Python - Sticker Collection

Have you ever collected stickers? I remember collecting Premier Leagure 94/95 stickers and swapping them with friends. We would invariably end up with a great big wad of ‘swaps’ and spend lunch and break time hunting for ones we did not have in our friends swap piles.

After the sticker collection got so far, I would invariably end up with 4 or 5 copies of the same people. Do the sticker companies do this deliberately? Are the stickers truly random? Let us experiment using code to find out.

Defining the function

We will define a function to simulate a single person collecting an entire sticker book 1 sticker at a time. This is obviously not as efficient as multiple people collecting and swapping, but it should illustrate that getting several copies of the same sticker is actually very likely.

This function will accept 1 argument called **bookSize** and return the number of stickers bought. The skeleton of the function will look like this.

```
1
2  def collectStickers(bookSize):
3      stickersBought = 0
4
5      return stickersBought
6
```

Create a Dictionary

In order to track how many copies of each sticker we have, I will create a **dictionary**.

The documentation for a dictionary can be found here
https://docs.python.org/3.7/tutorial/datastructures.html#dictionaries

Dictionaries (also called maps and associative arrays) are used to store key value pairs.

Our stickers dictionary will start out empty, then as each numbered sticker is collected, the sticker number will be used as a key, and the count will be used as the value. For example if I buy a packet of stickers with the numbers 1, 45, 60 and 20 then my dictionary will look like this.

Sticker Number (key)	Count (value)
1	1
45	1
60	1
20	1

If I buy a second packet, which contains the numbers 2, 45, 78 and 100 then my dictionary will look like this.

Sticker Number (key)	Count (value)
1	1
2	1
45	2
60	1
78	1
20	1
100	1

Note that I have my first swap! I have collected sticker 45 twice and so the count is pushed up to 2.

The collection will be considered complete when the number of keys in the dictionary is equal to the number of stickers available. As an example for a 5 sticker book (very small book!), the dictionary might look like this.

Sticker Number (key)	Count (value)
1	5
2	10
3	1
4	7
5	3

So I have a lot of swaps (particularly sticker 2) and only one of sticker 3. Presumably sticker 3 was the last sticker I was waiting for. So our algorithm will complete when the dictionary size equals the sticker book size.

Add a dictionary called **stickers** to our function as shown in line 4.

```
2  def collectStickers(bookSize):
3      stickersBought = 0
4      stickers = dict()
5
6      return stickersBought
```

Loop Collecting Stickers

We will use a **while** loop to check if the dictionary size is equal to the book size. The loop will continue *buying stickers* until the collection is complete. We should count each sticker bought by incrementing the **stickersBought** variable.

```
7      while not (len(stickers) == bookSize):
8          stickersBought += 1
```

Generate Random Stickers

We are going to make the sticker selection completely random, so we will need the **randint** function from the **random** library in Python. Add the following import to the top of your code.

```
1   from random import randint
2
3   def collectStickers(bookSize):
4       stickersBought = 0
```

Now that we have access to the random number library, we should use it to *buy a sticker*. Line 9 will choose a random number from 1 to **bookSize** and put it into the variable **s**.

```
7       while not (len(stickers) == bookSize):
8           stickersBought += 1
9           s = randint(1, bookSize)
```

We must now use this randomly selected sticker to mark our dictionary. If we haven't collected this sticker before, we should create a new entry in our dictionary with a count of 1. If we **have** seen this sticker before then we should increment the count already there.

Lines 10 through to 13 show how this works. Notice the use of the **if <> in <>** statement.

```
7       while not (len(stickers) == bookSize):
8           stickersBought += 1
9           s = randint(1, bookSize)
10          if s in stickers:
11              stickers[s] += 1
12          else:
13              stickers[s] = 1
```

That should be enough, let us have a look at the completed function.

```
1   from random import randint
2
3   def collectStickers(bookSize):
4       stickersBought = 0
5       stickers = dict()
6
7       while not (len(stickers) == bookSize):
8           stickersBought += 1
9           s = randint(1, bookSize)
10          if s in stickers:
11              stickers[s] += 1
12          else:
13              stickers[s] = 1
14
15      return stickersBought
```

Call our Function

If we run this source code now, nothing will happen. We have written our function to simulate the collection of stickers, but we still need to **invoke** it. Since we are now adding code outside of the function, the indentation should go back to the left hand edge of the text file.

Add line 17 as shown

```
15      return stickersBought
16
17  x = collectStickers(400)
18
19
```

If you run this code, it will call our function, but not print anything. So we will now add a print to see what happened!

```
17  x = collectStickers(400)
18  print("Bought {} stickers for a collection of 400".format(x))
19
```

Run your code again and hopefully you will see something like this. This screenshot is of me running the code from the terminal in Ubuntu Linux.

```
joe@lister:~/git/KidsProjects/Tutors/Joe_Sharp$ python3 StickerCollectorProcedural_tutorial.py
Bought 2717 stickers for a collection of 400
```

Let's zoom in

```
Bought 2717 stickers for a collection of 400
```

Put sticker size in variable

Those last two lines of code use the value 400 twice. I would rather put that into a variable to ease maintenance.

- Create a new variable called **myBookSize** and set it to 400.
- Use the variable in the call to **collectStickers**
- Edit the format string and format arguments to pass in the **myBookSize** variable.

The code should end up like this.

```
17  myBookSize = 400
18  x = collectStickers(myBookSize)
19  print("Bought {} stickers for a collection of {}".format(x, myBookSize))
```

Run it again and check it still works.

Try Yourself

Running this simulation once gives you a very small sample of how many stickers you are likely to need to complete a single collection. It would be better to call the function many times and take an average of the number of stickers bought.

Call the **collectStickers** function in a **for x in range(y)** type of loop and add up the total number of stickers bought. Then divide that number by the number of attempts (shown as **y** here) to get the average. Then print the average.

I will let you figure out the code for this step. You should have all the tools you need, good luck.