Phys105-Week11

December 8, 2019

1 Introduction to Computational Physics - Week 11

1.1 Table of contents week 11

1.2 Introduction to week 11

In the lecture this week, we will quickly remind ourselves of some of the main points we have covered in Phys105. We will do this by looking at some quiz questions that cover a range of Python's features and then discussing a short example program.

You should use the computer class to catch up on as many of the exercises as you can and check that they have been seen by a demonstrator and marked.

There is only one new question this week which gives you a chance to practice some of the things we have learned, but it will not be assessed

1.3 Python quiz

Predict the outcome of the following snippets of code!

1.3.1 Question 1

[1]:
```python
# <!-- Student -->
#
a = 14.0
print(a)
```

```
14.0
```

1.3.2 Question 2

[2]:
```python
# <!-- Student -->
#
print(type(a))
```

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

1.3.3 Question 3

[3]:
```python
# <!-- Student -->
#
b = 2
print(b)
```

```
2
```

1.3.4 Question 4

[4]:
```python
# <!-- Student -->
#
print(type(b))
```

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

1.3.5 Question 5

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

```
2
```

1.3.6 Question 6

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

```
1
```

1.3.7 Question 7

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

```
4
3
2
```

1.3.8 Question 8

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

```
4
3
2
1
```

1.3.9 Question 9

[9]:
```
# <!-- Student -->
#
x = 3
print(x > 3)
```

```
False
```

1.3.10 Question 10

[10]:
```
# <!-- Student -->
#
print(x <= 3)
```

```
True
```

1.3.11 Question 11

[11]:
```
# <!-- Student -->
#
a = False
b = True
print(a and b)
```

```
False
```

1.3.12 Question 12

[12]:
```
# <!-- Student -->
#
print(a or b)
```

```
True
```

1.3.13 Question 13

[13]:
```
# <!-- Student -->
#
myList = ["one", "two", "three", "four", "five"]
print(type(myList[2]))
```

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

1.3.14 Question 14

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

```
four
```

1.3.15 Question 15

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

```
one
two
three
four
five
```

1.3.16 Question 16

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

```
6
```

1.3.17 Question 17

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

```
      ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-17-7946237c3e99> in <module>
          1 # <!-- Student -->
          2 #
    ----> 3 myTuple[3] = "seven"
```

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

1.3.18 Question 18

[18]:
```
# <!-- Student -->
#
import numpy as np
xArr = np.linspace(0, 9, 10)
print(xArr)
```

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

1.3.19 Question 19

[19]:
```
# <!-- Student -->
#
yArr = xArr**2
print(yArr)
```

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

1.3.20 Question 20

[20]:
```
# <!-- Student -->
#
boolArr = xArr%2 == 0
print(boolArr)
```

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

1.3.21 Question 21

[21]:
```
# <!-- Student -->
#
selectArr = xArr[boolArr]
print(selectArr)
```

```
[0. 2. 4. 6. 8.]
```

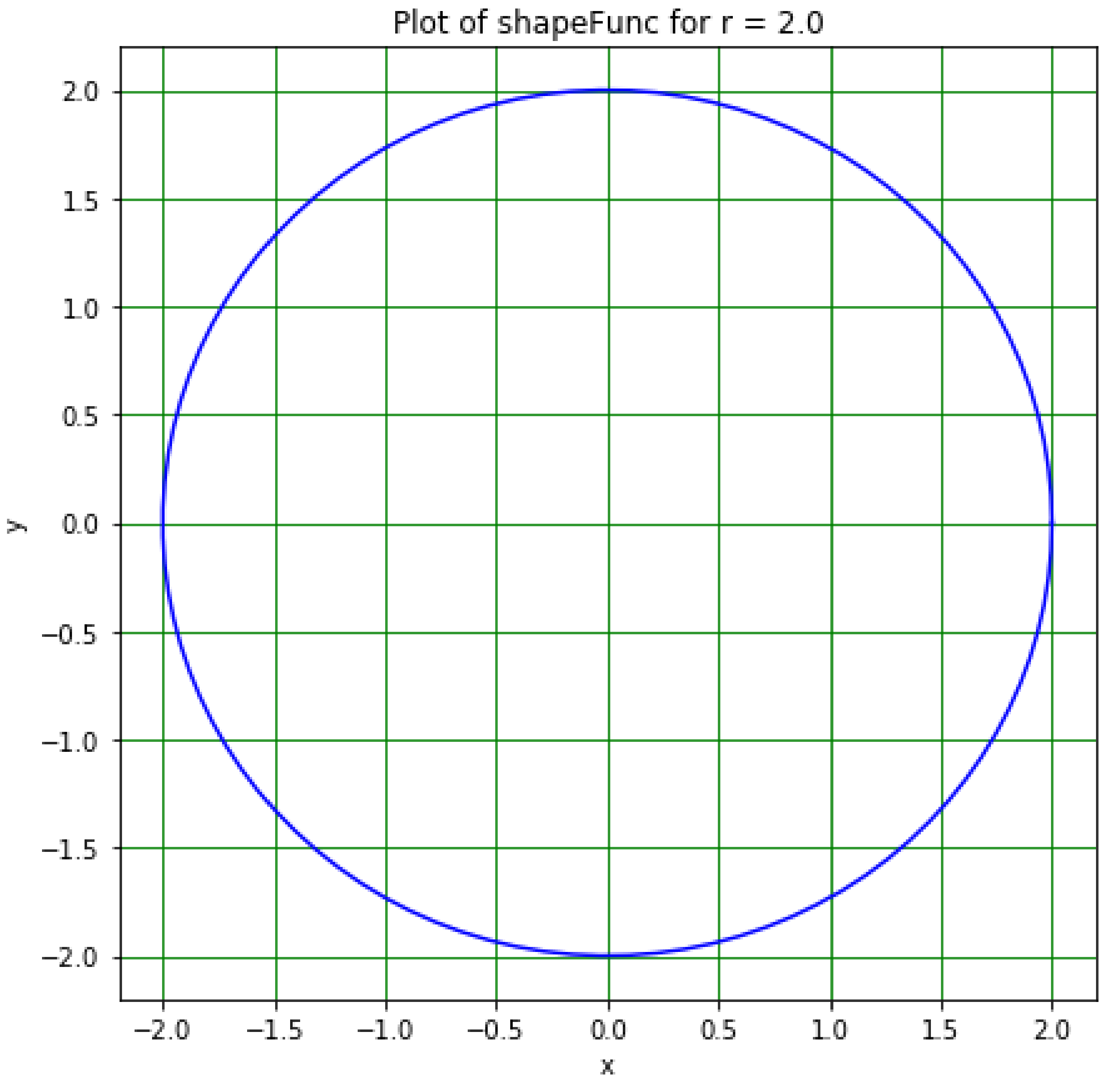
1.3.22 Question 22

[22]:
```
# <!-- Student -->
#
stepArr = xArr[1:9:2]
print(stepArr)
```

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

1.3.23 Question 23

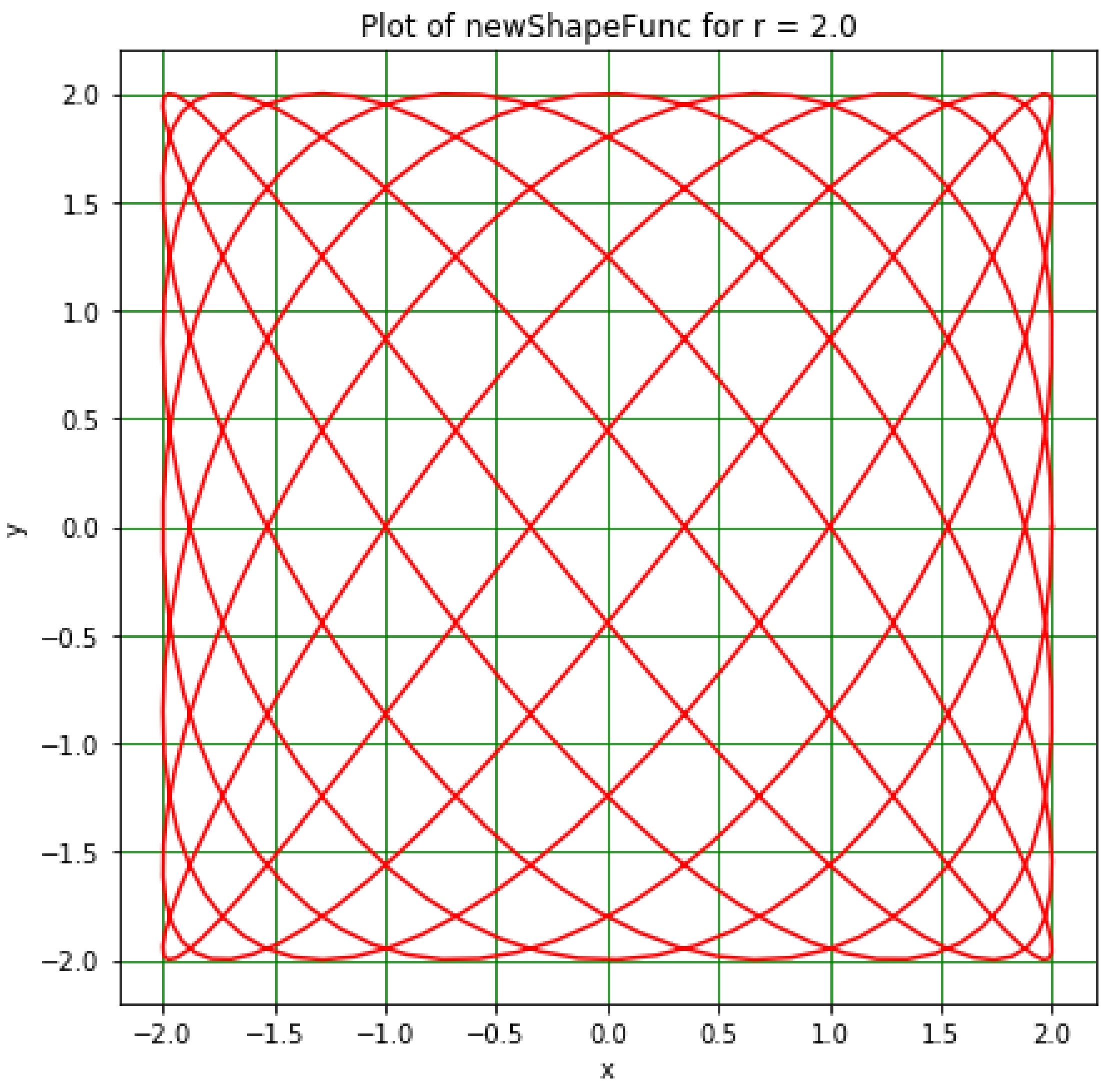
[23]:
```
# <!-- Student -->
#
import matplotlib.pyplot as plt
%matplotlib inline
#
def shapeFunc(r):
    q = np.linspace(0, 2*np.pi, 100)
    x = r*np.cos(q)
    y = r*np.sin(q)
    return x, y
#
r = 2.0
shapeX, shapeY = shapeFunc(r)
#
plt.figure(figsize = (7, 7))
plt.title("Plot of shapeFunc for r = " + str(r))
plt.xlabel('x')
plt.ylabel('y')
plt.plot(shapeX, shapeY, linestyle = '-', color = 'b')
plt.grid(color = 'g')
plt.show()
```



1.3.24 Question 24

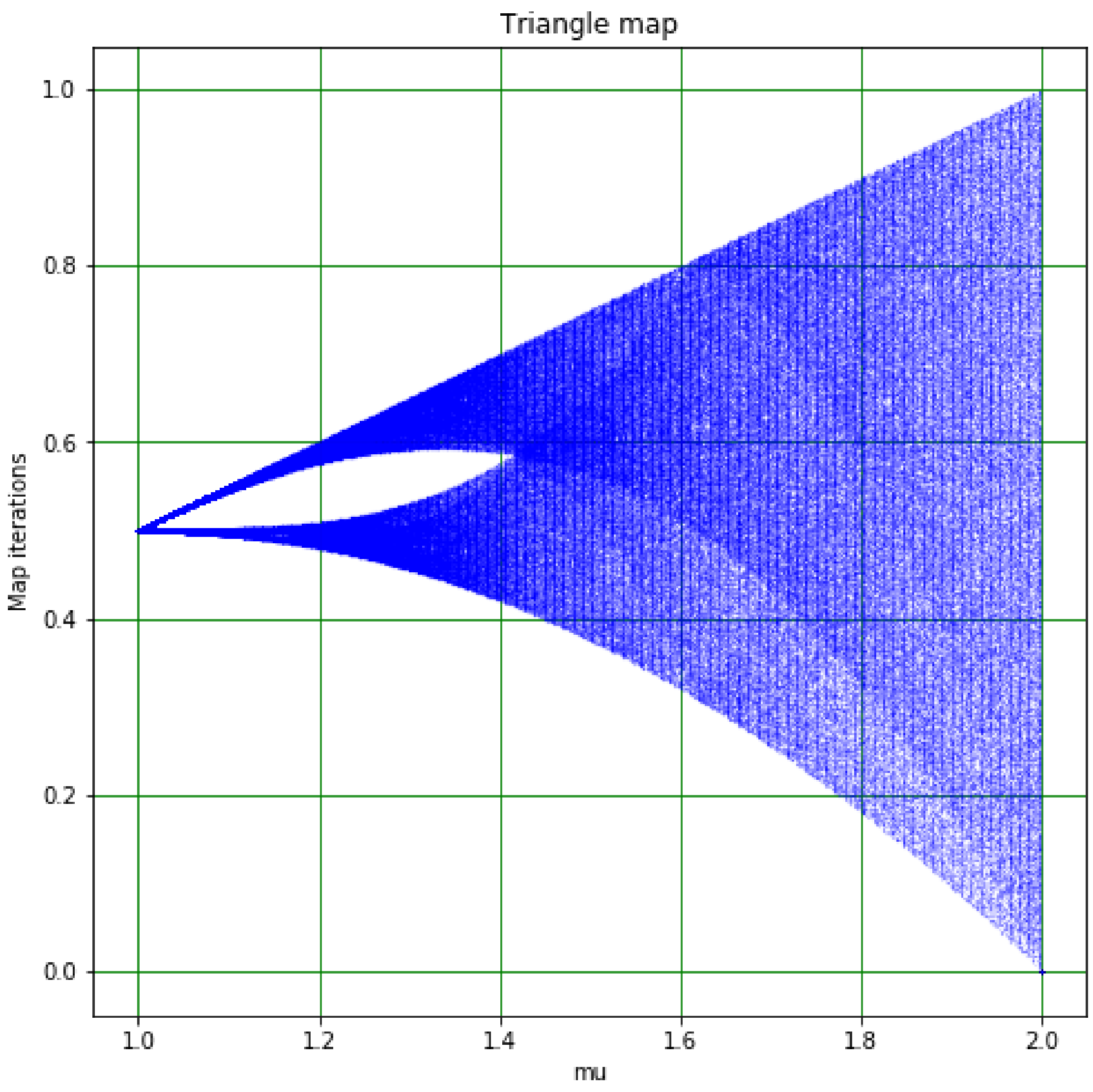
[24]:
```
# <!-- Student -->
#
import matplotlib.pyplot as plt
%matplotlib inline
#
def newShapeFunc(r):
    q = np.linspace(0, 2*np.pi, 500)
    x = r*np.cos(7*q)
    y = r*np.sin(9*q)
    return x, y
#
```

```
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()
```



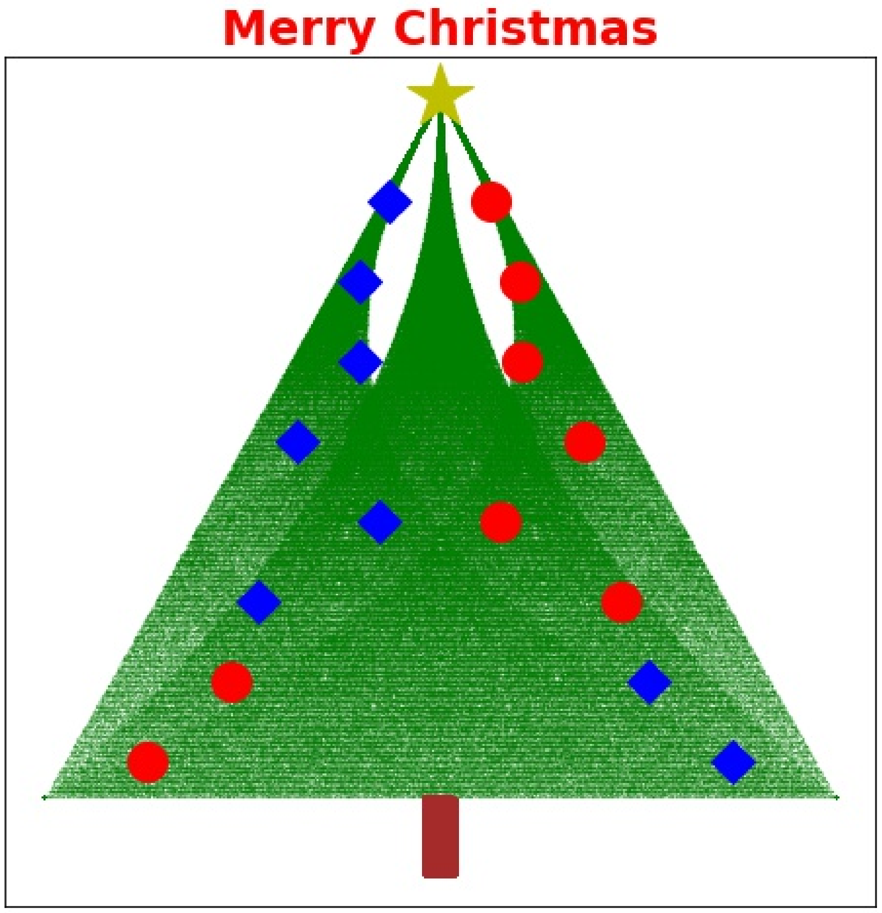
1.4 An example program - the triangle map

[25]:
```
# <!-- 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()
```

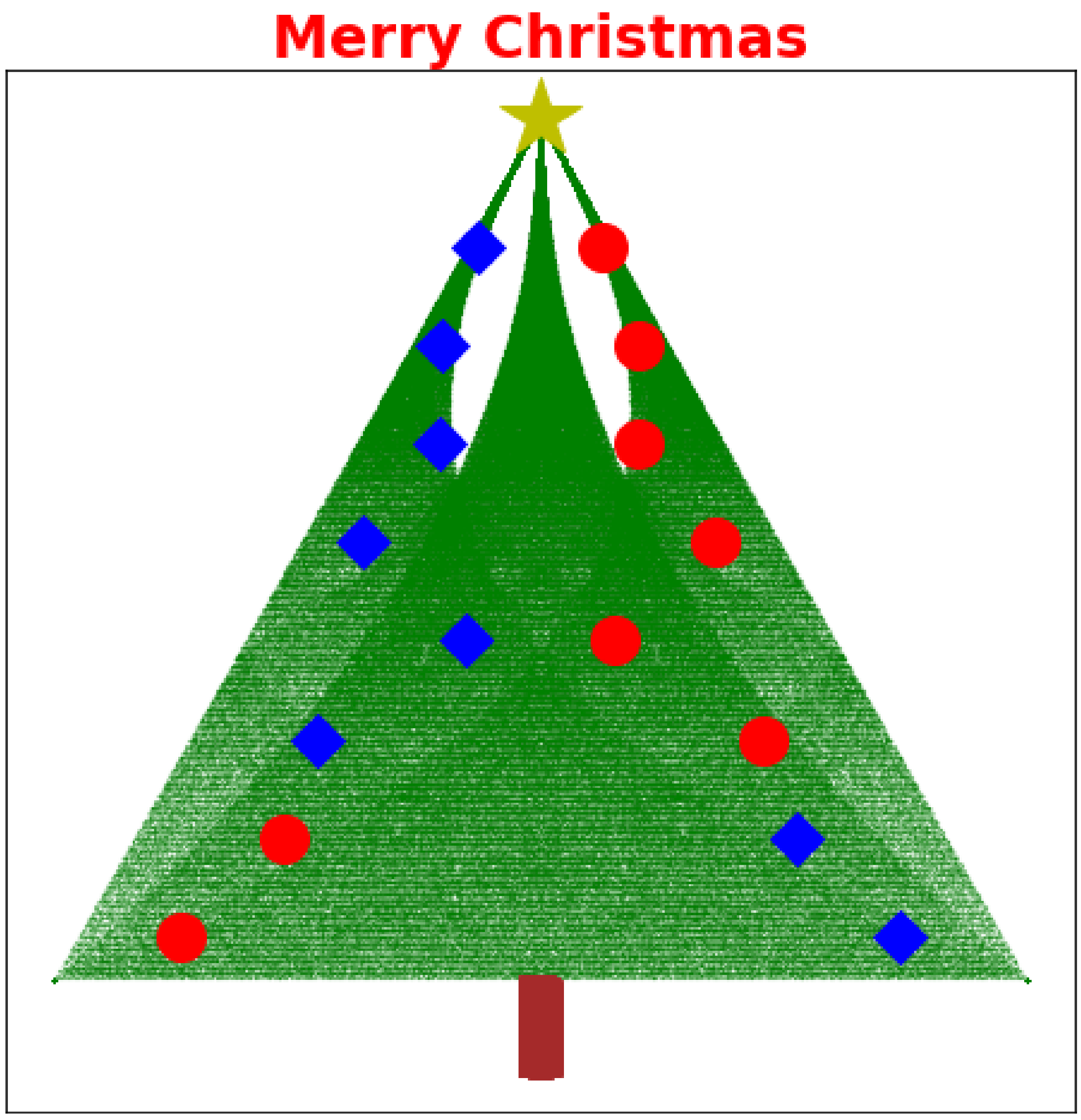


1.4.1 Week 11 exercise 1

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



[28]:
```python
# <!-- 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')
#
# Make the trunk!
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()
```



1.5 Week 11 marks

Exercise	Mark	Comments
1	0	Just for fun!
Total	**0**	