Assignment 3

Reading

Review Chapter 8 in **Introduction to Computing using Python: An Application Development Focus** by Ljubomir Perković.

Logistics

Please read the Academic Integrity pledge prior to beginning this assignment. It is crucial that you understand what kind of collaboration is allowed and what kind is disallowed on assignments. The Academic Integrity pledge can be found on the D2L site.

In this class programming assignments may be completed in consultation with up to two other classmates. You must identify the classmates with whom you collaborate in a comment at the top of the assignment, and the number of collaborators on any assignment **may not exceed two other people**. You must also submit a comment in your submission for each assignment that describes in detail how each collaborator contributed to the assignment. If you did not collaborate with anyone on the assignment, you must include a comment that says that. You may not under any circumstances discuss the assignments with classmates other than your identified collaborators. Working so closely with anyone other than your identified collaborators so as to produce identical or near identical code is a violation of the Academic Integrity policy. This policy will be strictly enforced.

Please include the following with your assignment submission:

1. A comment at the top of your Python file identifying any classmates with whom you discussed or in any other way collaborated on the assignment. You may work (directly or indirectly) with **no more than two** other people.
2. Add a comment at the top of your Python file that describes for each person what they contributed to the assignment. This must be at least 2-3 sentences and be **very specific and detailed**.

A submission that does not include a list of collaborators and a comments indicating how you collaborated with classmates will earn a 0. If you worked alone you must put a comment at the top of your file that indicates that or you will also receive a 0. There will be no exceptions to this rule.

Again, you are subject to all the rules specified in the Academic Integrity pledge. Please read it carefully before beginning this assignment.

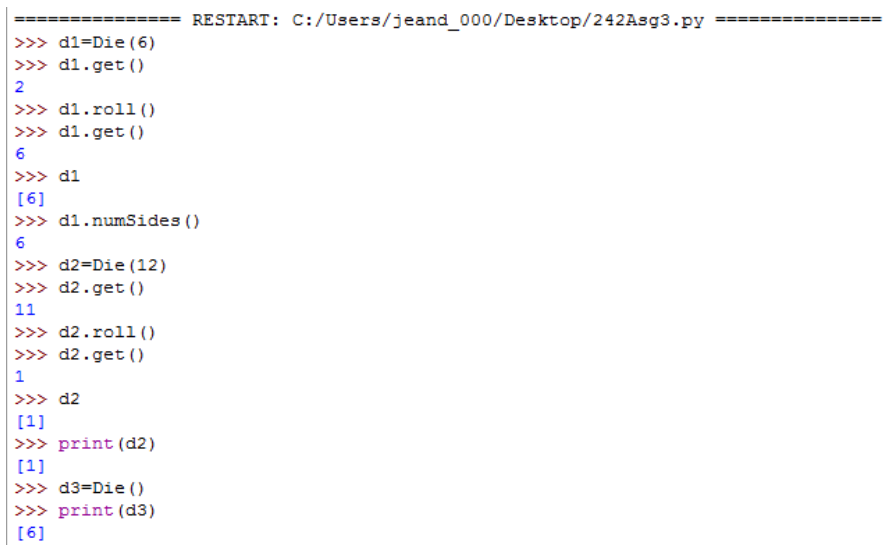
Assignment

Implement the classes and functions below and save them into a file called **[your last name]3.py**. I will not be providing a template for this assignment so you need to create the class, method, and function headers to meet the requirements described below. Please follow the formatting conventions found in the examples discussed in class. **You must also include appropriate doc strings** (e.g. strings that appear on the line following the class or method header) for the class and functions that clearly and concisely describe what the class and functions are doing. A submission without doc strings will not earn full credit.

Do not define any variables in the classes below (e.g. self.x for some variable name x) other than the ones specified in the assignment description. If you believe you can't write a solution without defining extra variables, please ask so that I can give you suggestions. You are absolutely encouraged to use as many local variables (e.g. x for some variable name x) as you desire.

1. Implement a class **Die** (that is a subclass of the object class) representing a die and supporting six methods:
    a. **__init__**() which either takes no parameters or takes an integer representing the number of sides of the die as a parameter. The constructor sets the number of sides of the object to the provided parameter or to the value 6 if no parameter is provided. The constructor also sets a variable representing the side of the die showing, which you can also think of as the current roll of the die, by calling a method from the random module that generates a random integer in the range [1, number of sides] where number of sides is the number of sides passed to the constructor or 6 if no parameter was provided.
    b. **get**() which returns the side of the die currently showing, that is, the current roll of the die object. It does NOT modify the roll but instead simply returns its value.
    c. **roll**() which simulates a new roll of the die by calling a method from the random module to generate a random integer in the range [1, number of sides] where number of sides is the number of sides stored in the die object and resets the variable in the object using that randomly generated value.
    d. **numSides**() which returns the number of sides of the die.
    e. **__repr__**() which returns the side of the die currently showing, i.e. the current roll of the die as shown in the sample output below.
    f. **__str__**() which returns the side of the die currently showing, i.e. the current roll of the die as shown in the sample output below.

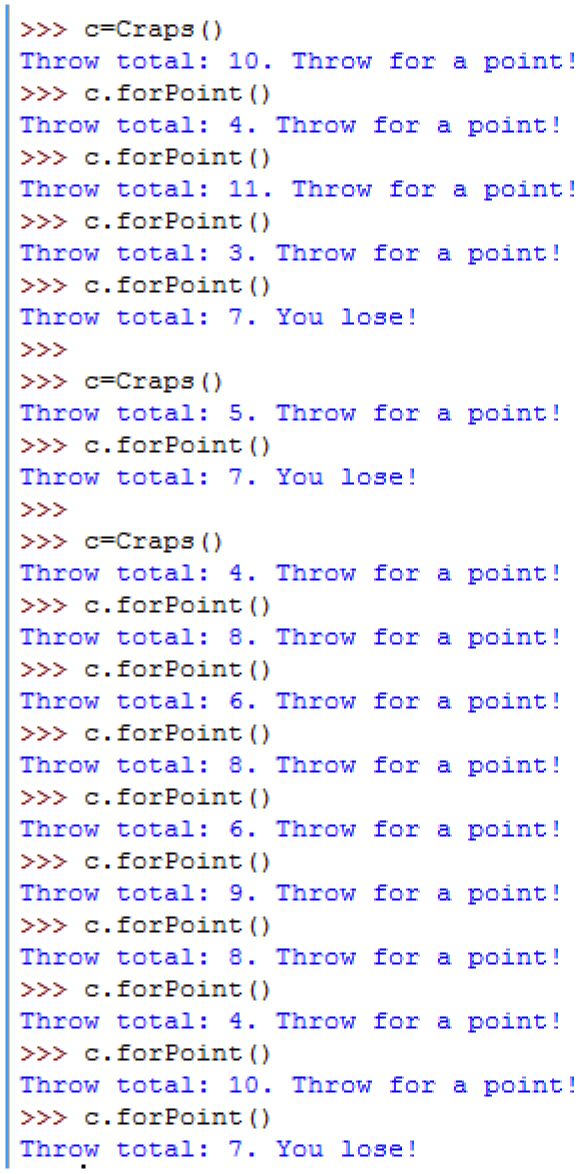
    The following shows how the **Die** class and its methods could be used:



```
=============== RESTART: C:/Users/jeand_000/Desktop/242Asg3.py ===============
>>> d1=Die(6)
>>> d1.get()
2
>>> d1.roll()
>>> d1.get()
6
>>> d1
[6]
>>> d1.numSides()
6
>>> d2=Die(12)
>>> d2.get()
11
>>> d2.roll()
>>> d2.get()
1
>>> d2
[1]
>>> print(d2)
[1]
>>> d3=Die()
>>> print(d3)
[6]
```

2. Craps is a dice-based game played in many casinos. Like blackjack, a player plays against the house. The game starts with the player throwing a pair of standard, six-sided dice. If the player rolls a total of 7 or 11, the player wins. If the player rolls a total of 2, 3, or 12, the player loses. For all other roll values, the player will repeatedly roll the pair of dice until either she/he rolls the initial value again (in which case she/he wins) or 7 (in which case she/he loses). Develop a class **Craps** that uses the Die class developed above to simulate a game of craps. The class should be a subclass of the object class, not Die. The class supports two methods:
    a. **__init__**() which takes no parameters. It creates two six-sided Die objects and stores them into two variables in the object. It then stores the sum of the initial rolls of the two dice into a variable in the class representing the initial roll of the player. If the initial roll is 7 or 11, the constructor reports that the player wins. If the initial roll is 2, 3, or 12, the constructor reports that the player loses. If the initial roll is any other value the constructor suggests that the player throw for point.
    b. **forPoint**() which takes no parameters and rolls the next round of the craps game. If the player rolls a 7, the method reports that the player lost. If the player rolls the same value as the initial roll, the method reports that the player wins. Otherwise the method suggests that the player again roll for point.

The following shows how the **Craps** class and methods could be used. Note that the situation where the user immediately loses is not represented although you have to handle that case in your implementation:



```
>>> c=Craps()
Throw total: 10. Throw for a point!
>>> c.forPoint()
Throw total: 4. Throw for a point!
>>> c.forPoint()
Throw total: 11. Throw for a point!
>>> c.forPoint()
Throw total: 3. Throw for a point!
>>> c.forPoint()
Throw total: 7. You lose!
>>>
>>> c=Craps()
Throw total: 5. Throw for a point!
>>> c.forPoint()
Throw total: 7. You lose!
>>>
>>> c=Craps()
Throw total: 4. Throw for a point!
>>> c.forPoint()
Throw total: 8. Throw for a point!
>>> c.forPoint()
Throw total: 6. Throw for a point!
>>> c.forPoint()
Throw total: 8. Throw for a point!
>>> c.forPoint()
Throw total: 6. Throw for a point!
>>> c.forPoint()
Throw total: 9. Throw for a point!
>>> c.forPoint()
Throw total: 8. Throw for a point!
>>> c.forPoint()
Throw total: 4. Throw for a point!
>>> c.forPoint()
Throw total: 10. Throw for a point!
>>> c.forPoint()
Throw total: 7. You lose!
```

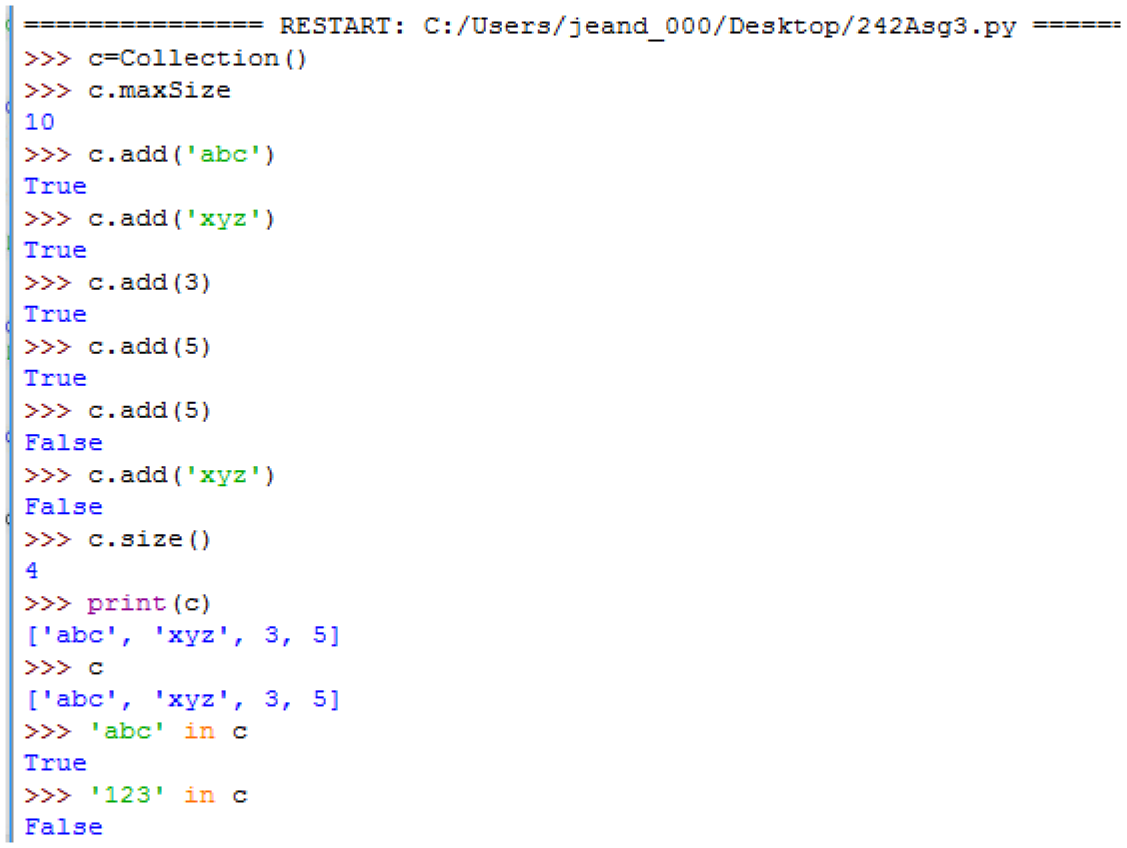
3. Write a class **Collection** that represents a collection of items. The class supports six methods:

   a. **__init__**() which takes an optional parameter representing the maximum size of the collection. It sets the maximum size of the collection to the provided value or to the value 10 if no number is given as a parameter. It also creates an empty list to store the items in the collection.

   b. **__contains__**() which takes an item as a parameter and returns True if the item is already in the collection and False otherwise. The function must return a Boolean and not a string.

   c. **add**() which adds a new item to the collection. It does this by first checking that the maximum capacity for the collection hasn't been reached. If the maximum capacity has been reached, then the method returns False without modifying the collection. If the maximum capacity has not yet been reached, the method checks that the item is not already in the collection using the

__contains__ method described above. If the item is already there, the method returns False and does not modify the collection. If the item is not present and the maximum capacity has not yet been reached, the method adds the item to the collection and returns True. The function must return a Boolean and not a string.
d. **size**() which returns the number of items currently in the collection.
e. **__str__**() which returns a string representing the object as seen in the sample output below.
f. **__repr__**() which returns a string representing the object as seen in the sample output below.



```
=============== RESTART: C:/Users/jeand_000/Desktop/242Asg3.py ======
>>> c=Collection()
>>> c.maxSize
10
>>> c.add('abc')
True
>>> c.add('xyz')
True
>>> c.add(3)
True
>>> c.add(5)
True
>>> c.add(5)
False
>>> c.add('xyz')
False
>>> c.size()
4
>>> print(c)
['abc', 'xyz', 3, 5]
>>> c
['abc', 'xyz', 3, 5]
>>> 'abc' in c
True
>>> '123' in c
False
```

Submitting the assignment

You must submit the assignment using the assignment 3 dropbox on the D2L site. Submit a Python file ([your last name]p3.py) with your implementation in it and comments describing your collaboration status. Submissions after the deadline listed above will be automatically rejected by the system. See the syllabus for the grading policy.

Grading

The assignment is graded according to the programming rubric which can be found on d2l associated with the dropbox.