# Crafting Catchy Computer Art with Python Turtle

Doug Purcell

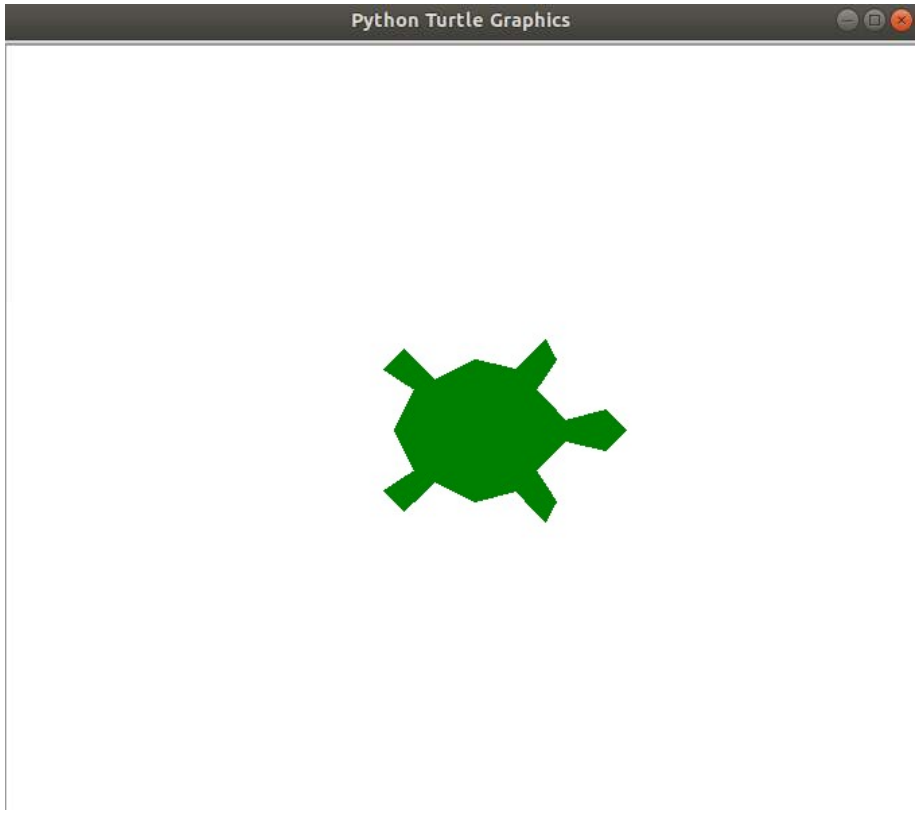
# I’m Doug Purcell

- A software engineer
- 4X published author
- A creative entrepreneur

# Why This Talk Can Unveil A New World of Possibilities!

- Perfect for educators that want to find a way for students to stay motivated in programming.
- Suitable for artists, designers, or creative folks that want to learn programming but can’t find the motivation to stick with it.
- Excellent for those who are completely new to programming and are having trouble mastering the fundamentals such as data types, data structures, iteration, and forming solutions.
- Suitable for new developers that want to learn algorithmic programming.
- Great for beginners that want to code something that they can showcase.

# Let’s Program This Image



# Code for the Big Green Turtle

```
import turtle

leonardo = turtle.Turtle()

leonardo.shape('turtle')

leonardo.shapesize(7.5, 7.5)

leonardo.color('green')

turtle.done()
```

## Line One

```
>>> import turtle
```

- Imports the turtle module so that we can use all of the builtin functionality.

## Line Two

```
>>> leonardo = turtle.Turtle()
```

- Once the module is imported we can go ahead and call the Turtle constructor.

## Line Three

```
>>> leonardo.shape('turtle')
```

- We can change the default shape of the turtle object to something more appealing like a green turtle using `shape`.

# Line Four

```
>>> leonardo.shapesize(7.5, 7.5)
```

- Stretches the turtle shape. The first parameter corresponds to the `width` while the second one corresponds to the `length`.

## Line Five

```
>>> leonardo.color('green')
```

- Sets the color of the turtle to green.

# Line Six

```
>>> turtle.done()
```

- Starts the event loop which in essence calls Tkinter’s main loop function.

Want to Explore all of the Cool Functionality of Turtle?

https://docs.python.org/3.3/library/turtle.html?highlight=turtle

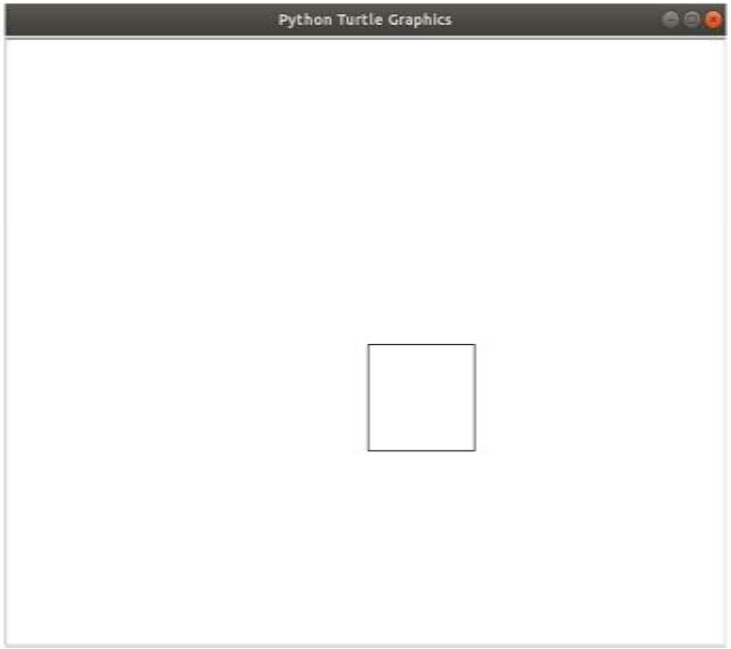
# How Would You Define a Square?

# A Square Has...

- Four sides of equal length
- Four right angles

How can we use this definition to code a square in turtle?

# A Simple Square Image in Python



# Here's The Code

```
import turtle

square = turtle.Turtle()

square.hideturtle()

square.forward(100)

square.right(90)

square.forward(100)

square.right(90)

square.forward(100)

square.right(90)

square.forward(100)

turtle.done()
```

# A lot of Repeated Statements in The Code

- 4 calls to `forward` and 3 calls to `right`

## Shorten The Code With a `for` Loop

```
import turtle

square = turtle.Turtle()

square.hideturtle()

for x in range(4):

      square.forward(100)

      square.right(90)

turtle.done()
```

# Adding Colors to Turtle Graphics

- Just like how film would be humdrum with traditional black-and-white television, the same concept applies to your turtle computer graphics.
- We can spice things up by adding color to our images. Turtle enables pythonistas the ability to do this via the color function.
- You can read about it in the turtle module here: https://docs.python.org/3.3/library/turtle.html?highlight=turtle#turtle.color
    - You can call it with no arguments, one argument, or two arguments. When you pass in one argument it returns the pencolor and current fillcolor as a pair; it doesn't actually manipulate the color, just returns what color is in the turtle object.

## This is Great, But...

Sometimes when you're creating images you're not exactly sure what colors to use… Ta-da, this is where the `PyRandomColor` comes to the rescue.

# The `PyRandomColor` Open source Module

An open source module that I created under the GPL license that allows pythonistas to randomly add colors to their turtle graphics.

There's two simple ways to download `PyRandomColor`:

- $ pip install `pyrandomcolor`
- $ git clone **https://github.com/purcellconsult/PyRandomColor.git**

# A list of functions from PyRandomColor

`get_random_color()`: Picks any random color.

`whites_and_pastels()`: Returns a random color that's within the white and pastel color scheme.

`grays()`: Returns a random color that's within the gray color scheme.

`blues()`: Returns a random color within the blue color scheme.

`greens()`: Returns a random color within the green color scheme.

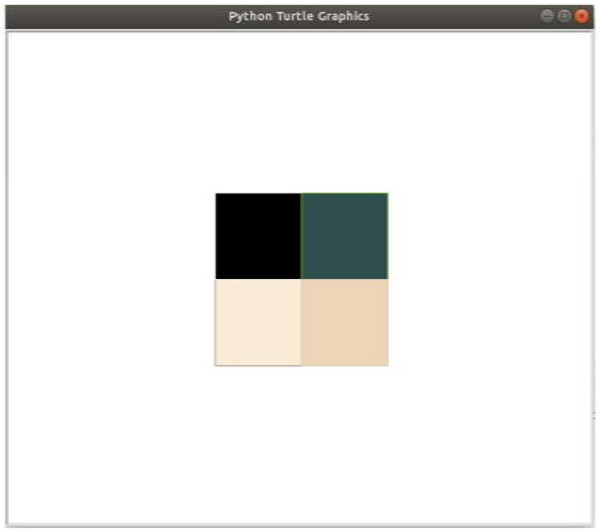
`yellows()`: Returns a random color within the yellow color scheme.

`browns()`: Returns a random color within the brown color scheme.

`oranges()`: Returns a random color within the orange color scheme.

`pinks_violets()`: Returns a random color within the pinkish and *violetish* color scheme.

# Let’s Create a Square Portrait Like The Below

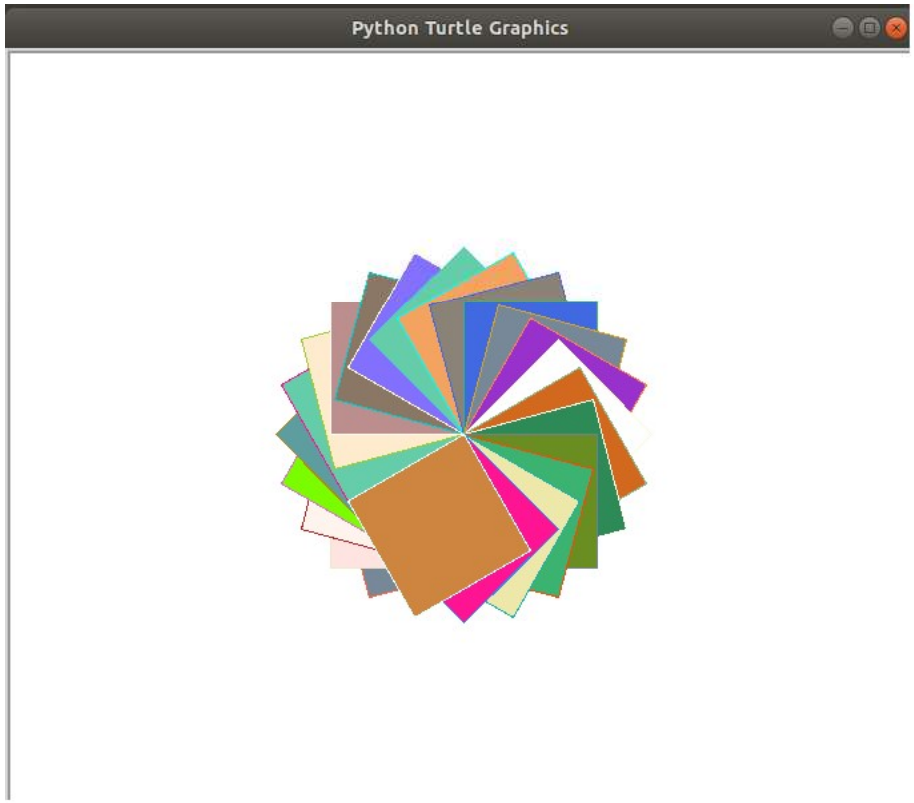


# Let’s See This in Action

```
$ python multi_color_cube.py
```

```
import turtle

from random_colors import get_random_color

square = turtle.Turtle()

square.hideturtle()

for x in range(4):

            square.color(get_random_color(), get_random_color())

            square.begin_fill()

            square.right(90)

            for y in range(4):

                square.forward(100)

                square.right(90)

            square.end_fill()

turtle.done()
```

# More Square Artwork

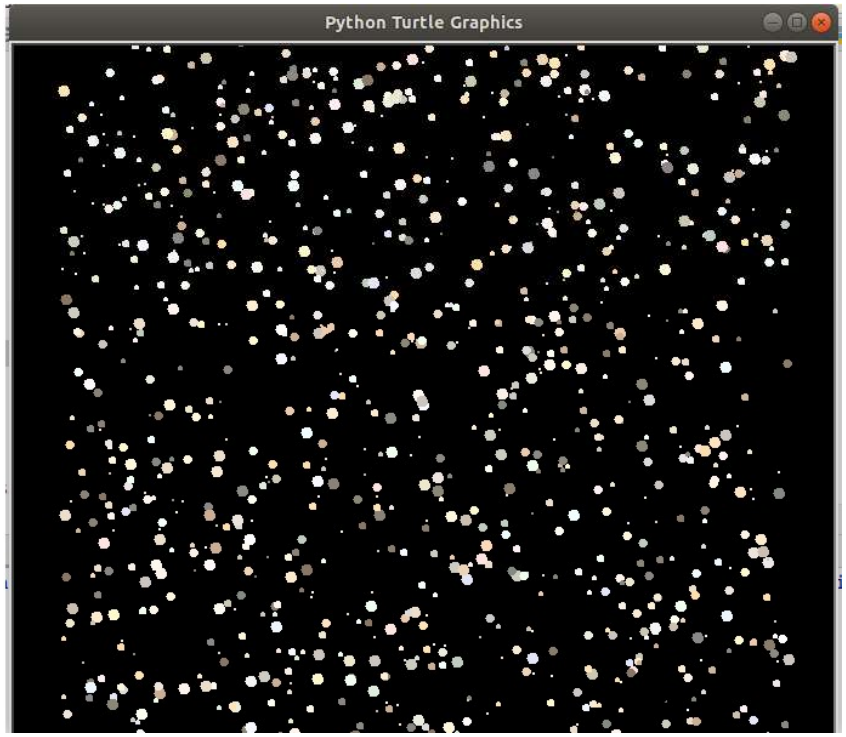


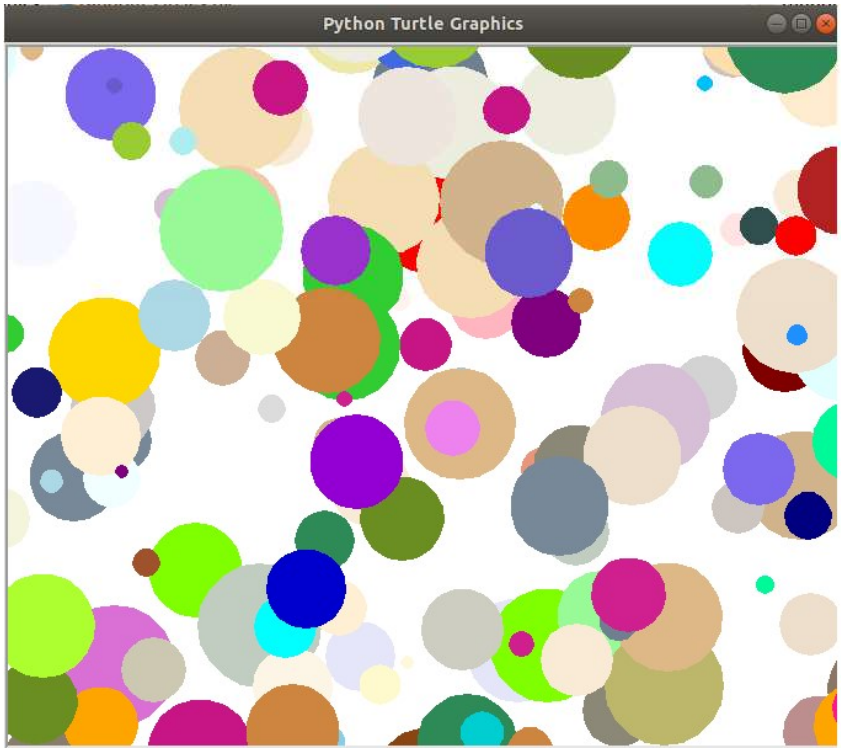
```
import turtle

from random_colors import get_random_color

square = turtle.Turtle()

square.hideturtle()

square.speed(0)

for x in range(50):

            square.color(get_random_color(), get_random_color())

            square.begin_fill()

            square.right(35)

            for y in range(4):

                square.forward(100)

                square.right(90)

                square.end_fill()

turtle.done()
```

# Let’s Move On to Another Shape

How about  a circle?

# We'll Create Cool Artwork Like The Following

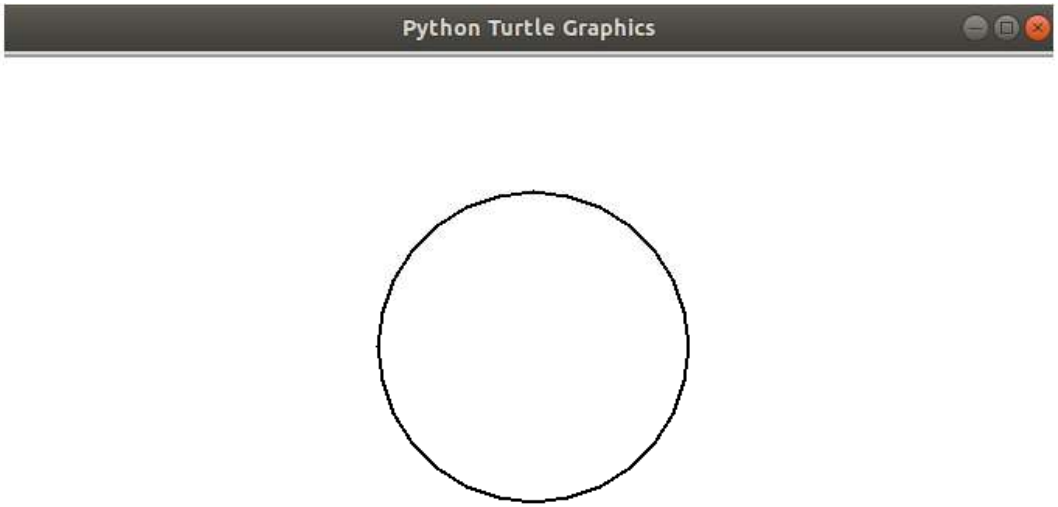


# How to Create a Circle in Python Turtle

```
turtle.circle(radius, extent=None,
steps=None)
```

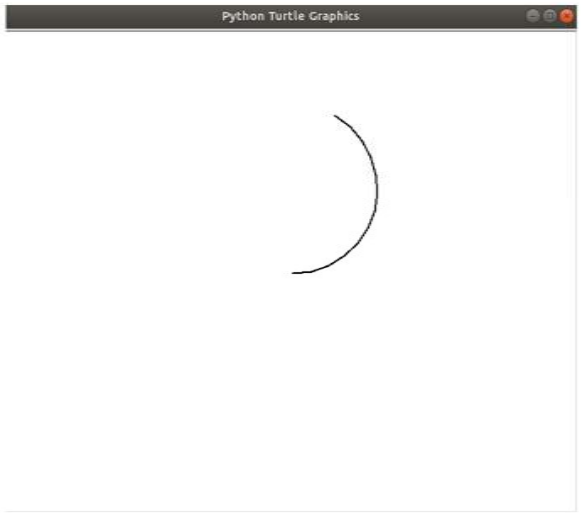
# How to Create a Simple Square in Turtle?



# Code

```
import turtle

x = turtle.Turtle()

x.hideturtle()

x.pensize(2)

x.circle(100)

turtle.done()
```

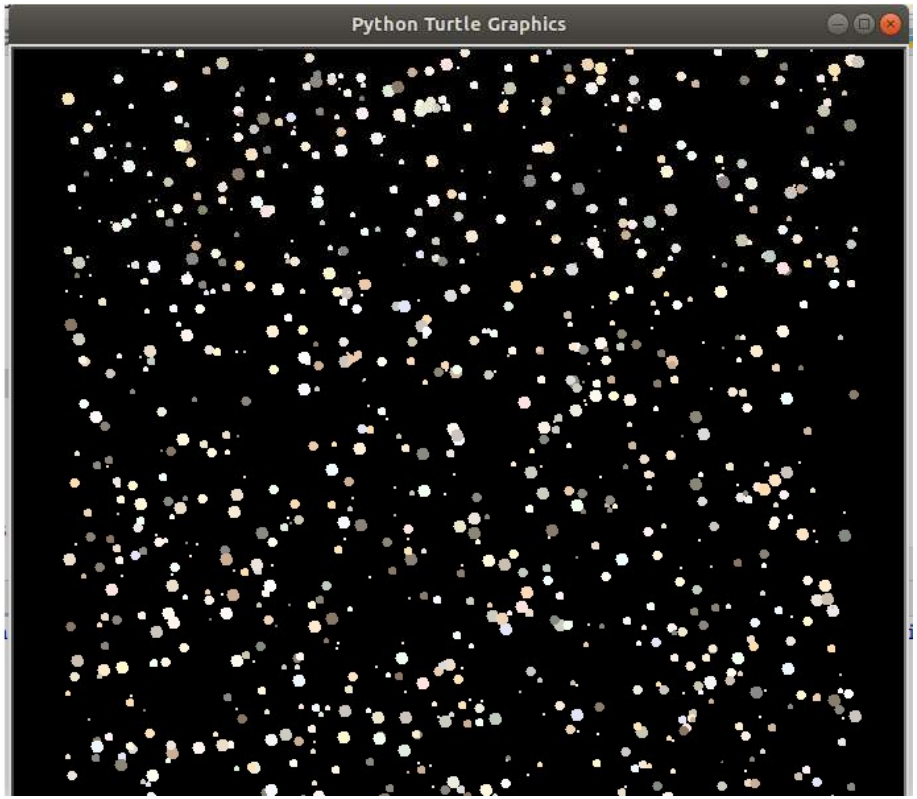
# To Create an Arc



# Do The Following

```
import turtle

x = turtle.Turtle()

x.hideturtle()

x.pensize(2)

x.circle(100, extent=150)

turtle.done()
```

# Ok, So How Do We Code A Randomly Generated Night Sky?



## Demo of the Night Sky

```
$ python night.py
```

# Our Checklist

What we know thus far:

- How to create circles
- How to randomly color objects

What we don’t know:

- How to change the background color of the screen
- How to randomly position various circles on the screen

How to change the background color?

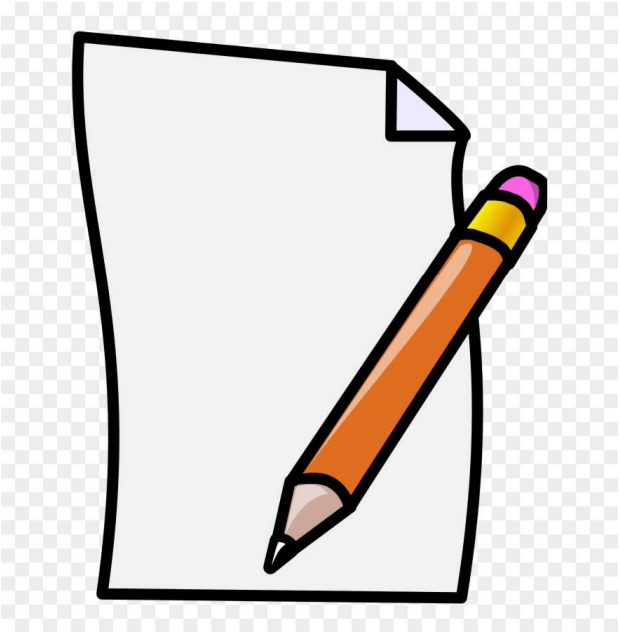
```
>>> turtle.Screen().bgcolor('black')
```

## How to randomly position circles on the screen?

There are three methods in turtle that can help out with this:

- `penup`: Picks the pen up, no drawing happens.
- `pendown`: Puts the pen down.
- `goto(x, y=None)`: Move the turtle to the x and y coordinates.

# Imagine That You're Drawing



## What happens when...

You’re chilling at your desk drawing a masterpiece on a blank piece of paper. What would happen if you decide to pick your pen up and move it to a different position on that same paper? The pen will now appear at the new spot. When you put the tip of your pen back down this is when the drawing happens.

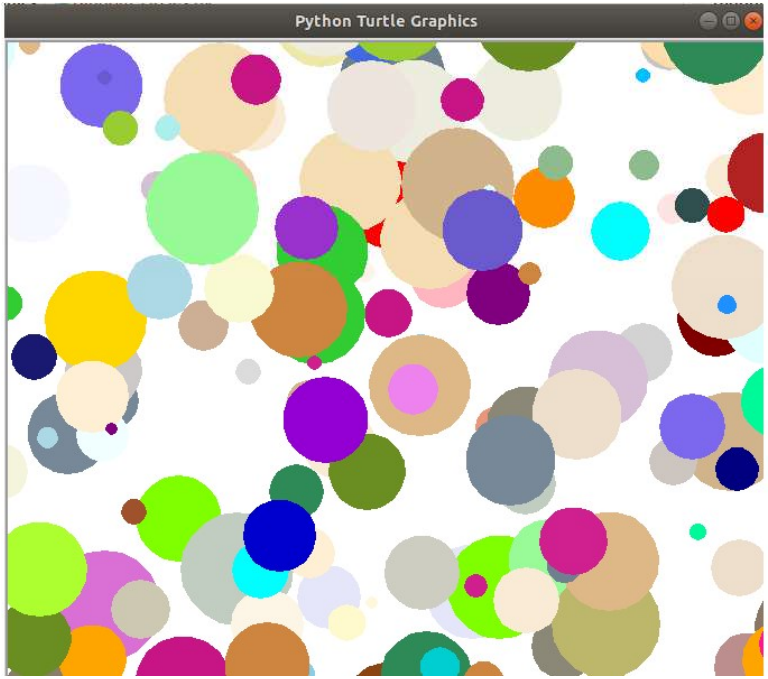
## Similar logic in Turtle

The `penup` and `pendown` methods operate in a similar fashion to how you draw on a piece of paper. When the `penup` function is called the pen is in the air, and when the `pendown` function is called that is like placing the tip of your pen on the paper to start drawing. These two functions are important because when you use the goto function, if you don't call the `penup` function then nasty squiggly lines will be drawn along the way, like if you're sketching a stickman on a piece of paper.

# Night Sky Solution

```
def create_night_sky(stars=1000):
    sky = turtle.Turtle()
    back_ground = turtle.Screen()
    back_ground.bgcolor('black')
    num = stars
    sky.speed(0)
    for x in range(num):
        sky.color(whites_and_pastels())
        sky.begin_fill()
        sky.penup()
        sky.goto(randint(-300, 300), randint(-300, 300))
        sky.circle(randint(1, 5))
        sky.pendown()
        sky.end_fill()
    turtle.done()
```

# More Circle Art

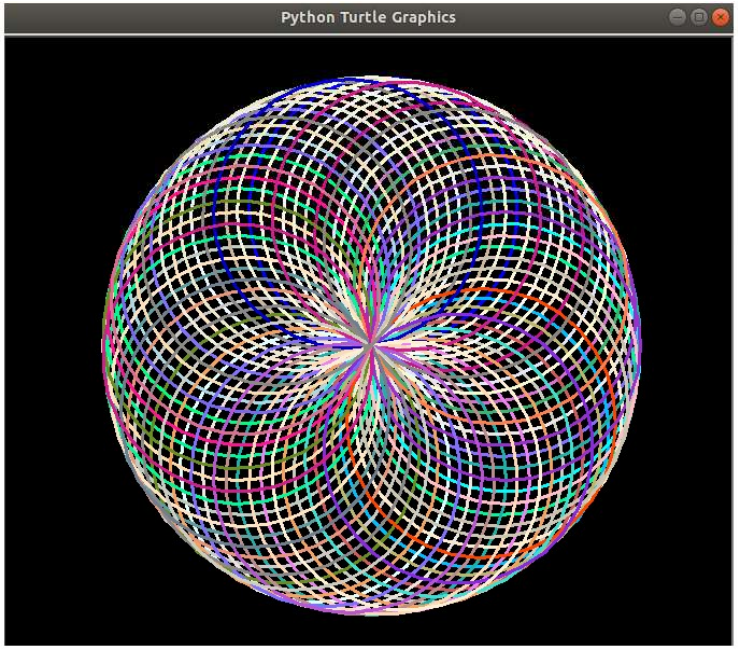


## Demo the Circle Ar

```
$ python random_circles.py
```

Source Code here:
https://github.com/purcellconsult/Code-Cool-Stuff-With-Python/blob/master/sourcecode/ch_04/random_circles.py

# Let’s Create a Spirograph



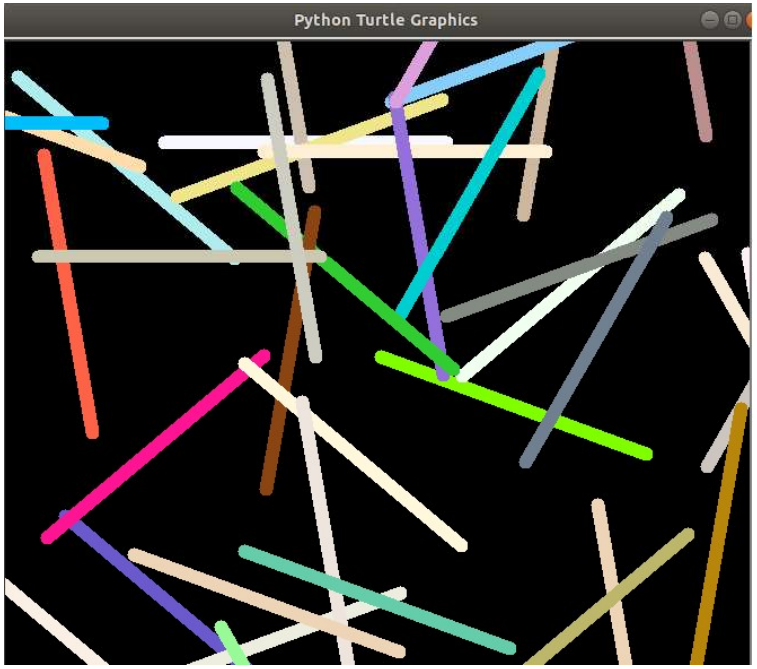
## Show Demo

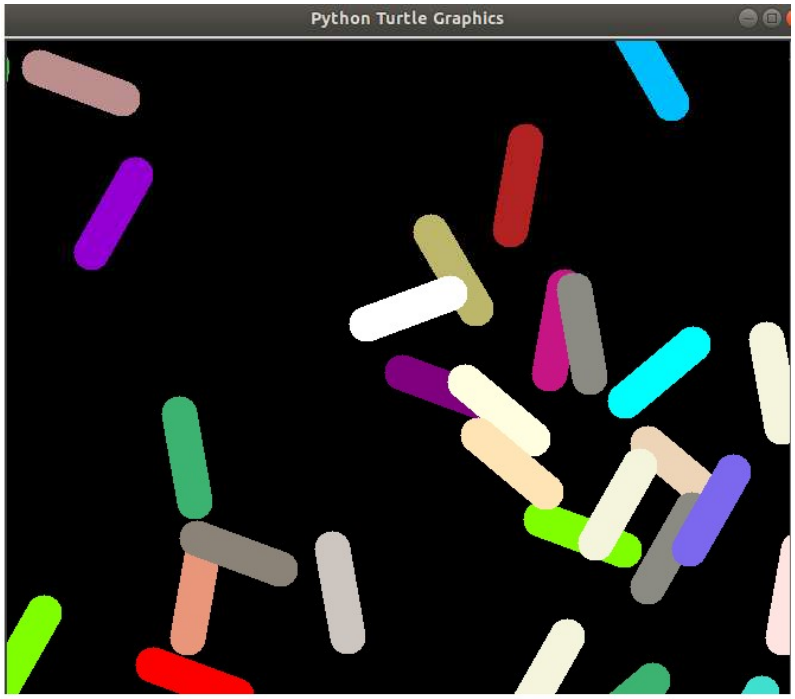
```
$ python spirograph_1.py
```

# The Solution

```
import turtle

from random_colors import get_random_color

def create_circles(cycles=100):

    turtle.bgcolor('black')

    turtle.pensize(3)

    turtle.speed(0)

    for i in range(cycles):

        turtle.color(get_random_color())

        turtle.circle(125)

        turtle.right(25)

    turtle.done()
```

# Creating Artwork With Lines in Turtle



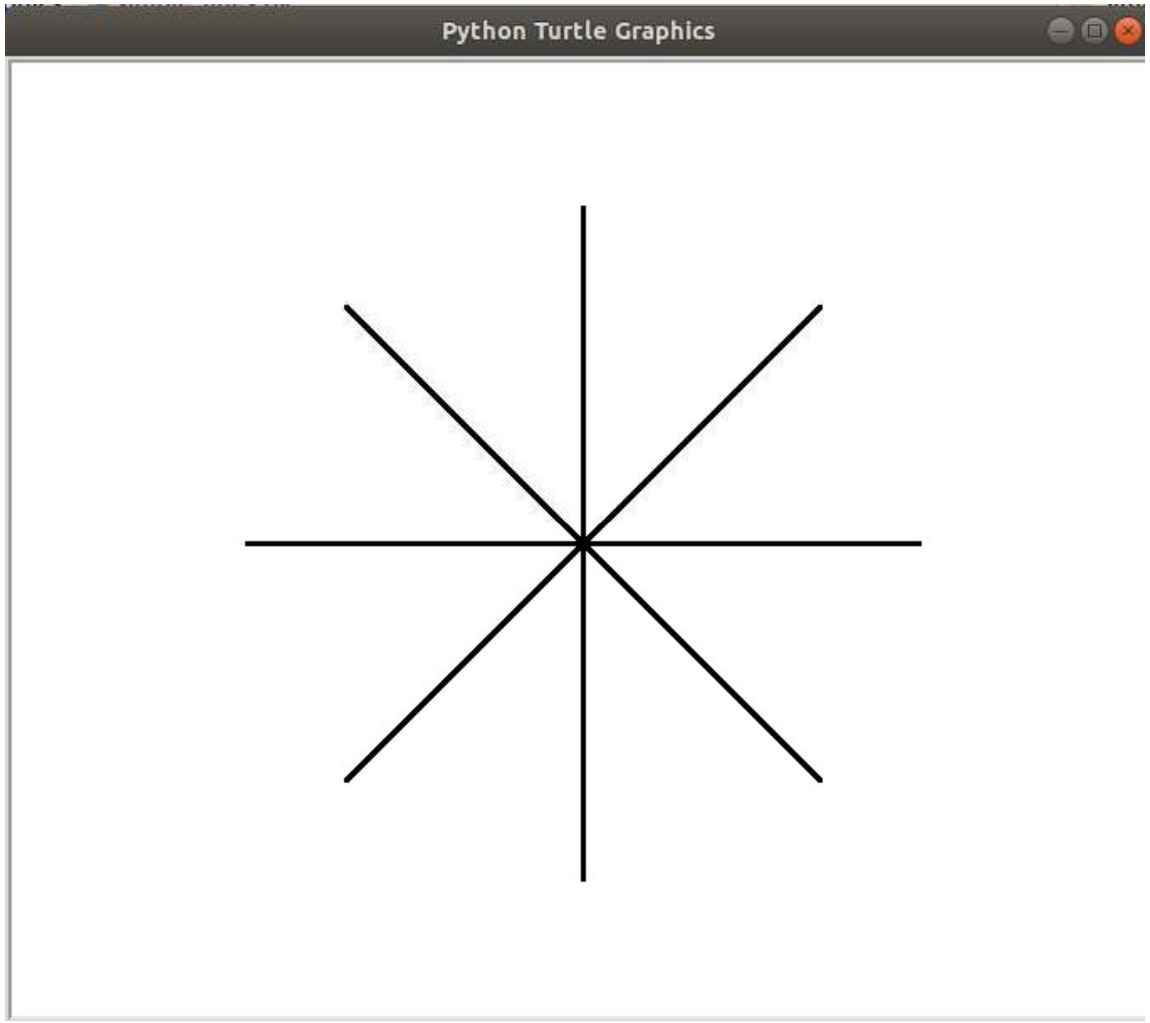



# How to Create Lines in Turtle?

- Simply use the `forward` function

# What Image Does the Following Produce?

```
import turtle

line = turtle.Turtle()

line.hideturtle()

line.pensize(3)

for x in range(8):

          line.forward(200)

          line.goto(0, 0)

          line.left(45)

turtle.done()
```


Python Turtle Graphics

# Demo Party Lights

```
$ python party_lights.py
```

# Solution

https://github.com/purcellconsult/Code-Cool-Stuff-With-Python/blob/master/sourcecode/ch_04/party_lights.py

## Show mike and ike demo

```
$ python mike_and_ike_candies
```

## View the source code

https://github.com/purcellconsult/Code-Cool-Stuff-With-Python/blob/master/sourcecode/ch_04/mike_and_ike_candies.py

# One Last Thing to Do With Turtle!

Let’s create a simple racing game.

# Let's Call The Game *4 Little Turtles*

a) Create a game that consists of 4 little turtles with unique colors.

b) Include a countdown that starts the game.

c) Create a graphical finish line with the text "Finish Line" directly above it.

d) Randomly generate the speed of the turtles.

e) Write an algorithm to accurately determine which turtle crossed the finish line first.

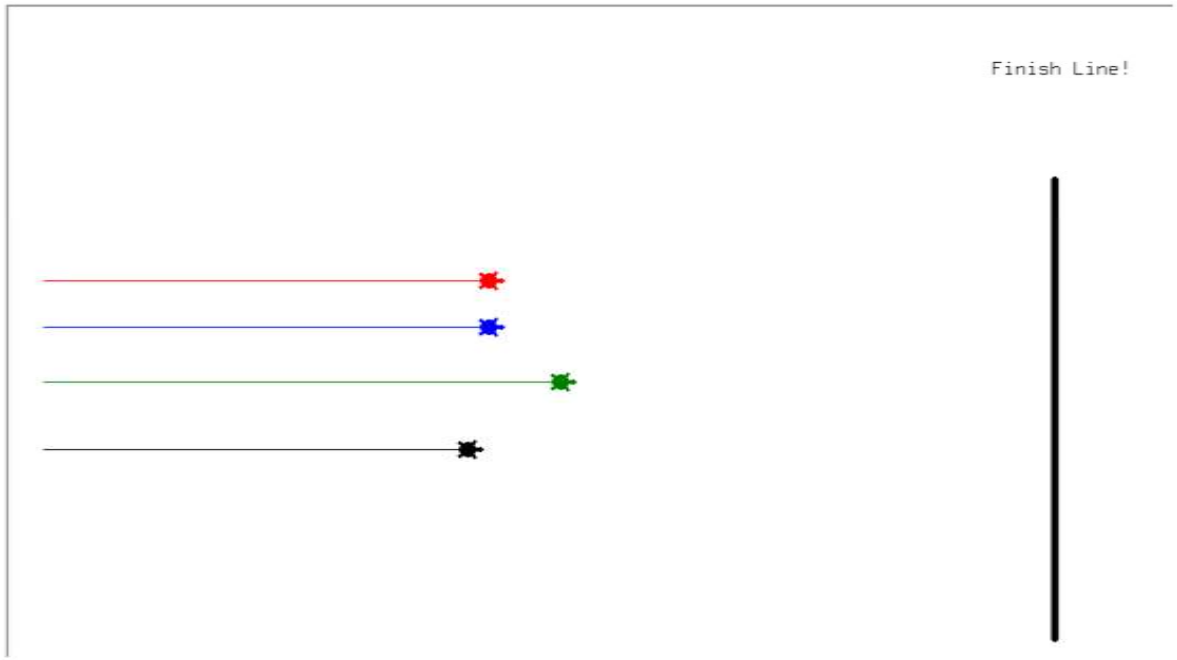
f) Write a message to let the user know which turtle won.

g) Write a simple algorithm to make the winning turtle spin and then grow in size afterwards.

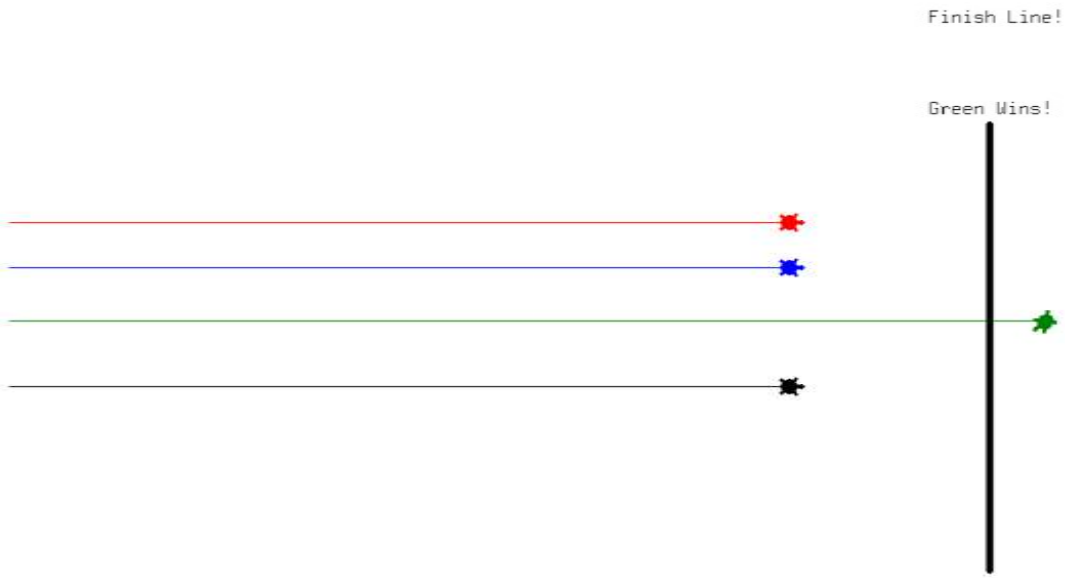
h) Have fun playing and showing the game off to your friends and family!

# Screenshot 1

# Screenshot 2



# Screenshot 3



# Let’s View The Demo

```
>>> python turtle_racing_game.py
```

Can anyone guess the winner?

# Let's Define Our Functions

`__init__`: The constructor which creates the four turtle objects along with getting the height and width of the screen.

`start_turtles`: Sets the default positions of the turtles.

`finish_line`: Draws the finish line in the racing game and aligns it on the right hand portion of the screen.

`countdown_timer`: Starts the countdown timer for the script starting at 5. Use the `sleep` function from the time module to pause the program.

`set_turtle_speeds`: Randomly sets the speed of the turtles within the range of 1-25.

# Study The Full Source Code

https://github.com/purcellconsult/Code-Cool-Stuff-With-Python/blob/master/sourcecode/ch_04/turtle_racing_game.py

The End :-)