CSC108H Lab 8: Classes

At the end of the lab, please show your work to your TA and return this handout. As usual, we will post the handout on the course website at the end of the week. Switch driver and navigator frequently.

Objectives

Understand how to create an instance of a class and call methods on it.

Solidify your understanding of terminology: "class" and "type" are synonyms; "instance", "object", and "value" are synonyms; "instance of a class" means the same thing as "value of a type".

Understand when methods `__init__`, `__str__`, and `__cmp__` are called automatically.

Basics

1. Download `card.py` and `deck.py` from the labs webpage. Get help on `Card` and `Deck`.

2. In the Wing editor, create a program `basics.py` that imports card and deck. In the main body of the program, do the following:

• Create a `Card` with suit "H" and rank "5".

• Create another `Card` with suit "D" and rank "8".

• Print the two cards, using plain old print.

• Now flip both cards over and print them out a second time.

• Use < to compare the two cards and print a message if the first one is smaller.

3. Run your program to see if it does what you expect. Why do you see XX when you print a `Card`? Hint: What method is called?

4. Now use the debugger to step through and observe which methods are called. You can **Step Over** the import statements, but once you get to the `if __name__ == "__main__"` use **Step Into**. The debugger demonstrates that three methods are executed although you probably didn't call any of them explicitly. Be prepared to explain to your TA which methods are called and why.

5. For the particular cards that you created in `basics.py`, the comparison comes out right, but the `__cmp__` function has a small bug in it. Take a look at the code and find a pair of cards that will not compare correctly. Test your hypothesis and, if you're correct, fix `__cmp__`. Try the comparison again to verify that your update really did fix the problem. (And compare the 5 of Hearts and the 8 of Diamonds again to make sure that your fix didn't break what was working!)

6. Add the follow functionality to `basics.py`:

• Create a Deck of Cards.

• Shuffle the Deck you created.

• Create a list and deal 5 Cards into it. Turn each card over after dealing it.

• Print all of the Cards in the list, one per line.

7. Run your program again to see if it does what you expect. If you detect any problems, use the debugger to step through your program and inspect each of the functions being called.

Class Player

1. Download `player.py` from the labs webpage. A Player keeps track of a list of the cards that player was dealt and a list of cards that player has won during the game. This class is incomplete; your next task is to complete it.

2. Complete method `total_cards`. Notice that this method, like all methods, has `self` as its first parameter. This particular method does not need any additional parameters in order to do its job. When referring to variables in the object, don't forget to prefix their name with the keyword self.

3. Complete method `win_hand`. This method needs to know the cards that this player has won, so it has a parameter in addition to `self`. Hint: Use the list method `extend`.

4. Complete method `play_card`. Hint: Use method `pop`, but be sure not to call it on an empty list. Finally, complete method `__cmp__`.

The game of "War"

War is a card game for two players. The entire deck is dealt to the two players, each getting a face-down stack of cards. Each player turns over their top card, and the one whose top card has the higher value keeps both cards. If there is a tie, they have a "war": each puts two more cards face down, and then another face-up. The player whose final face-up card is higher wins. If there is another tie, they continue the war until there is no tie. The game ends when one of the players runs out of cards — they are the loser. If you have never played War, grab the deck of cards from your TA and try it.

1. Download `war.py` from the labs webpage. It imports the `Player` class that you completed. Run `war.py` to make sure that you implemented `Player` correctly.

That is all! See you next week!