Phys105-AdventQuiz

December 7, 2020

1 Introduction to Computational Physics - Advent quiz

1.1 Advent calendar quiz contents

1.2 Advent calendar quiz

Predict the outcome of the following 25 snippets of code!

1.2.1 Window 1

[26]:
```
# <!-- Student -->
#
a = 14.0
print(a)
```

```
14.0
```

1.2.2 Window 2

[27]:
```
# <!-- Student -->
#
print(type(a))
```

```
<class 'float'>
```

1.2.3 Window 3

[28]:
```
# <!-- Student -->
#
b = 2
print(b)
```

```
2
```

1.2.4 Window 4

[29]:
```
# <!-- Student -->
#
print(type(b))
```

```
<class 'int'>
```

1.2.5 Window 5

[30]:
```
# <!-- Student -->
#
c = 5
print(c//2)
```

```
2
```

1.2.6 Window 6

[31]:
```
# <!-- Student -->
#
print(3%2)
```

```
1
```

1.2.7 Window 7

[32]:
```
# <!-- Student -->
#
i = 4
while i > 1:
    print(i)
    i = i - 1
```

```
4
3
2
```

1.2.8 Window 8

[33]:
```
# <!-- Student -->
#
i = 4
while i >= 1:
    print(i)
    i -= 1
```

```
4
3
2
1
```

1.2.9 Window 9

[34]:
```
# <!-- Student -->
#
x = 3
print(x > 3)
```

```
False
```

1.2.10 Window 10

[35]:
```
# <!-- Student -->
#
print(x <= 3)
```

```
True
```

1.2.11 Window 11

[36]:
```
# <!-- Student -->
#
a = False
b = True
```

```
print(a and b)
```

```
False
```

1.2.12 Window 12

[37]:
```
# <!-- Student -->
#
print(a or b)
```

```
True
```

1.2.13 Window 13

[38]:
```
# <!-- Student -->
#
myList = ["one", "two", "three", "four", "five"]
print(type(myList[2]))
```

```
<class 'str'>
```

1.2.14 Window 14

[39]:
```
# <!-- Student -->
#
print(myList[3])
```

```
four
```

1.2.15 Window 15

[40]:
```
# <!-- Student -->
#
for word in myList:
    print(word)
```

```
one
two
three
four
five
```

1.2.16 Window 16

[41]:
```
# <!-- Student -->
#
myTuple = ("zero", "one", "two", "three", "four", "five")
print(len(myTuple))
```

```
6
```

1.2.17 Window 17

[42]:
```
# <!-- Student -->
#
myTuple[3] = "seven"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-7946237c3e99> in <module>
      1 # <!-- Student -->
      2 #
----> 3 myTuple[3] = "seven"

TypeError: 'tuple' object does not support item assignment
```

1.2.18 Window 18

[43]:
```
# <!-- Student -->
#
import numpy as np
xArr = np.linspace(0, 9, 10)
print(xArr)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

1.2.19 Window 19

[44]:
```
# <!-- Student -->
#
yArr = xArr**2
print(yArr)
```

```
[ 0.  1.  4.  9. 16. 25. 36. 49. 64. 81.]
```

1.2.20 Window 20

[45]:
```
# <!-- Student -->
#
boolArr = xArr%2 == 0
print(boolArr)
```

```
[ True False  True False  True False  True False  True False]
```

1.2.21 Window 21

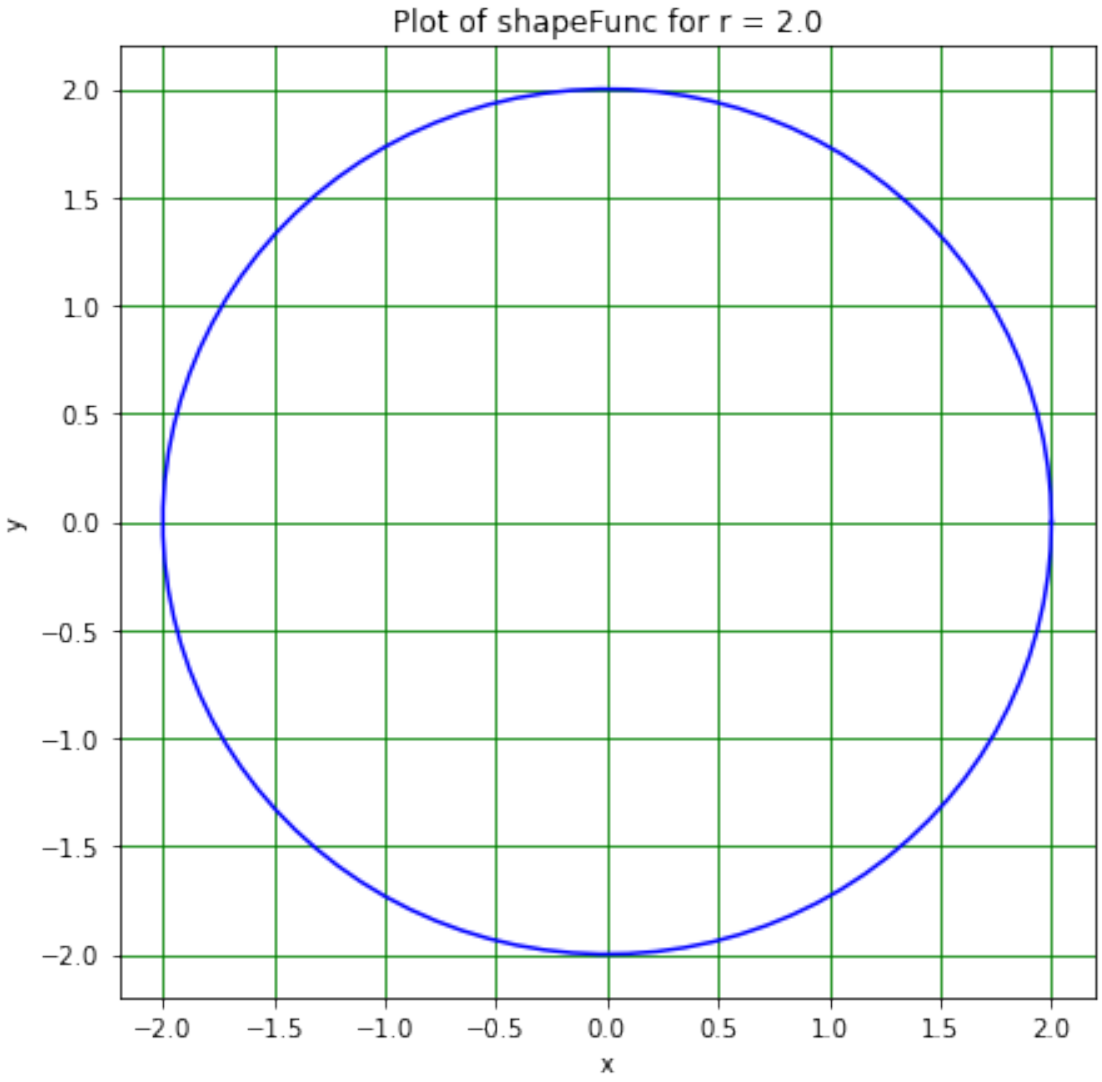
[46]:
```
# <!-- Student -->
#
stepArr = xArr[1:9:2]
```

```
print(stepArr)
```

```
[1. 3. 5. 7.]
```

1.2.22 Window 22

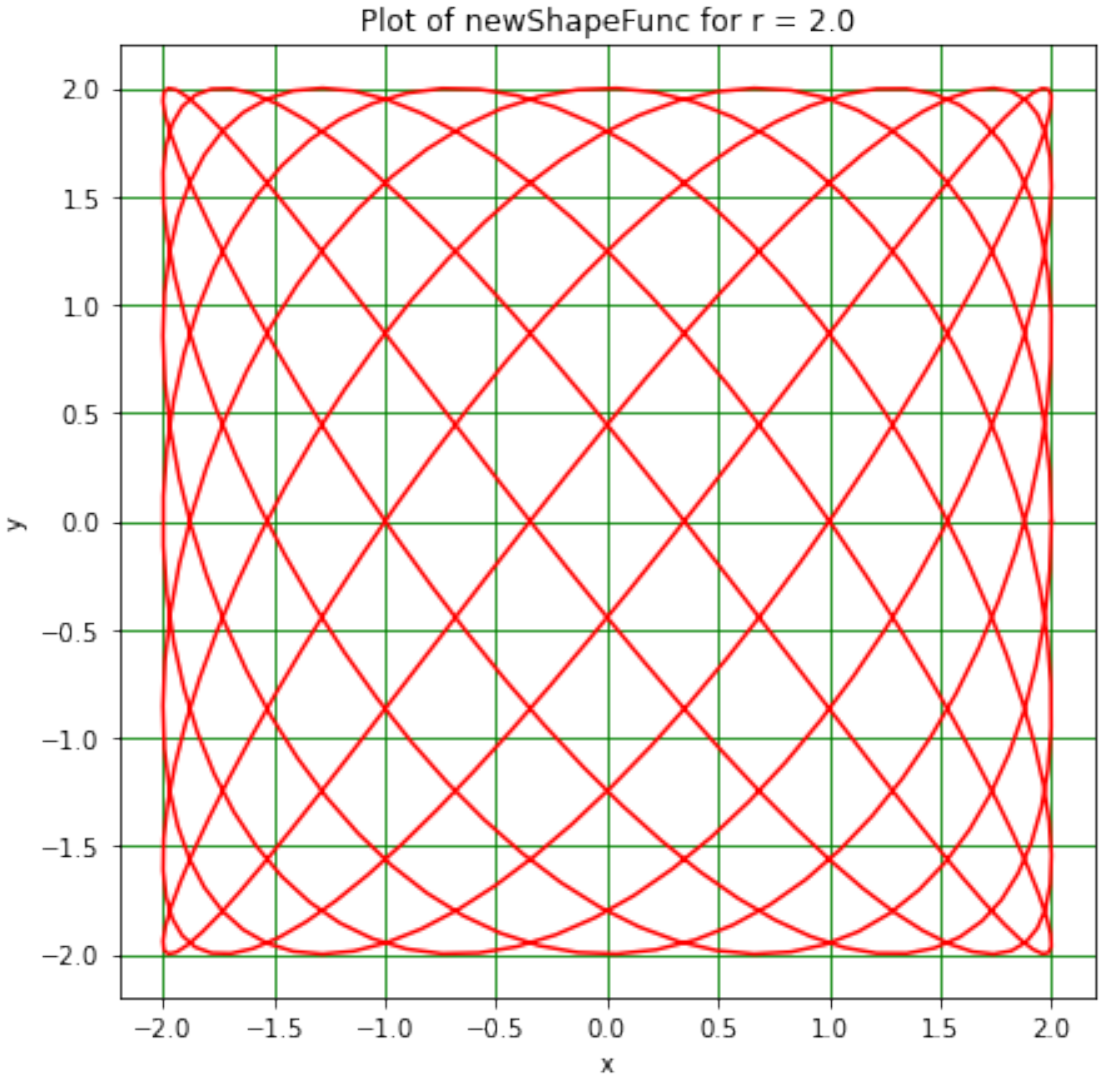
[47]:
```
# <!-- Student -->
#
import matplotlib.pyplot as plt
%matplotlib inline
#
def shapeFunc(r):
    q = np.linspace(0, 2*np.pi, 100)
    x = r*np.cos(q)
    y = r*np.sin(q)
    return x, y
#
r = 2.0
shapeX, shapeY = shapeFunc(r)
#
plt.figure(figsize = (7, 7))
plt.title("Plot of shapeFunc for r = " + str(r))
plt.xlabel('x')
plt.ylabel('y')
plt.plot(shapeX, shapeY, linestyle = '-', color = 'b')
plt.grid(color = 'g')
plt.show()
```



1.2.23 Window 23

[48]:
```
# <!-- Student -->
#
import matplotlib.pyplot as plt
%matplotlib inline
#
def newShapeFunc(r):
    q = np.linspace(0, 2*np.pi, 500)
    x = r*np.cos(7*q)
    y = r*np.sin(9*q)
    return x, y
#
```

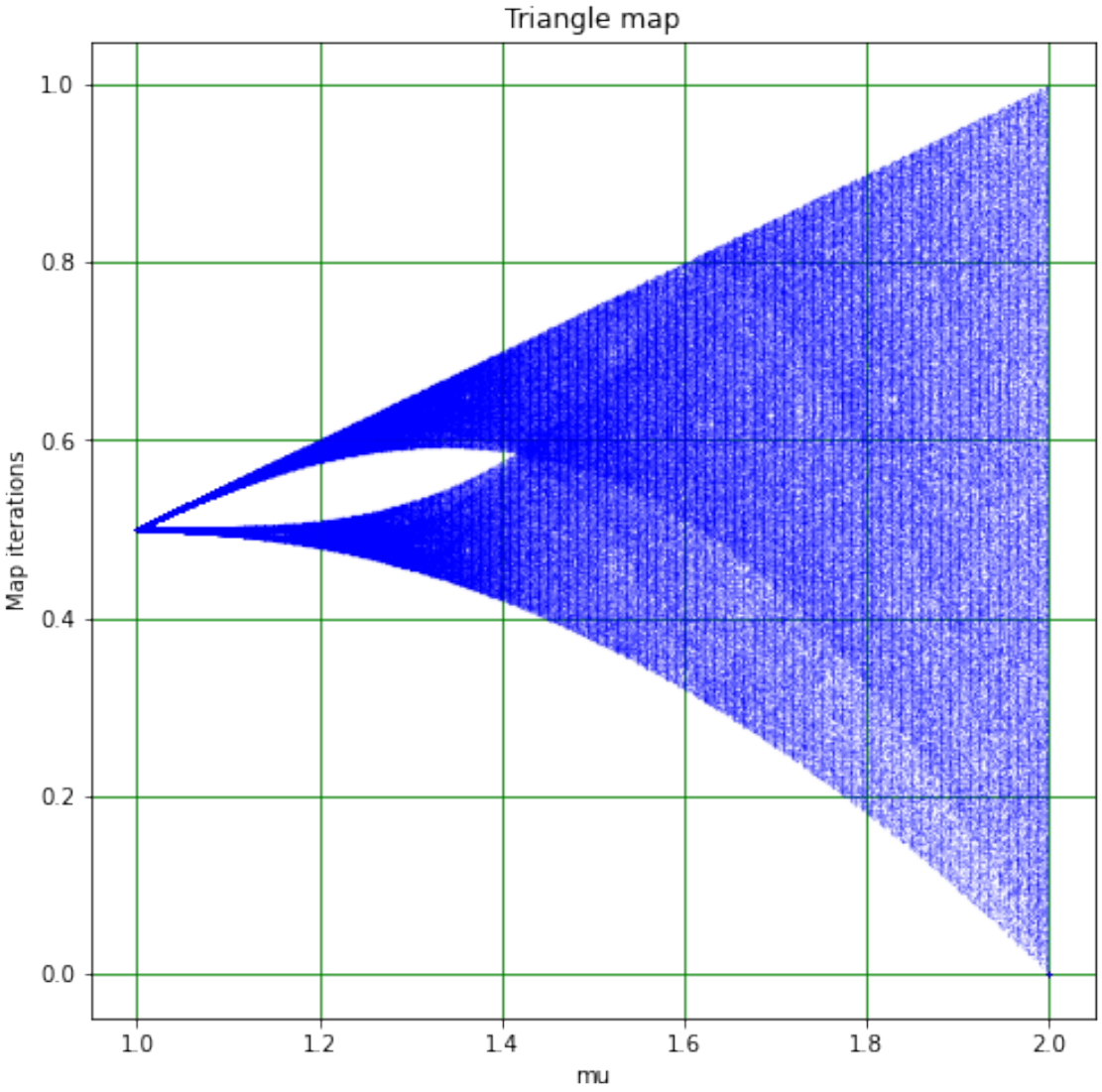
```
r = 2.0
shapeX, shapeY = newShapeFunc(r)
#
plt.figure(figsize = (7, 7))
plt.title("Plot of newShapeFunc for r = " + str(r))
plt.xlabel('x')
plt.ylabel('y')
plt.plot(shapeX, shapeY, linestyle = '-', color = 'r')
plt.grid(color = 'g')
plt.show()
```



1.3 An example program - the triangle map

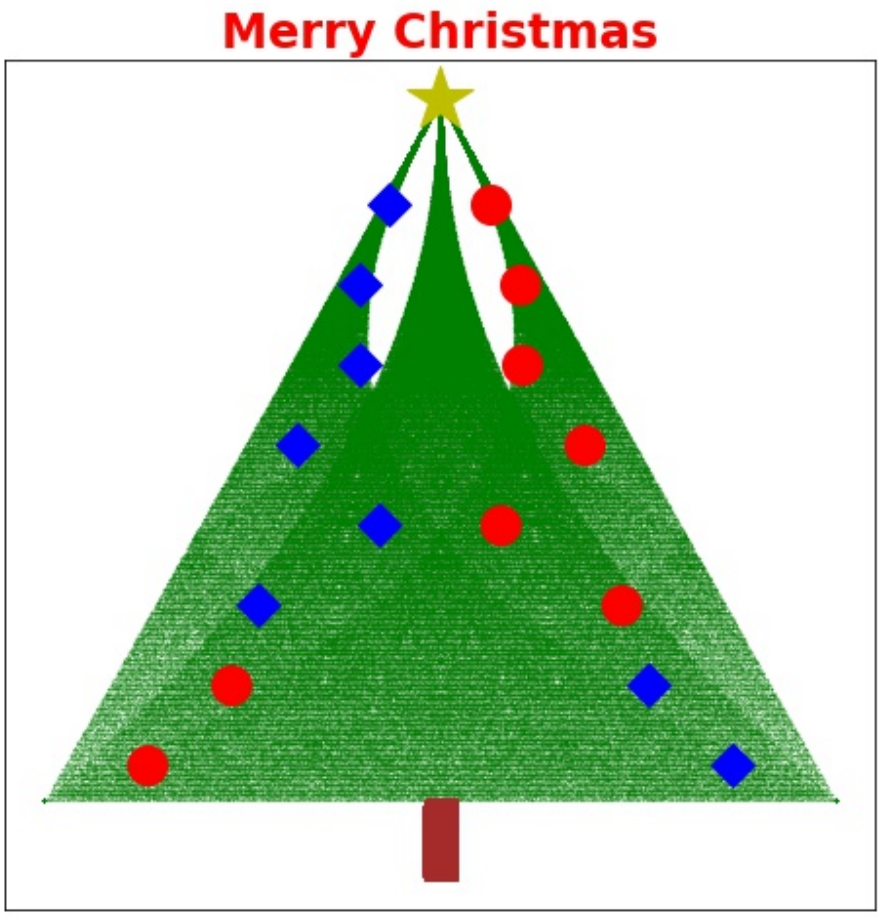
You will need this to make the Christmas card!

[50]:
```
# <!-- Student -->
#
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#
def triFunc(mu, x):
    '''
    Given value of x (between 0 and 1) and control parameter mu (between 0 and␣
 ↪2), returns next value of triangle map.
    '''
    if x < 0.5:
        out = mu*x
    else:
        out = mu*(1 - x)
    return out
#
nMu = 500
muMin = 1.0
muMax = 2.0
muArr = np.linspace(muMin, muMax, nMu)
xStart = 0.500001
nTrans = 500
nX = 1000
triMap = np.zeros((nMu, nX))
for n in range(0, nMu):
    trans = xStart
    for i in range(1, nTrans):
        trans = triFunc(muArr[n], trans)
    triMap[n, 0] = trans
    for i in range(1, nX):
        triMap[n, i] = triFunc(muArr[n], triMap[n, i - 1])
#
plt.figure(figsize = (8, 8))
plt.title("Triangle map")
plt.xlabel("mu")
plt.ylabel("Map iterations")
plt.plot(muArr[0:nMu], triMap[0:nMu, :], linestyle = '', marker = '.',␣
 ↪markersize = 0.1, color = 'b')
plt.grid(color = 'g')
plt.show()
```



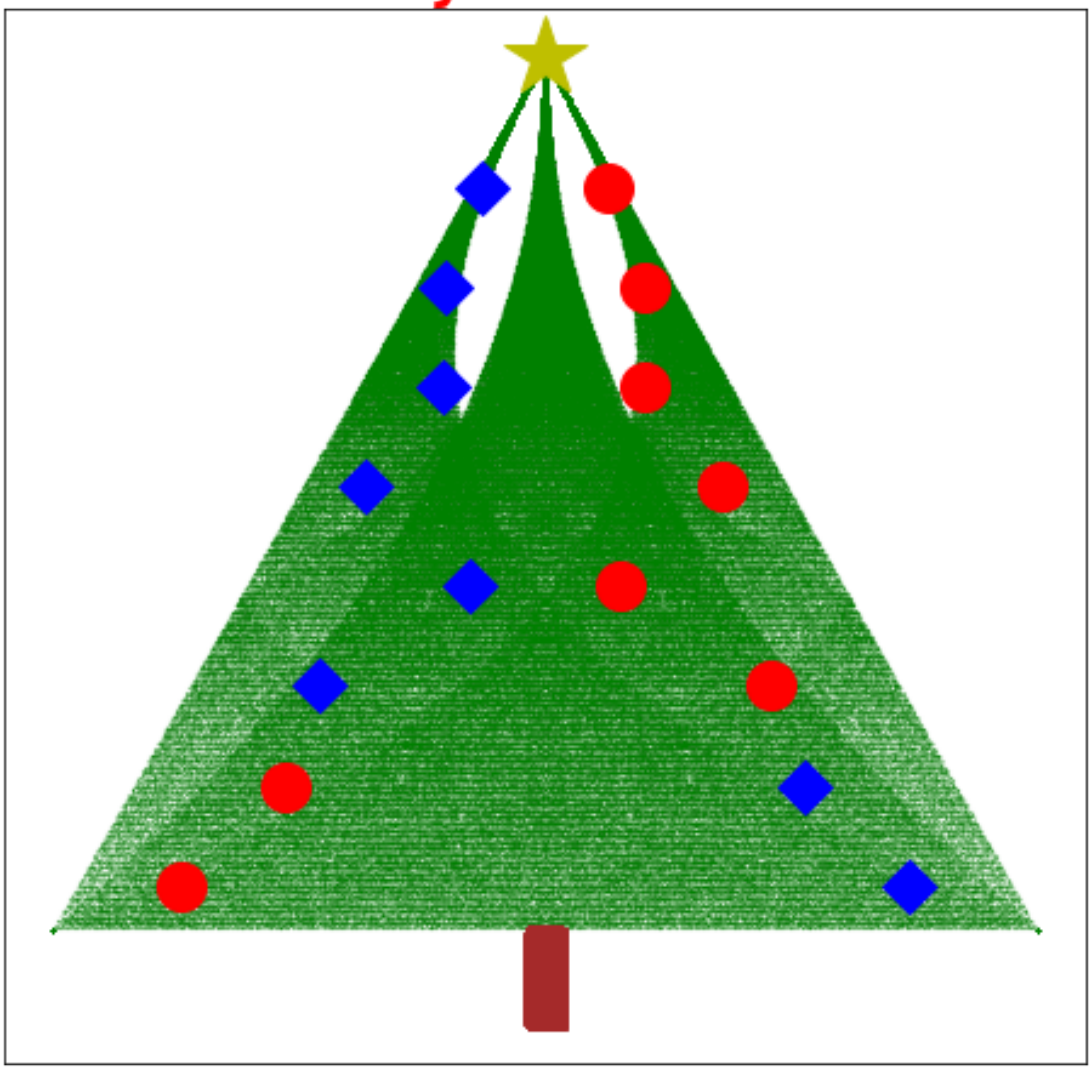
1.3.1 Window 25

Make a Python Christmas card using the triangle map and a few other bits and pieces!



[25]:
```
# <!-- Demo -->
#
nDecMu = 8 # Number of decorations (two sets made, so get twice this number)
muDecMin = 1.15 # Define range of mu values for decorations
muDecMax = 1.95 # ---"---
decMuArr = np.linspace(muDecMin, muDecMax, nDecMu)
nDecX = 1 # Number of decorations at each x value (becomes y when flip picture!)
decMap1 = np.full((nDecMu, nDecX), xStart)
decMap2 = np.full((nDecMu, nDecX), xStart)
for n in range(0, nDecMu):
    trans = xStart
    for i in range(1, nTrans):
        trans = triFunc(decMuArr[n], trans) # Let transients die away
    decMap1[n, 0] = trans # Set initial value for first decoration map
    for i in range(1, nDecX):
```

```
        decMap1[n, i] = triFunc(decMuArr[n], decMap1[n, i - 1]) # Fill first␣
↪decoration map
    decMap2[n, 0] = decMap1[n, nDecX - 1] # Set initial value for second␣
↪decoration map
    for i in range(1, nDecX):
        decMap2[n, i] = triFunc(decMuArr[n], decMap2[n, i - 1]) # Fill first␣
↪decoration map
#
plt.figure(figsize = (8, 8))
plt.title("Merry Christmas", color = 'r', fontsize = 24, fontweight = 'bold')
#
# Plot first version of triangle map, flipping x and y.
plt.plot(triMap[0:nMu, :], 2 - muArr[0:nMu], linestyle = '', marker = '.',␣
↪markersize = 0.1, color = 'g')
#
# Plot second version of triangle map, flipping x and y and flipping x so get␣
↪reflection in line x = 0.5
plt.plot(1 - triMap[0:nMu, :], 2 - muArr[0:nMu], linestyle = '', marker = '.',␣
↪markersize = 0.1, color = 'g')
#
# PLot first set of decorations, flipping x and y
plt.plot(decMap1[0:nMu, :], 2 - decMuArr[0:nMu], linestyle = '', marker = 'o',␣
↪markersize = 20.0, color = 'r')
#
# Plot second set of decorations, flipping x and y and flipping x so get␣
↪reflection in line x = 0.5
plt.plot(1 - decMap2[0:nMu, :], 2 - decMuArr[0:nMu], linestyle = '', marker =␣
↪'D', markersize = 15.0, color = 'b')
#
# Add the star at the top
plt.plot(0.5, 1.0, linestyle = '', marker = '*', markersize = 35.0, color = 'y')
#
# Define the trunk, values determined in plot below!
wTrunk = 0.01
nTrunk = 500
#
# Remove the scales
plt.xticks([])
plt.yticks([])
plt.plot(np.random.uniform(0.5 - wTrunk, 0.5 + wTrunk, nTrunk),
         np.random.uniform(-0.1, -0.01, nTrunk), marker = 's', markersize = 10.
↪0, color = 'brown')
plt.savefig("TriTreeXmas.jpg")
plt.show()
```


Merry Christmas

[ ]: