Assignment 07_20101197_AbirAhammedBhuiyan

October 3, 2022

[1]:
```
#task_1

class Student:
    def __init__(self, name='Just a student', dept='nothing'):
        self.__name = name
        self.__department = dept
    def set_department(self, dept):
        self.__department = dept
    def get_name(self):
        return self.__name
    def set_name(self,name):
        self.__name = name
    def __str__(self):
        return 'Name: '+self.__name+' Department: '+self.__department


class BBA_Student(Student):
    def __init__(self, name="default", dept='BBA'):
        super().__init__(name, dept)


print(BBA_Student())
print(BBA_Student('Humpty Dumpty'))
print(BBA_Student('Little Bo Peep'))
```

```
Name: default Department: BBA
Name: Humpty Dumpty Department: BBA
Name: Little Bo Peep Department: BBA
```

[8]:
```
#task_2

class Vehicle:
    def __init__(self):
        self.x = 0
        self.y = 0
    def moveUp(self):
        self.y+=1
    def moveDown(self):
```

```python
        self.y-=1
    def moveRight(self):
        self.x+=1
    def moveLeft(self):
        self.x-=1
    def __str__(self):
        return '('+str(self.x)+' , '+str(self.y)+')'

class Vehicle2010(Vehicle):
    def moveLowerLeft(self):
        super().moveDown()
        super().moveLeft()

    def moveUpperRight(self):
        super().moveUp()
        super().moveRight()

    def moveUpperLeft(self):
        super().moveUp()
        super().moveLeft()

    def moveLowerRight(self):
        super().moveDown()
        super().moveRight()

    def equals(self, obj):
        if(self.x == obj.x and self.y == obj.y):
            return True
        else:
            return False


print('Part 1')
print('------')
car = Vehicle()
print(car)
car.moveUp()
print(car)
car.moveLeft()
print(car)
car.moveDown()
print(car)
car.moveRight()
print(car)
print('------')
print('Part 2')
print('------')
```

```
car1 = Vehicle2010()
print(car1)
car1.moveLowerLeft()
print(car1)
car2 = Vehicle2010()
car2.moveLeft()
print(car1.equals(car2))
car2.moveDown()
print(car1.equals(car2))
```

```
Part 1
------
(0 , 0)
(0 , 1)
(-1 , 1)
(-1 , 0)
(0 , 0)
------
Part 2
------
(0 , 0)
(-1 , -1)
False
True
```

[3]:
```
#task_3

class Tournament:
    def __init__(self,name='Default'):
        self.__name = name
    def set_name(self,name):
        self.__name = name
    def get_name(self):
        return self.__name


class Cricket_Tournament(Tournament):
    def __init__(self, name='Default', teams=0, typ="No type"):
        super().__init__(name)
        self.teams = teams
        self.typ = typ

    def detail(self):
        return f"Cricket Tournament Name: {super().get_name()}\nNumber of Teams:
 ↪ {self.teams}\nType: {self.typ}"

class Tennis_Tournament(Tournament):
    def __init__(self, name='Default', players=0):
        super().__init__(name)
        self.players = players

    def detail(self):
        return f"Tennis Tournament Name: {super().get_name()}\nNumber of␣
 ↪Players: {self.players}"

ct1 = Cricket_Tournament()
print(ct1.detail())
print("-----------------------")
ct2 = Cricket_Tournament("IPL",10,"t20")
print(ct2.detail())
print("-----------------------")
tt = Tennis_Tournament("Roland Garros",128)
print(tt.detail())
```

```
Cricket Tournament Name: Default
Number of Teams: 0
Type: No type
-----------------------
Cricket Tournament Name: IPL
Number of Teams: 10
Type: t20
-----------------------
Tennis Tournament Name: Roland Garros
Number of Players: 128
```

[4]:
```
#task_4

class Product:
    def __init__(self,id, title, price):
        self.__id = id
        self.__title = title
        self.__price = price
    def get_id_title_price(self):
        return "ID: "+str(self.__id)+" Title: "+self.__title+" Price:␣
 ↪"+str(self.__price)

class Book(Product):
    def __init__(self, id, title, price, ISBN, publisher):
        super().__init__(id, title, price)
        self.ISBN = ISBN
        self.publisher = publisher

    def printDetail(self):
        return f"{super().get_id_title_price()}\nISBN: {self.ISBN} Publisher:␣
 ↪{self.publisher}"

class CD(Product):
    def __init__(self, id, title, price, band, duration, genre):
        super().__init__(id, title, price)
        self.band = band
        self.duration = duration
        self.genre = genre

    def printDetail(self):
        return f"{super().get_id_title_price()}\nBand: {self.band} Duration:␣
 ↪{self.duration}minutes\nGenre: {self.genre}"

book = Book(1,"The Alchemist",500,"97806","HarperCollins")
print(book.printDetail())
print("-----------------------")
cd = CD(2,"Shotto",300,"Warfaze",50,"Hard Rock")
print(cd.printDetail())
```

```
ID: 1 Title: The Alchemist Price: 500
ISBN: 97806 Publisher: HarperCollins
-----------------------
ID: 2 Title: Shotto Price: 300
Band: Warfaze Duration: 50minutes
Genre: Hard Rock
```

[5]:

```
#task_5

class Animal:
    def __init__(self, sound):
        self.__sound = sound

    def makeSound(self):
        return self.__sound


class Printer:
    def printSound(self, a):
        print(a.makeSound())

class Dog(Animal):
    def __init__(self, sound):
        super().__init__(sound)

class Cat(Animal):
    def __init__(self, sound):
        super().__init__(sound)



d1 = Dog('bark')
c1 = Cat('meow')
a1 = Animal('Animal does not make sound')
pr = Printer()
pr.printSound(a1)
pr.printSound(c1)
pr.printSound(d1)
```

```
Animal does not make sound
meow
bark
```

[6]:
```
#task_6

class Shape:
    def __init__(self, name='Default', height=0, base=0):
        self.area = 0
        self.name = name
        self.height = height
        self.base = base

    def get_height_base(self):
        return "Height: "+str(self.height)+", Base: "+str(self.base)

class triangle(Shape):
    def __init__(self, name='Default', height=0, base=0):
        super().__init__(name, height, base)

    def calcArea(self):
        self.area = 0.5*self.height*self.base

    def printDetail(self):
        return f"Shape name: {self.name}\n{super().get_height_base()}\nArea:␣
 ↪{self.area}"

class trapezoid(Shape):
    def __init__(self, name='Default', height=0, base=0, side_A=0):
        super().__init__(name, height, base)
        self.side_A = side_A

    def calcArea(self):
        self.area = ((self.side_A+self.base)/2)*self.height
    def printDetail(self):
        return f"Shape name: {self.name}\n{super().get_height_base()}, Side_A:␣
 ↪{self.side_A}\nArea: {self.area}"

tri_default = triangle()
tri_default.calcArea()
print(tri_default.printDetail())
print('---------------------------')
tri = triangle('Triangle', 10, 5)
tri.calcArea()
print(tri.printDetail())
print('----------------------------')
trap = trapezoid('Trapezoid', 10, 6, 4)
trap.calcArea()
print(trap.printDetail())
```

```
Shape name: Default
Height: 0, Base: 0
Area: 0.0
---------------------------
Shape name: Triangle
Height: 10, Base: 5
Area: 25.0
----------------------------
Shape name: Trapezoid
Height: 10, Base: 6, Side_A: 4
Area: 50.0
```

[7]:
```
#task_7

class Football:
    def __init__(self, team_name, name, role):
        self.__team = team_name
        self.__name = name
        self.role = role
        self.earning_per_match = 0
    def get_name_team(self):
        return 'Name: '+self.__name+', Team Name: ' +self.__team


class Player(Football):
    def __init__(self, team_name, name, role, t_g, t_p):
        super().__init__(team_name, name, role)
        self.t_g = t_g
        self.t_p = t_p

    def calculate_ratio(self):
        self.ratio = self.t_g/self.t_p

    def print_details(self):
        print(super().get_name_team())
        print(f"Team Role: {self.role}")
        print(f"Total Goal: {self.t_g}, Total Played: {self.t_p}")
        print(f"Goal Ration: {self.ratio}")
        print(f"Match Earning: {(self.t_g*1000)+(self.t_p*10)}K")


class Manager(Football):
    def __init__(self, team_name, name, role, t_w):
        super().__init__(team_name, name, role)
        self.t_w = t_w

    def print_details(self):
        print(super().get_name_team())
        print(f"Team Role: {self.role}")
        print(f"Total Win: {self.t_w}")
        print(f"Match Earning: {self.t_w*1000}K")

player_one = Player('Juventus', 'Ronaldo', 'Striker', 25, 32)
player_one.calculate_ratio()
player_one.print_details()
print('------------------------------------------')
manager_one = Manager('Real Madrid', 'Zidane', 'Manager', 25)
manager_one.print_details()
```

```
Name: Ronaldo, Team Name: Juventus
Team Role: Striker
Total Goal: 25, Total Played: 32
Goal Ration: 0.78125
Match Earning: 25320K
------------------------------------------
Name: Zidane, Team Name: Real Madrid
Team Role: Manager
Total Win: 25
Match Earning: 25000K
```

```
[ ]:
```