Corrigé -exercice 1

```
class Cellule:
    def __init__(self,valeur = None,suivant=None):

        self.valeur=valeur
        self.suivant=suivant

class Pile():
    '''
    le code de l'implémentation n'est pas demandé.
    on le donne pour tester
    '''

    def __init__(self):
        self.sommet = None

    def estVide(self):
        return self.sommet == None

    def empiler(self,valeur):

        self.sommet = Cellule(valeur,self.sommet)

    def depiler(self):

        if not self.estVide():
            x = self.sommet
            self.sommet = self.sommet.suivant
            return x.valeur
        return None

    def retournerSommet(self):

    if not self.estVide():
        return self.sommet.valeur
    return None


    def nb_elements(self):

        compteur = 0
        cellule_courante = self.sommet
        while cellule_courante != None:
            compteur += 1
            cellule_courante = cellule_courante.suivant
        return compteur



#Q1
pile1=Pile()
pile1.empiler(7)
pile1.empiler(5)
pile1.empiler(2)
```

```
b) affichage : 7 , 5 ,5 ,2


#Q2
cas1 : 3 2
cas2 : 3 2 5 7
cas3 :3
cas4 : pile vide

#Q3
    def afficher(self):
        if self.estVide():
            return
        pile2 = Pile()
        nb_elements=self.nb_elements()
        for i in range(nb_elements):
            elem = self.depiler()
            pile2.empiler(elem)
        for j in range(nb_elements):
            elem = pile2.depiler()
            print(elem,end = ' ')
            self.empiler(elem)




#Q4
def etendre(pile1, pile2):
    nb_elements=pile2.nb_elements()
    for i in range(nb_elements):
        pile1.empiler(pile2.depiler())



#Q5
def supprime_toutes_occurrences(pile,element):
    pile2=Pile()
    nb_elements = pile.nb_elements()
    for i in range(nb_elements):
        elem=pile.depiler()
        if elem!=element:
            pile2.empiler(elem)
    while not pile2.estVide():
        pile.empiler(pile2.depiler())




#Q6
```

```
def inserer(pile,a):
    '''
    CU : pile contient des entiers triés dans l ordre croissant
    le plus petit étant en bas de la pile

    la méthode insère l entier a à la bonne place
    '''
    pile2 = Pile()
    while not pile.estVide() and a < pile.retournerSommet() :
        pile2.empiler(pile.depiler())
    pile.empiler(a)
    while not pile2.estVide():
        pile.empiler(pile2.depiler())

#Q7
def trier(pile):
    pile2 = Pile()
    while not pile.estVide():
        pile2.empiler(pile.depiler())
    while not pile2.estVide():
        pile.inserer(pile2.depiler())
```

Corrigé exercice 2

```
class Carte:
#Q1
    def __init__(self,valeur):
        self.valeur = valeur
        self.TdB = self.calcul_TdB()
#Q2
    def calcul_TdB(self):
        TdB=0
        if self.valeur % 11 == 0:
            TdB +=5
        if self.valeur %10 ==0:
            TdB +=3
        if self.valeur % 10== 5:
            TdB +=2


        if TdB==0:
            TdB = 1
        return TdB
#Q3
    def est_superieur_a(self,autre):
        return self.valeur > autre.valeur

    def difference(self,autre):
        return abs(self.valeur - autre.valeur)
#Q4 et Q5
class Paquet:
    def __init__(self,L):
        self.contenu = L
```

```
    def afficher(self):
        for carte in self.contenu:
            print(carte.valeur , end= ' ')


    def ajouter_carte(self,carte):
        self.contenu.append(carte)

    def nombre_TdB(self):
        total = 0
        for carte in self.contenu:
            total += carte.TdB
        return total

#Q6     def distribuer(self, nbr):
        L=[Paquet([]) for i in range(nbr)]
        for i in range(10):
             for j in range(nbr):
                 carte = self.contenu.pop()
                 L[j].ajouter_carte(carte)
       return L




class Joueur():

    def __init__(self,nom,main):
        self.nom =nom
        self.main = main #main est le paquet de cartes du joueur
        self.cartes_ramassees =Paquet([])
        self.penalites = 0

#Q7
 J1 =Joueur(Joueur1 , L[0])

#Q8

#création des 104 cartes
jeu =[ Carte(i) for i in range(1,105)]
#mélange des cartes
shuffle(jeu)
jeu_initial =Paquet(jeu)

#distribution de 10 cartes aux deux joueurs
distri = jeu_initial.distribuer(2)
ordi = Joueur("ordi" , distri[0])
J1=Joueur("J1" , distri[1])
```