**Step 1: Drawing Basic Shapes With Python Turtle**

1. Create an new Repl by selecting the **“Python with Turtle”** language / environment.



1. Begin all of your turtle programs with the following code to create a “pen”:

import turtle

myPen = turtle.Turtle()

1. Review the following chart for a list of Turtle commands.



1. Use the following program to draw a red square.





1. Switch to the “Result” window to see the square.
2. Create a program to draw any one of the shapes “b”, ”d”, or “e” shown in the figures below.   
   Provide a listing of your program code.
3. Create a program to draw any one of the shapes “c”, or “f” shown in the figures below.   
   Provide a listing of your program code.



**Step 2: Christmas / Winter Theme Card**

1. Use your creativity to create a card design using Turtle.
   1. The design must have multiple figures.
   2. The design must have at least two different patterns.
   3. You may repeat patterns.
   4. Provide a listing of your program code.
   5. Provide an image of your program result.

import turtle

import random

web\_based = True

if web\_based:

i\_scale = 1.5

snow\_size = 4

snow\_speed = 3

draw\_speed = 10

rate\_of\_snow\_balls = 6

else:

i\_scale = 1

snow\_size = 7

snow\_speed = 2

draw\_speed = 10

rate\_of\_snow\_balls = 2

width = 600 / i\_scale

height = 600 / i\_scale

screen = turtle.Screen()

if not web\_based:

screen.setup(width, height)

screen.title("Happy Christmas from Simardeep")

def make\_triangle(x, y, size, outline, triangle):

triangle.hideturtle()

triangle.penup()

triangle.setposition(x, y)

triangle.pensize(3)

if outline:

triangle.pendown()

if not outline:

triangle.fillcolor("forest green")

triangle.begin\_fill()

triangle.setposition(x + size, y - size)

triangle.setposition(x - size, y - size)

triangle.setposition(x, y)

if not outline:

triangle.end\_fill()

def make\_ball(x, y, size, colour, ball):

ball.hideturtle()

ball.penup()

ball.setposition(x, y)

ball.color(colour)

ball.dot(size)

def move\_snow(snow):

position = snow.position()

snow.clear()

make\_ball(position[0], position[1] - snow\_speed, snow\_size, "white", snow)

def snow\_fall():

rand\_make\_snow = random.randint(0, rate\_of\_snow\_balls)

if rand\_make\_snow == 0:

snow = turtle.Turtle()

snow.hideturtle()

snow.penup()

list\_of\_snow.append(snow)

make\_ball(random.randint(-width / 2, width / 2), width / 2, snow\_size,

"white", snow)

for snow in list\_of\_snow:

move\_snow(snow)

if snow.position()[1] <= -width / 2:

snow.clear()

list\_of\_snow.remove(snow)

del snow

screen.update()

triangle\_1 = turtle.Turtle()

triangle\_1.speed(draw\_speed)

outline = True

for repeat in range(2):

make\_triangle(0, width / 3, width / 6, outline, triangle\_1)

make\_triangle(0, width / 4, width / 4, outline, triangle\_1)

make\_triangle(0, width / 8, width / 3, outline, triangle\_1)

outline = False

screen.tracer(0)

stem = turtle.Turtle()

stem.penup()

stem.hideturtle()

stem.setposition(-width, -width / 3)

stem.color("white")

stem.begin\_fill()

stem.setposition(width, -width / 3)

stem.setposition(width, -width / 2)

stem.setposition(-width, -width / 2)

stem.end\_fill()

screen.update()

stem.color("brown")

stem.setposition(-width / 30, -width / 4.8)

screen.tracer(1)

stem.pendown()

stem.begin\_fill()

stem.setposition(width / 30, -width / 4.8)

stem.setposition(width / 30, -3 \* width / 8)

stem.setposition(-width / 30, -3 \* width / 8)

stem.setposition(-width / 30, -width / 4.8)

stem.end\_fill()

screen.bgcolor("sky blue")

screen.tracer(2)

ball\_colours = ["red", "red", "red", "gold", "violet", "white"]

ball\_positions = [(-width / 30, width / 4), (3 \* width / 40, width / 5),

(-width / 20, width / 6), (width / 30, width / 9),

(-width / 12, width / 30), (width / 12, width / 24),

(-width / 9, -width / 20), (width / 8, -width / 15),

(0, -width / 6), (-width / 6, -width / 6),

(width / 5, -width / 7.5)

]

for position in ball\_positions:

make\_ball(position[0], position[1], 20 / i\_scale,

random.choice(ball\_colours),

turtle.Turtle())

screen.update()

list\_of\_snow = []

screen.tracer(0)

for \_ in range(50):

snow\_fall()

text\_1 = turtle.Turtle()

text\_1.hideturtle()

text\_1.penup()

text\_1.setposition(0, width / 2.7)

text\_1.color("red")

text\_1.write("Merry Christmas",

font=("Apple Chancery", max(int(30 / i\_scale), 15), "bold"),

align="center")

for \_ in range(25):

snow\_fall()

text\_1.setposition(width / 60, -width / 2.18)

text\_1.color("black")

text\_1.write("from",

font=("Apple Chancery", max(int(20 / i\_scale), 10), "normal"),

align="center")

if web\_based:

text\_1.setposition(width / 6, -width / 2.14)

else:

text\_1.setposition(width / 7.5, -width / 2.14)

text\_1.color("red")

text\_1.write("Simar", font=("Avenir", max(int(30 / i\_scale), 15), "normal"),

align="center")

if web\_based:

text\_1.setposition(width / 2.95, -width / 2.14)

else:

text\_1.setposition(width / 3.85, -width / 2.14)

text\_1.color("forest green")

text\_1.write("Anand", font=("Avenir", max(int(30 / i\_scale), 15), "normal"),

align="center")

if web\_based:

for \_ in range(200):

snow\_fall()

else:

while True:

snow\_fall()

turtle.done()