Python - Sticker Collection

Continued from Procedural

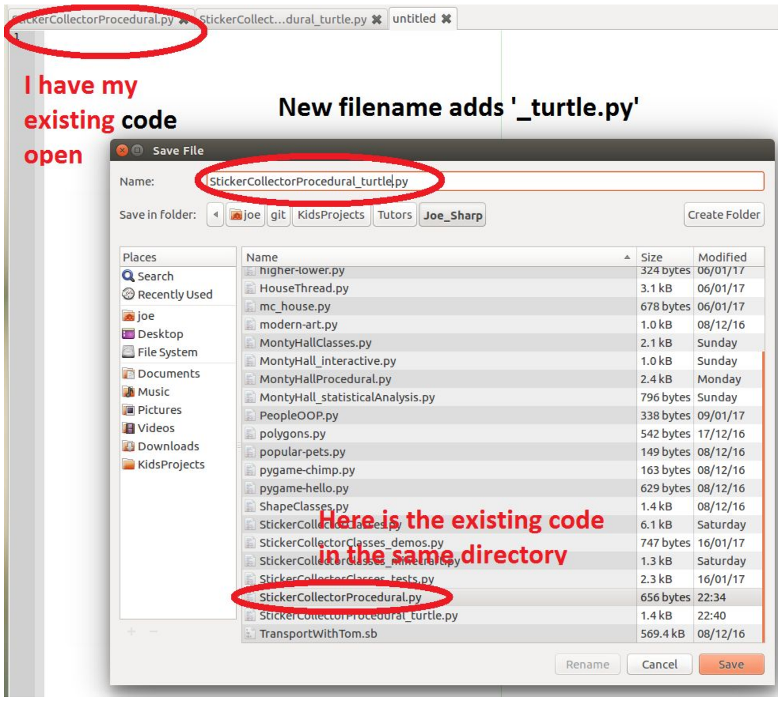
This exercise assumes that you have completed the Python Sticker Collection - Procedural Version. If you have not yet done that, then go back and complete it first. This tutorial literally picks up from where you left off.

Create New Source File

I am assuming you have your existing Sticker Collection code saved to a python file. We are going to write another python file and **import** the existing one. Our existing python file will now become a *library*.

Create a new file in geany, and save it into the same directory as your existing sticker collection python code. Give it a different name. For example:
Myy original file is called **StickerCollectorProcedural.py**
My new file will be called **StickerCollectorProcedural_turtle.py**

When I save my new file, it looks like this:


ickerCollectorProcedural.py
StickerCollect...dural_turtle.py
untitled
I have my existing code open
New filename adds '_turtle.py'
Save File
Here is the existing code in the same directory
Rename
Cancel
Save

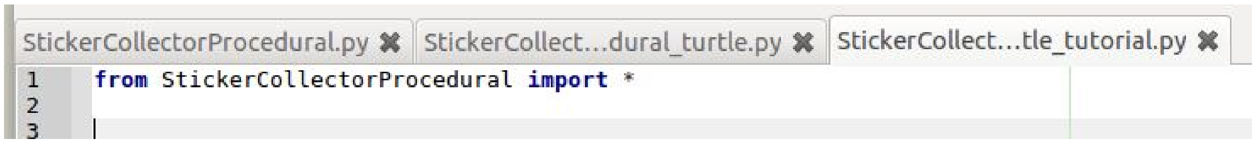
Name: StickerCollectorProcedural_turtle.py

Save in folder: joe / git / KidsProjects / Tutors / Joe_Sharp

Name	Size	Modified
higher-lower.py	324 bytes	06/01/17
HouseThread.py	3.1 kB	06/01/17
mc_house.py	678 bytes	06/01/17
modern-art.py	1.0 kB	08/12/16
MontyHallClasses.py	2.1 kB	Sunday
MontyHall_interactive.py	1.0 kB	Sunday
MontyHallProcedural.py	2.4 kB	Monday
MontyHall_statisticalAnalysis.py	796 bytes	Sunday
PeopleOOP.py	338 bytes	09/01/17
polygons.py	542 bytes	17/12/16
popular-pets.py	149 bytes	08/12/16
pygame-chimp.py	163 bytes	08/12/16
pygame-hello.py	629 bytes	08/12/16
ShapeClasses.py	1.4 kB	08/12/16
StickerCollectorClasses.py	6.1 kB	Saturday
StickerCollectorClasses_demos.py	747 bytes	16/01/17
StickerCollectorClasses_minecraft.py	1.3 kB	Saturday
StickerCollectorClasses_tests.py	2.3 kB	16/01/17
StickerCollectorProcedural.py	656 bytes	22:34
StickerCollectorProcedural_turtle.py	1.4 kB	22:40
TransportWithTom.sb	569.4 kB	08/12/16

Import the Existing Source File

You are now editing the new source file, add an import statement that looks like this (your filename may vary)

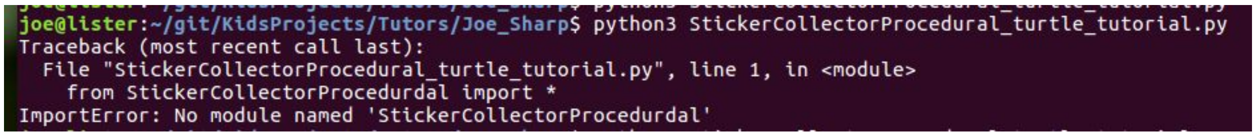

StickerCollectorProcedural.py
StickerCollect...dural_turtle.py
StickerCollect...tle_tutorial.py

```python
from StickerCollectorProcedural import *

```

Test the Import

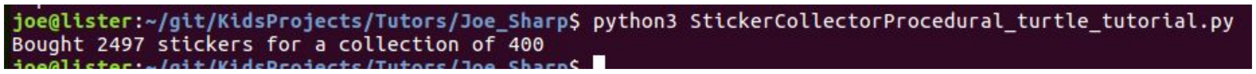
Before we go any further, let's test the import process. Run the code by using F5 or by invoking it from the terminal.

If the import is not working you will see an error like this:



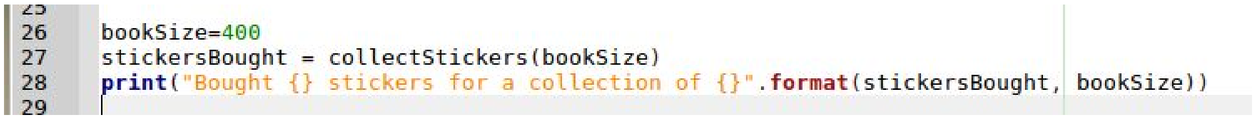
```
joe@lister:~/git/KidsProjects/Tutors/Joe_Sharp$ python3 StickerCollectorProcedural_turtle_tutorial.py
Traceback (most recent call last):
  File "StickerCollectorProcedural_turtle_tutorial.py", line 1, in <module>
    from StickerCollectorProcedurdal import *
ImportError: No module named 'StickerCollectorProcedurdal'
```

After I corrected the spelling error it looked like this:



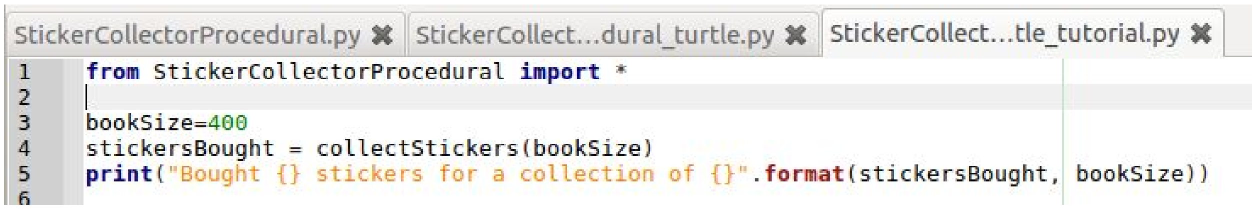
```
joe@lister:~/git/KidsProjects/Tutors/Joe_Sharp$ python3 StickerCollectorProcedural_turtle_tutorial.py
Bought 2497 stickers for a collection of 400
joe@lister:~/git/KidsProjects/Tutors/Joe_Sharp$
```

This shows that the **new** source file has successfully imported the **old** source file. It has run the sticker simulation using the code in the **old** source file. We will now move the lines shown:



```python
bookSize=400
stickersBought = collectStickers(bookSize)
print("Bought {} stickers for a collection of {}".format(stickersBought, bookSize))
```

Cut and Paste them into your **new** source file (the one with the import).


StickerCollectorProcedural.py
StickerCollect...dural_turtle.py
StickerCollect...tle_tutorial.py

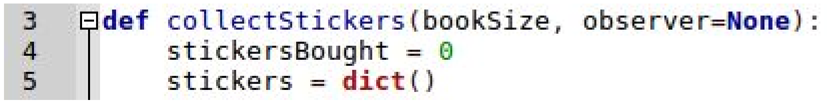
```python
from StickerCollectorProcedural import *

bookSize=400
stickersBought = collectStickers(bookSize)
print("Bought {} stickers for a collection of {}".format(stickersBought, bookSize))
```

TEST YOUR CODE: If you run the **new** source file, the test should run. Note that if you run the **old** code nothing will happen. It defines a function but the **old** code will no longer call it.

Allow Observer Function

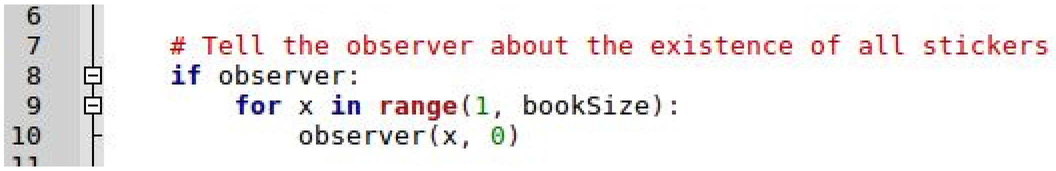
Our turtle drawing will fill in the stickers as we collect them, for this to work we must intercept the code where the stickers are collected. How do we do this? We pass a function into the **collectStickers** function. I will call this function the **observer**.

Edit the **collectStickers** function as shown, we have just added a parameter called **observer** and given it a default value of **None**.



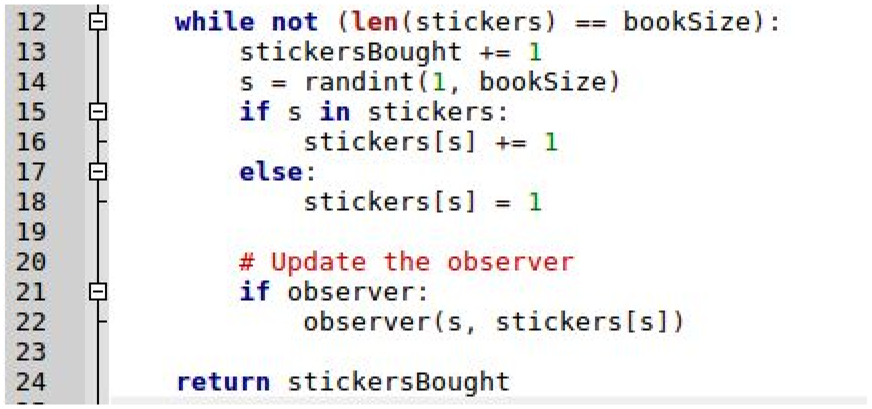
```
 3  def collectStickers(bookSize, observer=None):
 4      stickersBought = 0
 5      stickers = dict()
```

We should tell the observer about all of our stickers at the start, add the code shown from lines 8 to 10 before the main collection loop.



```
 6
 7      # Tell the observer about the existence of all stickers
 8      if observer:
 9          for x in range(1, bookSize):
10              observer(x, 0)
```

Now we need to intercept the stickers each time they are collected. This will happen inside the main collection loop. Lines 20 to 22 show the call to the observer.

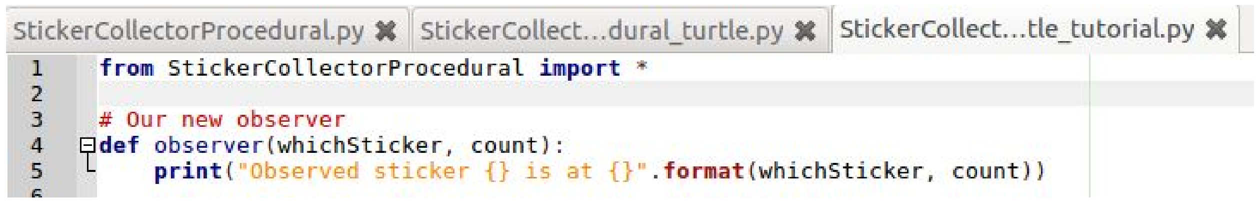


```
12      while not (len(stickers) == bookSize):
13          stickersBought += 1
14          s = randint(1, bookSize)
15          if s in stickers:
16              stickers[s] += 1
17          else:
18              stickers[s] = 1
19
20          # Update the observer
21          if observer:
22              observer(s, stickers[s])
23
24      return stickersBought
```

TEST YOUR CODE: Run the **new** source file and check that you haven't broken anything. The test should still run. We have not provided an observer yet, we will do this in the next section.

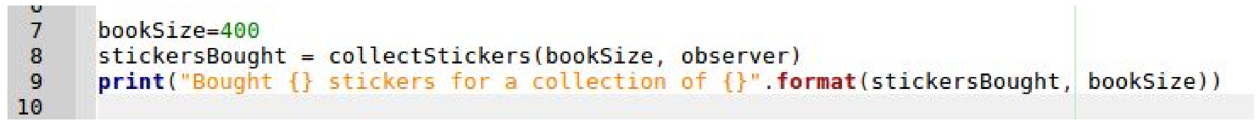
Write Observer Function

The **collectStickers** function now allows us to provide an observer. We will now write one and pass it in. Edit your **new** source file by adding the function shown from lines 4 to 5 (3 is just a comment).


StickerCollectorProcedural.py
StickerCollect...dural_turtle.py
StickerCollect...tle_tutorial.py

```
1  from StickerCollectorProcedural import *
2
3  # Our new observer
4  def observer(whichSticker, count):
5      print("Observed sticker {} is at {}".format(whichSticker, count))
```

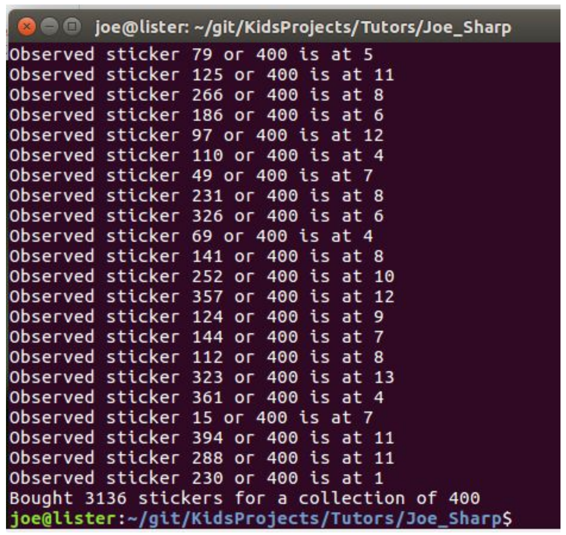
We will now pass this function into the **collectSticker** function as shown on line 8.



```
7  bookSize=400
8  stickersBought = collectStickers(bookSize, observer)
9  print("Bought {} stickers for a collection of {}".format(stickersBought, bookSize))
10
```

When you run this code, it should collect the stickers as before, but this time each time a sticker is collected the observer function will be called. This will result in a print statement as per line 5 above.

TEST YOUR CODE: Run the new source file and it should look something like this:



```
Observed sticker 79 or 400 is at 5
Observed sticker 125 or 400 is at 11
Observed sticker 266 or 400 is at 8
Observed sticker 186 or 400 is at 6
Observed sticker 97 or 400 is at 12
Observed sticker 110 or 400 is at 4
Observed sticker 49 or 400 is at 7
Observed sticker 231 or 400 is at 8
Observed sticker 326 or 400 is at 6
Observed sticker 69 or 400 is at 4
Observed sticker 141 or 400 is at 8
Observed sticker 252 or 400 is at 10
Observed sticker 357 or 400 is at 12
Observed sticker 124 or 400 is at 9
Observed sticker 144 or 400 is at 7
Observed sticker 112 or 400 is at 8
Observed sticker 323 or 400 is at 13
Observed sticker 361 or 400 is at 4
Observed sticker 15 or 400 is at 7
Observed sticker 394 or 400 is at 11
Observed sticker 288 or 400 is at 11
Observed sticker 230 or 400 is at 1
Bought 3136 stickers for a collection of 400
joe@lister:~/git/KidsProjects/Tutors/Joe_Sharp$
```

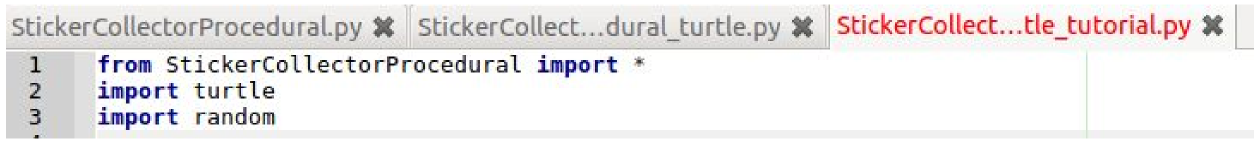
Can you see how our observer function is being called? Time to draw something!

Write a Create Turtle function

We will create a single turtle for each sticker that we need to collect.

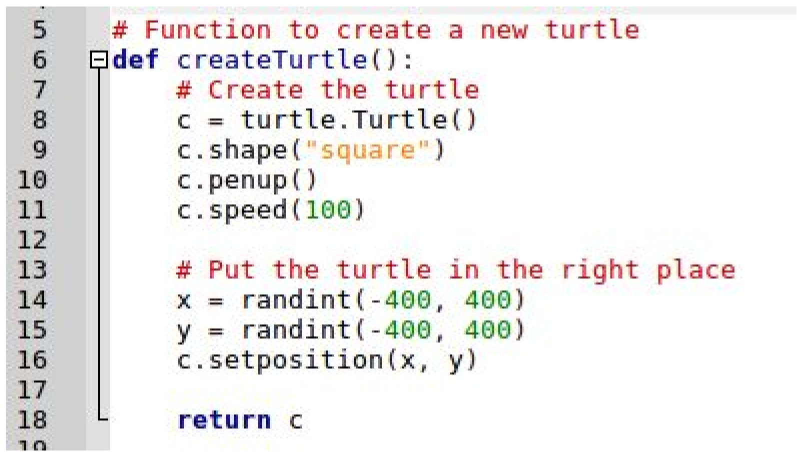
- When the sticker is first created it with a count of zero, we will make the turtle RED.
- When a sticker is collected (count > 0) we will make it GREEN
- As we get duplicates of a sticker, we will increase the SIZE of that turtle.

Import the turtle library and the random library (lines 2 and 3)


StickerCollectorProcedural.py
StickerCollect...dural_turtle.py
StickerCollect...tle_tutorial.py

```
1   from StickerCollectorProcedural import *
2   import turtle
3   import random
```

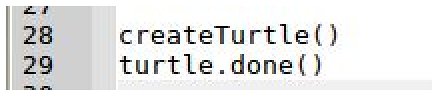
The code below shows our new turtle function from line 6 to 18



```
5    # Function to create a new turtle
6    def createTurtle():
7        # Create the turtle
8        c = turtle.Turtle()
9        c.shape("square")
10       c.penup()
11       c.speed(100)
12
13       # Put the turtle in the right place
14       x = randint(-400, 400)
15       y = randint(-400, 400)
16       c.setposition(x, y)
17
18       return c
```

I am placing each turtle in a random location within the game area.
I am calling penup() to stop the turtle drawing lines as it moves from the centre to its location.
I am setting the speed so that the animation is quick.
The function returns the created turtle, we will be calling this from our observer.

Add the following lines to the end of the code just to test that the function works.



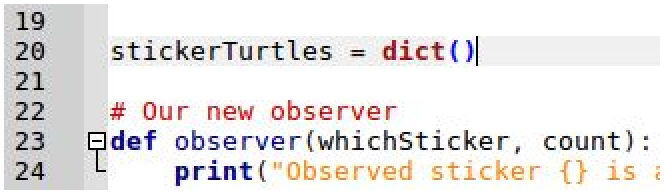
```
28   createTurtle()
29   turtle.done()
```

TEST YOUR CODE: Run the code and it should run the observed test as before, then create a random single turtle (square shape) and leave the window open.

Create a Turtle for Each Sticker

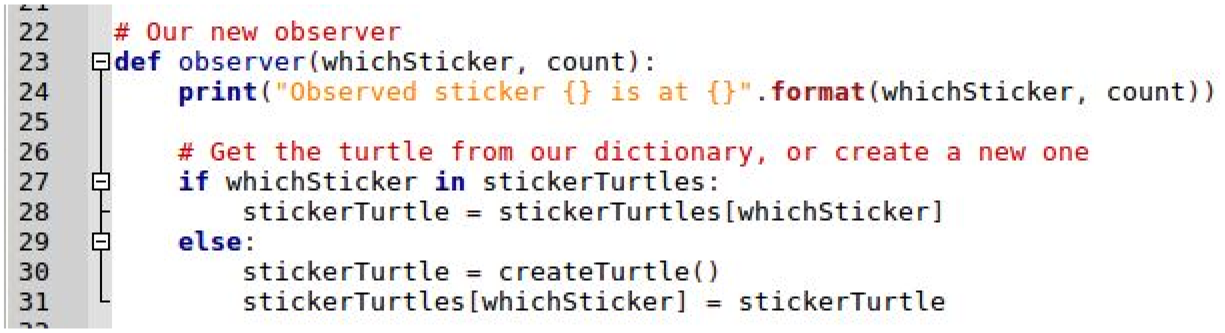
In this section we will create a **dictionary** of turtles. The keys will be the sticker numbers, the values will be the turtles created by our **createTurtle** function.

Add the following lines above the **observer** function (shown here as line 20)



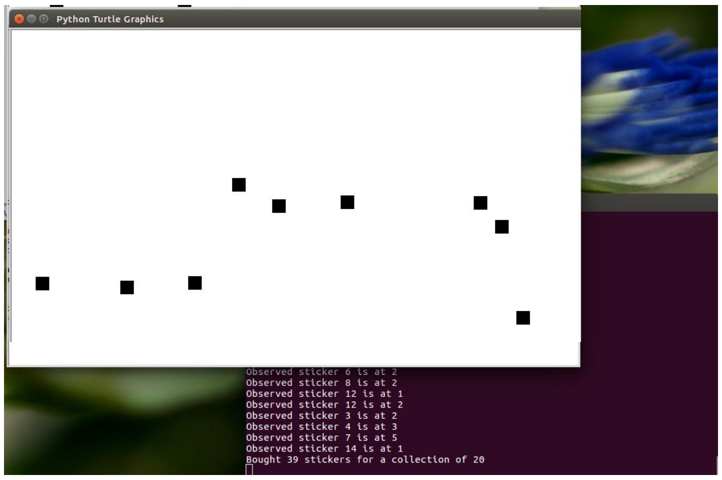
```
19
20   stickerTurtles = dict()
21
22   # Our new observer
23   def observer(whichSticker, count):
24       print("Observed sticker {} is a
```

Add the lines 27 to 31 to your observer function.



```
22   # Our new observer
23   def observer(whichSticker, count):
24       print("Observed sticker {} is at {}".format(whichSticker, count))
25
26       # Get the turtle from our dictionary, or create a new one
27       if whichSticker in stickerTurtles:
28           stickerTurtle = stickerTurtles[whichSticker]
29       else:
30           stickerTurtle = createTurtle()
31           stickerTurtles[whichSticker] = stickerTurtle
```

RUN YOUR CODE: This should now create a turtle for each sticker that is first received. At this stage we are just plonking them onto the screen and leaving them there. When the code has run it should look a bit like this (I have shortened the turtle window here):

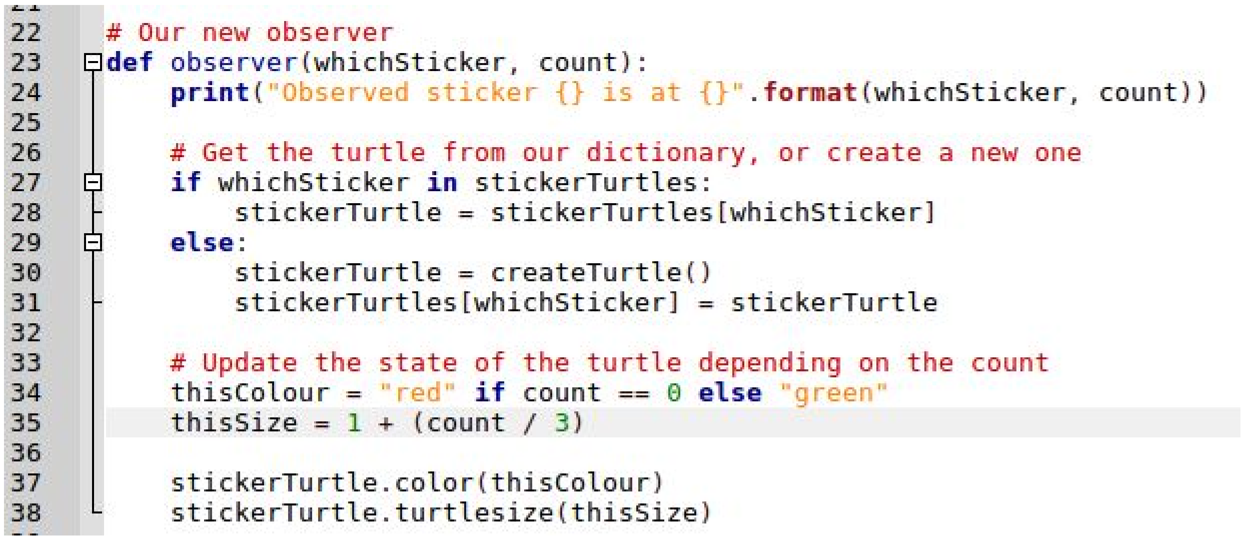


```
Observed sticker 6 is at 2
Observed sticker 8 is at 2
Observed sticker 12 is at 1
Observed sticker 12 is at 2
Observed sticker 3 is at 2
Observed sticker 4 is at 3
Observed sticker 7 is at 5
Observed sticker 14 is at 1
Bought 39 stickers for a collection of 20
```

Modify Turtles as Stickers are Collected

We will be modifying the turtles colour and size as the stickers are collected.

Lines 34 to 38 show how I am doing this.

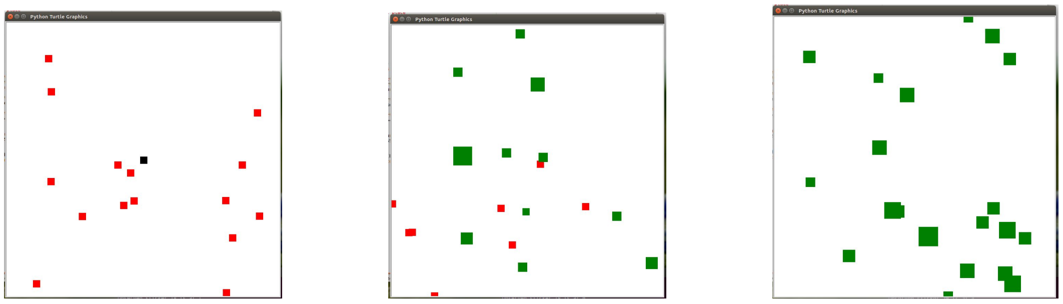


```
22      # Our new observer
23      def observer(whichSticker, count):
24          print("Observed sticker {} is at {}".format(whichSticker, count))
25
26          # Get the turtle from our dictionary, or create a new one
27          if whichSticker in stickerTurtles:
28              stickerTurtle = stickerTurtles[whichSticker]
29          else:
30              stickerTurtle = createTurtle()
31              stickerTurtles[whichSticker] = stickerTurtle
32
33          # Update the state of the turtle depending on the count
34          thisColour = "red" if count == 0 else "green"
35          thisSize = 1 + (count / 3)
36
37          stickerTurtle.color(thisColour)
38          stickerTurtle.turtlesize(thisSize)
```

We populated the **stickerTurtle** variable from the dictionary. Creating a new turtle if required.

Line 34 shows us calculating the colour by using the count. Sticker we have yet to collect are RED, collected stickers are GREEN.
Line 35 shows us calculating the size of the turtle. It must be at least 1, I then divide the count by 3 to prevent high counts drowning out the display.
Lines 37 and 38 show us setting the color and size of the turtle. Simples.

TEST YOUR CODE: The project is now finished, when you run your code it should end up looking like this:



CONGRATULATIONS, YOU HAVE FINISHED THE STICKER COLLECTOR TURTLE