Daniel Williamson

Anthony Peloso

11/19/15

CS126L: Section 3

Lab 10: Casino Night

1. Understanding the Problem

Create classes that would model objects you’d find in a casino. It will consist of a deck of cards and also a chip bank that will hold chips used for betting. The card will be either face up or faced down. Assuming the deck has 52 cards with four suits. Assuming that the ace is high. The bank cannot go negative but will max out the bank instead. The black chips are worth 100, green chips are worth 25, red chips are worth 5, and blue chips are worth 1.

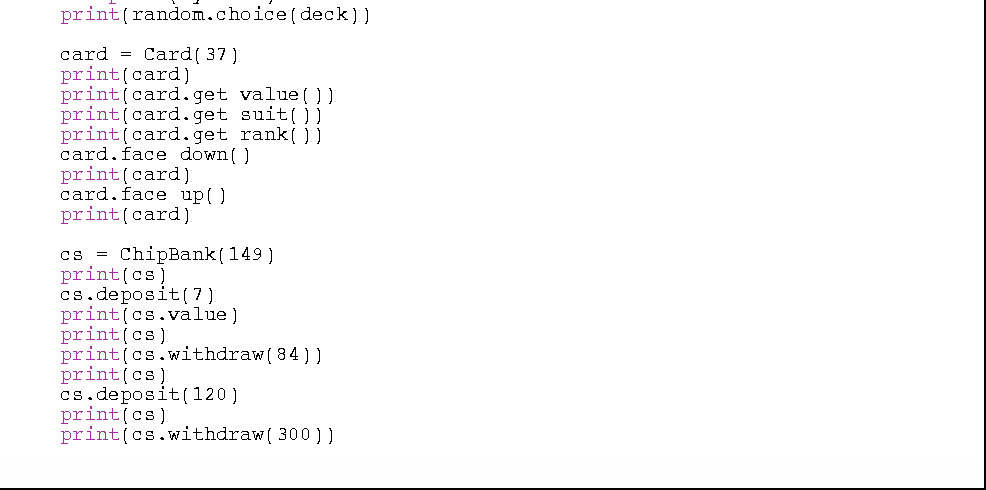
1. Planning

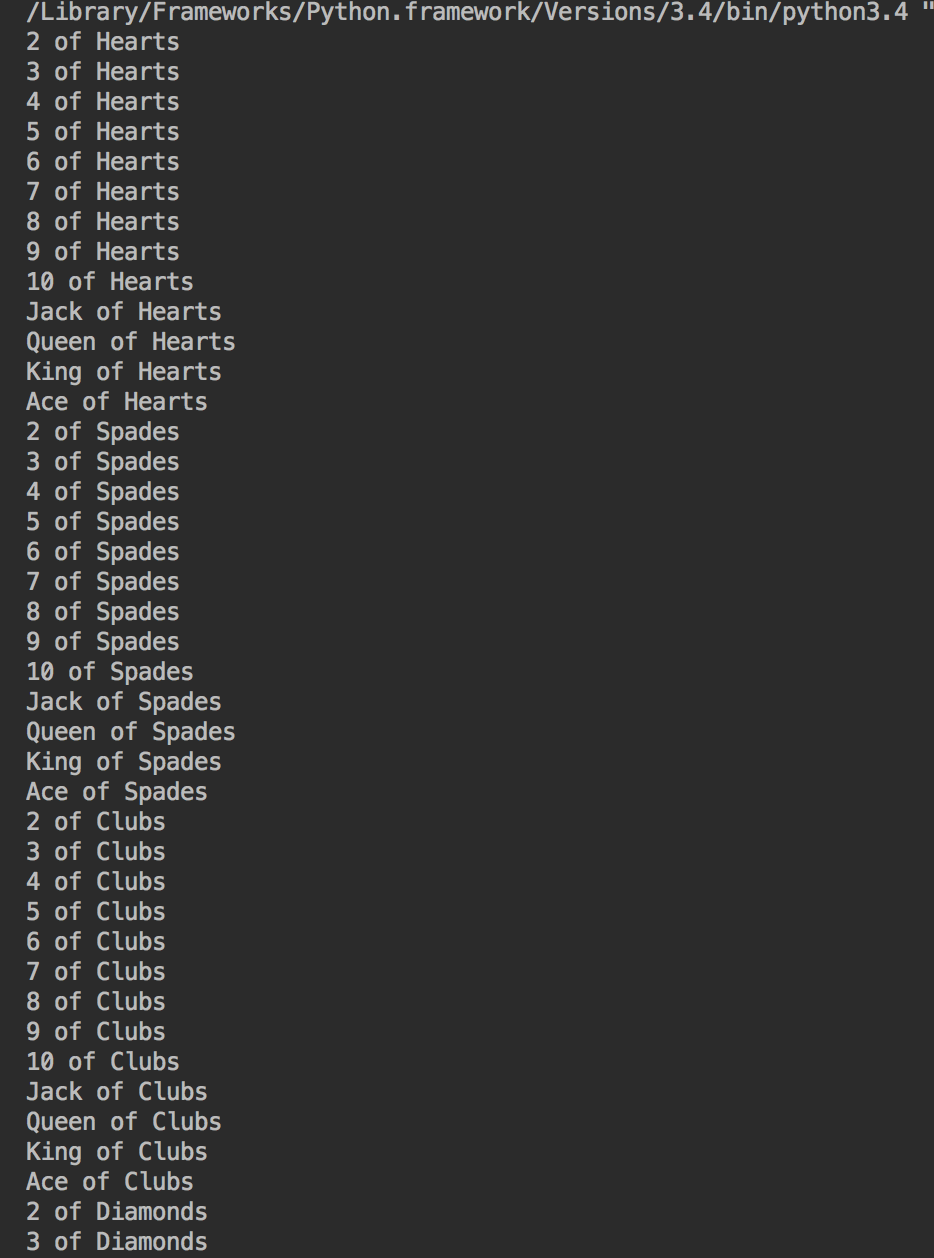
The use of import random was planned for later in the code to randomly select a card in the deck. The use of class was planned to hold all other functions needed for our cards. For the class, we planned on passing in self, as well as, the card number. Within our class, we planned on starting with initializing our function and then we planned to create lists that hold the cards suit, name, and value to be called upon later. We also planned on setting the face up to True by default and setting it to face down if wanted. Creating the methods to get the cards suit, rank, value, face up or down, and returning the suit and name if it was face up. From the requirements, we knew we had to plan a class called ChipBank that would store the value and be able to withdraw from it. We planned on using a method “withdraw” and conditional statements to prevent the bank going negative. Also, we knew we had to have another method to add chips to the bank. Our last method associated each chip color to it’s value. We figured out how to get those values by using floored division and mod division.

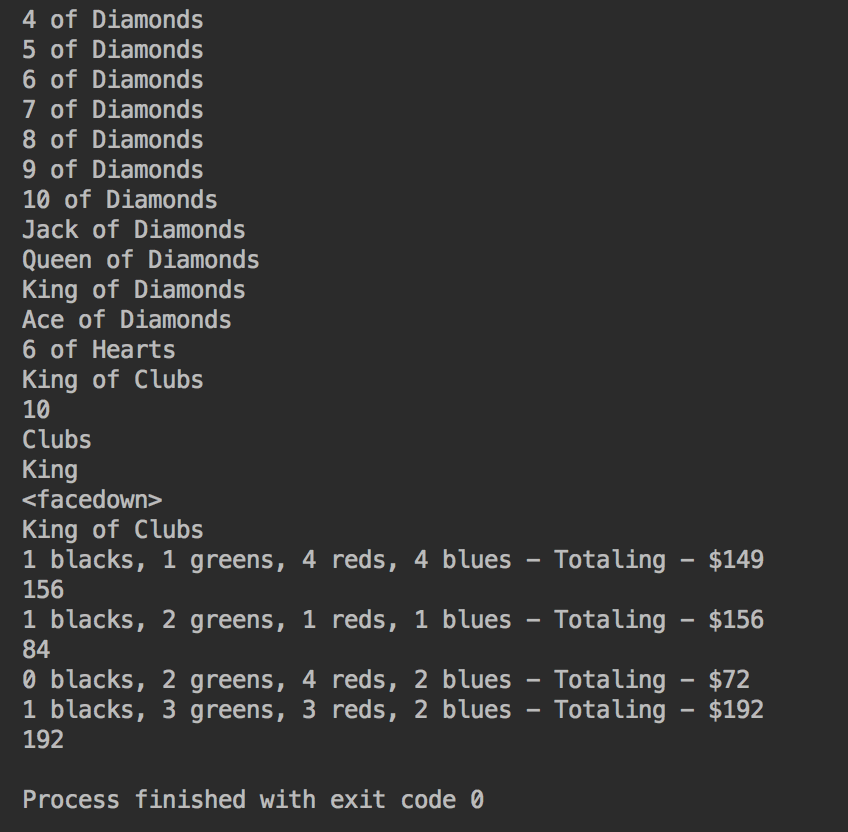
1. Implementation and Testing

We implemented our code by using classes and methods for suit, rank, value, face up, and face down.









Our code took in the test and printed each value with their suit. Printed the card being faced down and returned the value based on the chips.

1. Reflection/Refactoring

The main success to our code was the use of classes. We initialized the class by using init. Creating lists for our suit, value, and our card name. We had to first figure out how to associate the number to the suit and then the name. We used floored division plus mod division to do so. We decided to default our card as face up and make it face down if wanted. We created methods to get the information needed like suit, rank, value, and whether it was face up or down. Before going further, we tested to make sure that our code worked. We went back and adjusted a few issues and renamed variables to be clean and easy to understand.

After getting the first class to our liking, we went ahead and started the second class, “ChipBank”. We used if and elif statements that would either allow withdraws if the bank had sufficient chips or prevent the bank from going negative and only withdrawing what was in the bank. We also created a method that would allow deposits into the bank. This was relatively easy by just appending the deposit with the value of the bank. Finally, we used an example from a previous homework assignment to break up the value in the bank into their chip value. We ran into issues because the method seemed to want everything into a string format. We tried making the integer into a string in the return statement but it didn’t care for that. We ended up moving the str() inside the method and then returning. After getting the ChipBank to work, we went back and adjusted the names of our variables. We added the test to our code and ran it. We were very happy with the outcome and ran into small pep8 issues.