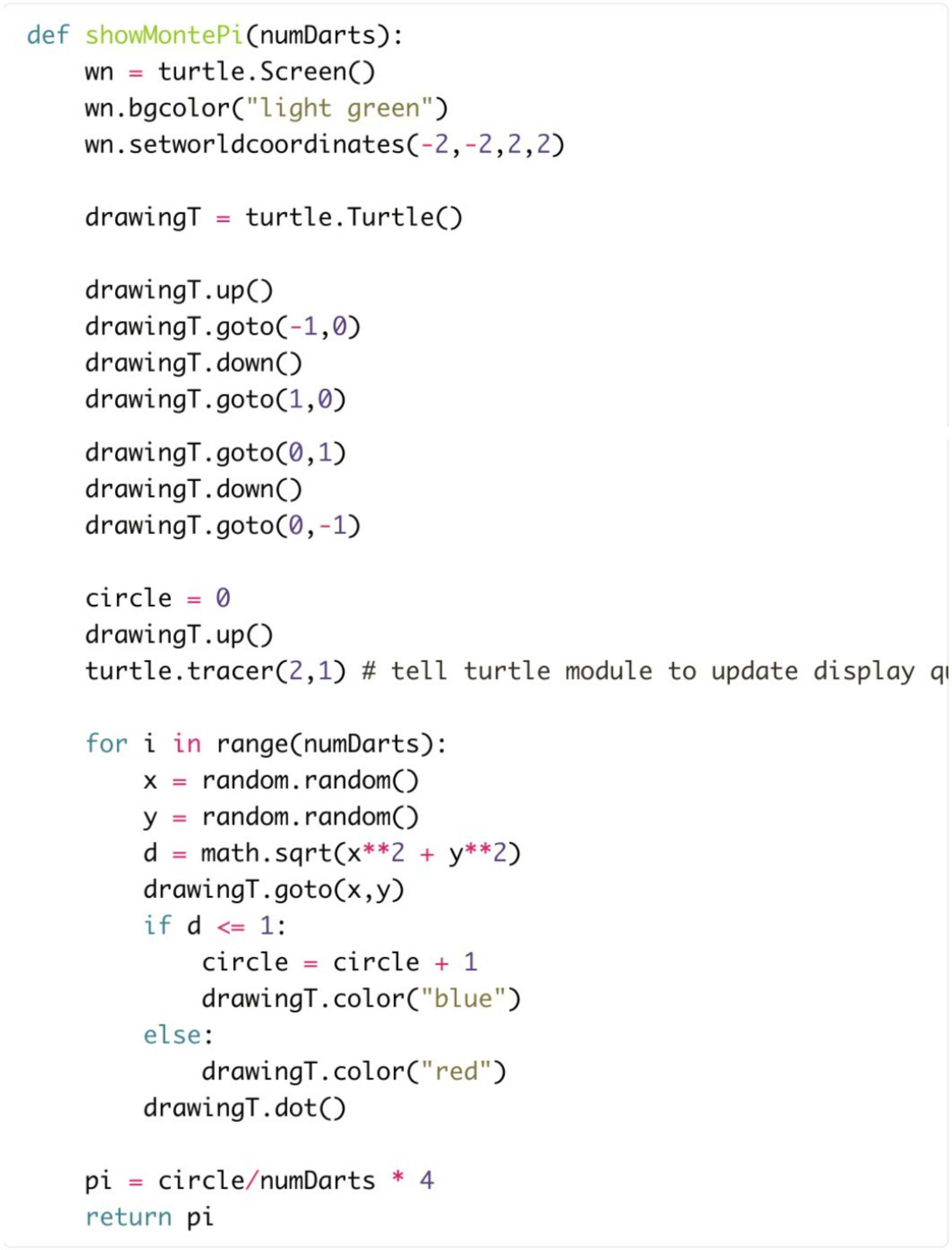
Activity: Pi – Monte Carlo Approximation

1. Create a file named **monteCarlo.py** and at the top import the libraries: **math**, **random**, and **turtle**
2. Add the function below to the file

```python
def montePi(numDarts):
    inCircle = 0
    for i in range(numDarts):
        x = random.random()
        y = random.random()
        d = math.sqrt(x**2 + y**2)
        if d <= 1:
            inCircle = inCircle + 1
    pi = inCircle/numDarts * 4
    return pi
```

3. Call **montePi** function with **numDarts**=100 and print the result. You should get a number close to Pi. Run your program for several times with the same number of darts, **numDarts**=100, do you get the same approximation of Pi? Why? Write your answer in comments.

4. Let’s use **turtle** to visualize the simulation! Add inside **monteCarlo.py** the following function and call it with 50 darts.



```python
def showMontePi(numDarts):
    wn = turtle.Screen()
    wn.bgcolor("light green")
    wn.setworldcoordinates(-2,-2,2,2)

    drawingT = turtle.Turtle()

    drawingT.up()
    drawingT.goto(-1,0)
    drawingT.down()
    drawingT.goto(1,0)
    drawingT.goto(0,1)
    drawingT.down()
    drawingT.goto(0,-1)

    circle = 0
    drawingT.up()
    turtle.tracer(2,1) # tell turtle module to update display q

    for i in range(numDarts):
        x = random.random()
        y = random.random()
        d = math.sqrt(x**2 + y**2)
        drawingT.goto(x,y)
        if d <= 1:
            circle = circle + 1
            drawingT.color("blue")
        else:
            drawingT.color("red")
        drawingT.dot()

    pi = circle/numDarts * 4
    return pi
```

5. Repeat the animation again but with 100 darts.
6. Let’s involve the whole square and circle in our animation! Replace the **randome.random()** by **randome.uniform(-1, 1)** and call again with 100 darts.
7. Add a line in your **showMontePi** function that will change the color of the background. (See pg. 75 or the online Python documentation if you do not remember how to do this.)
8. Explain in comments what the following command from s**howMontePi** accomplishes: **wn.setworldcoordinates(-2,-2,2,2)**
9. Adjust the world coordinates so that the window contains only the upper-right quadrant of the circle. Of course after you revert back to using **random.random()** instead of **uniform()**.