Chapitre 2

TP. Programmation Orientée Objet en Python

Corrigé

Exercice 1 :

```
#Q1
class Point :
    def __init__(self):
        self.x = 3.0
        self.y = 4.0
    #Q3
    def afficher(self):
        print("x={}, y={}".format(self.x, self.y))
#Q4
class Rectangle:
    def __init__(self, p, larg, haut):
        if isinstance(p,Point):
            self.coinSupG = p
            self.largeur = larg
            self.hauteur = haut
        else:
            raise TypeError("Type incorrect")
#Q6
def trouverCentre(r):
    p= Point()
    p.x = r.coinSupG.x + r.largeur /2
    p.y = r.coinSupG.y - r.hauteur /2
    return p
#Q2
p1 = Point()

#Q3 : Appel de la méthode afficher
p1.afficher()

#Q5
p = Point()
p.x = 12.0
p.y = 27.0
boite = Rectangle(p, 50.0, 35.0)

#Q7
print(trouverCentre(boite))

#Q8
boite.largeur /=2
boite.hauteur *=2
```

Exercice 2 :

```
#Q1
class Time :
    def __init__(self, h=0, m=0, s=0):
        self.heures = h
        self.minutes = m
        self.secondes = s
    #Q4; méthode : dans la classe => toujours avec self
    def affiche_heure(self):
        print("{}:{}:{}".format(self.heures,self.minutes,self.secondes))

#Q2
#nomObjet = nomClasse(param)
instant = Time(12, 32, 45)

#Q3; fonction : en dehors de la classe => pas de self
def affiche_heure(t):
    print("{}:{}:{}".format(t.heures,t.minutes,t.secondes))

#appel~Q3
#nomFonction(nomObjet)
affiche_heure(instant)

#Q5
maintenant = Time()
#nomObjet.nomMethode(param)
maintenant.affiche_heure()
#nomClasse.nomMethode(nomObjet,param)
Time.affiche_heure(maintenant)
```

Exercice 3 :

```
class CompteBancaire:
    def __init__(self,nom="salim",solde=1000):
        self.nom = nom
        self.solde = solde

    def depot(self,somme):
        assert isinstance(somme,float) or isinstance(somme,int)
        self.solde += somme

    def retrait(self,somme):
        assert isinstance(somme,float) or isinstance(somme,int)
        assert somme <= self.solde
        self.solde -= somme

    def __repr__(self): #un seul param toujours = self
        #retourne ==> return (pas print)
```

```
        #une seule ==> pas de virgule
        #chaine ==> de type str
        ch = "Le solde du cpte Bancaire de" + self.nom +" est de
"+str(self.solde)+"dinars"
        return ch



#appel
cb = CompteBancaire()
```

Exercice 4 :

```
class Voiture:
    def __init__(self,marque="Ford",couleur="rouge"):
        self.marque = marque
        self.couleur = couleur
        self.pilote = "personne"
        self.vitesse = 0

    def choix_conducteur(self,pilote):
        self.pilote = pilote

    def accelerer(self,taux,duree):
        assert self.pilote != "personne"
        self.vitesse += taux * duree

    def affiche_tout(self):
        ch = "{} {} pilotée par {}, vitesse = {} m/s"
        print(ch.format(self.marque,self.couleur\
        ,self.pilote,self.vitesse))

    def __str__(self): #prend tjs un seul paramètre : self
        #retourne tjs une seule chaine str
        if self.pilote == "personne"
            return "Cette voiture n'a pas de conducteur"
        else
            ch = "{} {} pilotée par {}, vitesse = {} m/s"
            return ch.format(self.marque,self.couleur,\
            self.pilote,self.vitesse)

v = Voiture ("Peugeot","Bleu")
v.choix_conducteur("Ali")
```