Discussion Quiz 7

OOP – Spring 2022 (Python)

Question 1 Statement

Consider class P_M_Record (saved in file p_m_record.py) and write class Player. Player class has class level members *count of players*, and data member's *player name, match count and a list having details of player's matches* (objects of P_M_Record).

Write *init* method without parameter, assume there are getter methods to get values. Assign values to player name and match count. Run loop for match count, get values for match details. Create object of match record and add into the list. Next write str function to return complete player object, see sample output for guidance:

Class Player_Match_Record

```
OUT = True
NOTOUT = False
class P_M_Record:   #Player Match Record
    count = 0
    def __init__(self, score, balls, fours=0, sixes=0, is_out=OUT):
        self.__score = score
        self.__balls = balls
        self.__is_out = is_out
        self.__fours = fours
        self.__sixes = sixes

    def __str__(self):
        s =f'{self.__score}\t{self.__balls}\t'
        if self.__is_out:   s+'Out'
        else:               s+'Notout'
        return s + f'\t{self.__fours}\t{self.__sixes}'
```

```
Player Name: Kashif
Number of Matches: 5
Runs Balls Fours Sixes
 68    39     13    0
  7    18      0    0
 119  111     18    1
   5    7      0    0
  70   54      0    0

Player Name: Azeem
Number of Matches: 4
Runs Balls Fours Sixes
 19   36     2    1
 91   38    14    4
119  102    24    1
120  101    23    4
```

Solution Question 1

```
from p_m_record import *
class Player:
    count_players = 0
 def __init__(self):
    Player.count_players += 1
    self.__player_name = get_player_name()
    self.__match_count = get_match_count()
    self.__mathes_details=[]
    for i in range(self.__match_count):
        record = P_M_Record(get_runs(), get_balls(), get_fours(), get_sixes())
        s self.__mathes_details.append(record)
```

Solution Question 1 – str function

```
def __init__(self):
    Player.count_players += 1
    self.__player_name = get_player_name()
    self.__match_count = get_match_count()
    self.__mathes_details=[]
    for i in range(self.__match_count):
        record = P_M_Record(get_runs(), get_balls(), get_fours(), get_sixes())
        s self.__mathes_details.append(record)
 def __str__(self):
    s = f'Playeyer Name: {self.__player_name}\n'
    s += f'Number of Matches: {self.__match_count}\n'
    s += f'Runs\tBalls\tFours\tSixes\n'
    for record in self.__mathes_details:
        s += str(record)+'\n'
    return s
```

```
Player Name: Kashif
Number of Matches: 5
Runs Balls Fours Sixes
 68    39     13    0
  7    18      0    0
 119  111     18    1
   5    7      0    0
  70   54      0    0

Player Name: Azeem
Number of Matches: 4
Runs Balls Fours Sixes
 19   36     2    1
 91   38    14    4
119  102    24    1
120  101    23    4
```

Question 2 Statement

Write a class Shapes. Shapes class has count of shape and a list having different shapes. You have class Line, Circle, Triangle, Rectangle saved in files 'line.py', 'circle.py' etc. Write following functions in Shapes class:

> ***init -*** with single parameter count of shapes. Run loop for count of shapes. Draw a random variable type. According to the type create one of the shape and add into the list
>
> ***draw -*** run loop and call draw function for all the objects in the list

On the right side of the page, create a box and write signature (first lines only) of *init* functions in the classes *Line*, *Circle*, *Triangle*, *Rectangle*.

Question 2 – Required Signatures

On the right side of the page, create a box and write signature (first lines only) of *init* functions in the classes *Line*, *Circle*, *Triangle*, *Rectangle*.

Line:

```
def __init__(self, screen, x1, y1, x2, y2)
```

Circle:

```
def __init__(self, screen, center_x, center_y, radius)
```

Rectangle:

```
def __init__(self, screen, x, y, width, height)
```

Triangle:

```
def __init__(self, screen, x1, y1, x2, y2, x3, y3)
```

Solution Question 2

```
from shape import *
from rectangle import *
from triangle import *
from line import *
from circle import *
from random import *
import pygame


class Shapes:
    def __init__(self, count):
        self.__count = count
        self.__shapes = []
        self.screen = py.display.set_mode((1200, 800))
```

Solution Question 2 - Continued

```
for i in range(count):
    type = randint(0,3)
    if type == 0:
        shape = Line(self.screen, randint(10, 400), randint(10, 400), randint(450, 790),
    elif type == 1:
        shape = Circle(self.screen, randint(500, 700), randint(300, 400), randint(50, 150))
    elif type == 2:
        shape = Rectangle(self.screen, randint(10, 400), randint(10, 300),
                                                    randint(100, 200), randint(100, 200))
    else:
        shape = Triangle(self.screen, randint(10, 300), randint(10, 200),
                randint(700, 1190), randint(150, 250), randint(350, 650), randint(450, 790))
    self.__shapes.append(shape)
```

Solution Question 1 – draw function

```
def draw(self):
    for shape in self.__shapes:
        shape.draw()
```

```
class Shapes:
    def __init__(self, count):
        self.__count = count
        self.__shapes = []
```