Week 11 In-Class Exercise

1. Define class Player such that there are 3 attributes:
   - name: name of player
   - num_wins: number of game wins
   - num_plays: number of game plays

   In addition, define __str__ function in class Player to print instances of Player objects as shown on the right.

run_player.py	Output

`from player import *`
 	
`ann = Player('Ann', 2, 4)`
`bob = Player('Bob', 3, 5)`
`print(ann)`
`print(bob)`
`print(ann.name)`
`print(bob.num_wins)`
	Ann: Wins = 2: Plays = 4
	Bob: Wins = 3: Plays = 5
	Ann
	3

2. Continue to run instances of player 'Ann' as followed.
   What do you think the output of the last two lines would be?

run_player.py	Output

`from player import *`
`ann = Player('Ann', 2, 4)`
`bob = Player('Bob', 3, 5)`
`print(ann)`
`print(bob)`
 	
`ann.name = 'Anna'`
`print(ann)`
`ann.num_wins = 0`
`print(ann)`
	Ann: Wins = 2: Plays = 4
	Bob: Wins = 3: Plays = 5
	___: Wins = __: Plays = __
	___: Wins = __: Plays = __

3. Change code inside class Player to prevent values of attributes "not to be changed" from outside class.
   After you change the code inside class Player, you should get the output as shown on the right hand side.
   Note that the attributes that cannot be changed from outside class is considered "private" attribute; otherwise, they are considered as "public."

run_player.py	Output

`from player import *`
`ann = Player('Ann', 2, 4)`
`bob = Player('Bob', 3, 5)`
`print(ann)`
`print(bob)`
	Ann: Wins = 2: Plays = 4
	Bob: Wins = 3: Plays = 5
	Ann: Wins = 2: Plays = 4
	Ann: Wins = 2: Plays = 4

```
ann.name = 'Anna'
print(ann)
ann.num_wins = 0
print(ann)
```

4. In the case that we allow the outside class to change values of **name** and **num_wins**, add getter (or accessor) and setter (or mutator) functions to **name** and **num_wins** in the class Player.

run_player.py

```
from player import *
ann = Player('Ann', 2, 4)
bob = Player('Bob', 3, 5)
print(ann)
print(bob)

ann.set_name('Anna')
print(ann)
ann.set_num_wins(3)
print(ann)
print(ann.get_num_wins())
print(ann.get_name())
print(bob.get_name())
```

Output

```
Ann: Wins = 2: Plays = 4
Bob: Wins = 3: Plays = 5
Anna: Wins = 2: Plays = 4
Anna: Wins = 3: Plays = 4
3
Anna
Bob
```

5. If we allow the outside class to change values of **num_plays** as well, set property for functions **num_plays**, so that they are equivalent to getter (or accessor) and setter (or mutator) functions of **num_plays** in the class Player.

   Note that with this property setting, values of private attributes can be used as if they are public attributes.
   In object-oriented programming, you need to be careful of attributes that can be changed from outside class. Do not set up all attributes, so that they can be changed from outside class. Restrict the outside class to change only needed attributes.

run_player.py

```
from player import *
ann = Player('Ann', 2, 4)
bob = Player('Bob', 3, 5)
print(ann)
print(bob)

ann.set_name('Anna')
print(ann)
ann.set_num_wins(3)
print(ann)
print(ann.get_num_wins())
ann.num_plays = 5
print(ann)
print(ann.num_plays)
```

Output

```
Ann: Wins = 2: Plays = 4
Bob: Wins = 3: Plays = 5
Anna: Wins = 2: Plays = 4:
Anna: Wins = 3: Plays = 4
3
Anna: Wins = 3: Plays = 5
5
```

6. Add another attribute called **hand** to store value of player's hand. Assign default value of hand to be 'None'. Also, set up property for functions hand, so that value of **hand** can be changed from outside class.

run_player.py	Output

`from player import *`
`ann = Player('Ann', 2, 4)`
`bob = Player('Bob', 3, 5)`
`print(ann)`
`print(bob)`
`print(ann.hand)`
`print(bob.hand)`
	Ann: Wins = 2: Plays = 4: Hand = None
	Bob: Wins = 3: Plays = 5: Hand = None
	None
	None

7. Add a function randomize_hand to class Player, so that this function will randomize integer between 1 and 3.

Random Value	Hand Value
1	'Rock'
2	'Paper'
3	'Scissors'

To generate random integer x where $m \leq x \leq n$, do as followed:

```
import random

x = random.randint(m,n)  # plug in needed values for m and n
# x will be randomly generated integer between m <= x <= n
```

Given tested code as followed.

Note that you may not get Paper and Scissors as shown below. Run several times to see whether you randomly generate value for hand.

run_player.py	Sample Output

`from player import *`
`ann = Player('Ann', 2, 4)`
`bob = Player('Bob', 3, 5)`
`print(ann)`
`print(bob)`
`ann.randomize_hand()`
`print(ann)`
`bob.randomize_hand()`
`print(bob)`
	Ann: Wins = 2: Plays = 4: Hand = None
	Bob: Wins = 3: Plays = 5: Hand = None
	Ann: Wins = 2: Plays = 4: Hand = Paper
	Bob: Wins = 3: Plays = 5: Hand = Scissors

8. Create another class called Team such that there are 4 attributes:
    - player_list: list of players. Player information will be read from file.
    - team_name: name of team
    - team_points: points of team. Set default value to be zero.
    - current_player: current player as the representative of the team

The partial code of the Team class is given below:

team.py

```
import random
from player import *
class Team:
      def __init__(self, filename, team_name='No Name'):
            self.__player_list = self.__read_team(filename)
            ???

      def __read_team(self, filename):
            player_list = []
            ???
            return player_list

      def __str__(self):
            ???
```

Run run_team.py and its output are given below

run_team.py

```
from team import *
team_a = Team('team_a.txt','A')
print(team_a)

team_b = Team('team_b.txt','B')
print(team_b)
```

Sample Output

```
Team A
Team Points: 0
Ann: Wins = 1: Plays = 3: Hand = None
Bob: Wins = 2: Plays = 5: Hand = None
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
Team Points: 0
David: Wins = 3: Plays = 4: Hand = None
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 6: Hand = None
```

9. In class Team, add function select_player.
   - Inside this function, randomly choose one player from the list.
   - Then assign the randomly-chosen player to the current player of the team.
   - After choosing one player, randomly generate hand value for the chosen player.

team.py

```
import random
from player import *
class Team:
          …

          def select_player(self):
             ???
```

Run run_team.py and its output are given below

run_team.py	Sample Output

`from team import *`
`team_a = Team('team_a.txt','A')`
`print(team_a)`
 	
`team_b = Team('team_b.txt','B')`
`print(team_b)`
 	
`team_a.select_player()`
`print(team_a.current_player)`
 	
`team_b.select_player()`
`print(team_b.current_player)`
	Team A
	Team Points: 0
	Ann: Wins = 1: Plays = 3: Hand = None
	Bob: Wins = 2: Plays = 5: Hand = None
	Charlie: Wins = 0: Plays = 1: Hand = None
	
	Team B
	Team Points: 0
	David: Wins = 3: Plays = 4: Hand = None
	Eric: Wins = 0: Plays = 3: Hand = None
	Francis: Wins = 2: Plays = 5: Hand = None
	Gary: Wins = 1: Plays = 6: Hand = None
	
	Charlie: Wins = 0: Plays = 1: Hand = Scissors
	David: Wins = 3: Plays = 4: Hand = Scissors

10. In run_team.py, add a function called find_winner.
    In find_winner, check player from which team is the winner based on their hand values.
    If the player from the first team wins, return 1.
    If the player from the second team wins, return 2.
    If the players from both teams tie, return 0.

Run run_team.py and its output are given below

run_team.py

```
def find_winner(first_player, second player)
     ???

from team import *
team_a = Team('team_a.txt','A')
print(team_a)

team_b = Team('team_b.txt','B')
print(team_b)

team_a.select_player()
print(team_a.current_player)

team_b.select_player()
print(team_b.current_player)

winning_team = find_winner(team_a.current_player,
                           team_b.current_player)
print(winning_team)
```

Sample Output

```
Team A
Team Points: 0
Ann: Wins = 1: Plays = 3: Hand = None
Bob: Wins = 2: Plays = 5: Hand = None
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
Team Points: 0
David: Wins = 3: Plays = 4: Hand = None
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 6: Hand = None

Ann: Wins = 1: Plays = 3: Hand = Rock
David: Wins = 3: Plays = 4: Hand = Scissors
1
```

11. In class Team, add function update_team_points.
    - Inside this function, if the receive value parameter is 'win', update team_points and current player's num_wins. In addition, update the current player's num_plays.

team.py

```
import random
from player import *
class Team:
          …

          def select_player(self):
             ???

         def update_team_points(self,value):
              ???
```

Run run_team.py and its output are given below

run_team.py

```
from team import *
team_a = Team('team_a.txt','A')
print(team_a)

team_b = Team('team_b.txt','B')
print(team_b)

team_a.select_player()
print(team_a.current_player)

team_b.select_player()
print(team_b.current_player)

team_a.update_team_points('win')
print(team_a)
team_b.update_team_points('lose')
print(team_b)
```

Sample Output

```
Team A
Team Points: 0
Ann: Wins = 1: Plays = 3: Hand = None
Bob: Wins = 2: Plays = 5: Hand = None
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
Team Points: 0
David: Wins = 3: Plays = 4: Hand = None
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 6: Hand = None

Bob: Wins = 2: Plays = 5: Hand = Paper
Gary: Wins = 1: Plays = 6: Hand = Scissors
Team A
Team Points: 1
```

```
Ann: Wins = 1: Plays = 3: Hand = None
Bob: Wins = 3: Plays = 6: Hand = Paper
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
Team Points: 0
David: Wins = 3: Plays = 4: Hand = None
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 7: Hand = Scissors
```

12. In run_team.py, add a function called update_points.
    In update_points, print which team wins. If tie, print 'Both team tie'.
    Then, for each team, call update_team_points.

Run run_team.py and its output are given below

run_team.py

```
def find_winner(first_player, second player)
     ???

def update_points(winning_team, first_team, second_team):
     ???

from team import *
team_a = Team('team_a.txt','A')
print(team_a)

team_b = Team('team_b.txt','B')
print(team_b)

team_a.select_player()
print(team_a.current_player)

team_b.select_player()
print(team_b.current_player)

winning_team = find_winner(team_a.current_player,
                           team_b.current_player)
print(winning_team)

update_points(winning_team, team_a, team_b)
print(team_a)
print(team_b)
```

Sample Output

```
Team A
Team Points: 0
Ann: Wins = 1: Plays = 3: Hand = None
Bob: Wins = 2: Plays = 5: Hand = None
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
```

```
Team Points: 0
David: Wins = 3: Plays = 4: Hand = None
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 6: Hand = None

Ann: Wins = 1: Plays = 3: Hand = Rock
David: Wins = 3: Plays = 4: Hand = Scissors
1
A wins.
Team A
Team Points: 1
Ann: Wins = 2: Plays = 4: Hand = Rock
Bob: Wins = 2: Plays = 5: Hand = None
Charlie: Wins = 0: Plays = 1: Hand = None

Team B
Team Points: 0
David: Wins = 3: Plays = 5: Hand = Scissors
Eric: Wins = 0: Plays = 3: Hand = None
Francis: Wins = 2: Plays = 5: Hand = None
Gary: Wins = 1: Plays = 6: Hand = None
```

13. Rewrite run_team.py, so that both teams will play until one team gets team_points = 5 first. Then, the game will stop.