prompts in these two statements with the variable names.



```

1.1 1.2 1.3 *Doc RAD X
*randcircs.py 18/42
while get_key() != "esc":
  ** x = randint(0,317)
  ** y = randint(0,212)
  ** radius = randint(5,25)
  ** red = randint(0,255)
  ** green = randint(0,255)
  ** blue = randint(0,255)
  **
  **
  ** set_color(red,green,blue)
  ** fill_circle(x,y,radius)
  **
  **
  
```

Run the program when ready and press **esc** to stop the program.

Feel free to modify the random number ranges. Suppose you want the circles to appear more reddish... what would you do?



6. And, just in case you want to control the *speed* of the drawing process, use the **sleep()** function in the time module again. Import sleep from the time module using

**from time import sleep**

and add **sleep(1)** at the bottom of the loop (after the **fill\_circle** function and still indented).

This statement pauses processing for 1 second. Too slow? You can make things go faster or slower by using smaller or larger numbers in **sleep()**.

7. If you look at the image at the beginning of this lesson, you'll notice that each circle has a black edge. Can you make that happen, too?

```
*randcircs.py 17/41
while get_key() != "esc":
  x = randint(0,317)
  y = randint(0,212)
  radius = randint(5,25)
  red = randint(0,255)
  green = randint(0,255)
  blue = randint(0,255)
  set_color(red,green,blue)
  fill_circle(x,y,radius)
  sleep(1)
```

