```
import numpy as np
masses={'H': 1.0,'He':4.0,'Li':6.9,'Be':9.0,'B':6,'C':14.0,
        'N':14.0,'O':15.9,'F':18.9,'Ne':20.1}

class Position:
    def __init__(self,a,b,c):
        self.tab=np.array([a,b,c])
    def __repr__(self):
        return('Position('+str(self.tab[0])+','+str(self.tab[1])+','+str(self.tab[2])+')')

    def __add__(self,autre):
        a=self.tab[0]+autre.tab[0]
        b=self.tab[1]+autre.tab[1]
        c=self.tab[2]+autre.tab[2]
        return Position(a,b,c)

    """version 2 de __add__
    def __add__(self,autre):
        T=self.tab+autre.tab
        return(Position(T[0],T[1],T[2]))"""


    def __getitem__(self,index):
        if index<len(self.tab):
            return self.tab[index]
        else:
            return "erreur"

    def Long(self):
        s=0
        for i in self.tab:
            s=s+i**2
        return np.sqrt(s)

    #Version2 de la methode Long
    def Long2(self):
        return np.sqrt(np.sum(self.tab**2))


class Atome:
    def __init__(self,a,b,c):
        self.sb=a
        self.pos=b
```

```
        self.np=c

    def masse(self):
        return masses[self.sb]

    def __repr__(self):
        return ('['+str(self.np)+'/'+str(self.masse())+']'+str(self.sb))

    def translation(self,P1):
        for i in range(len(self.pos.tab)):
                self.pos.tab[i]=self.pos.tab[i]+P1.tab[i]


    #version2 de la methode translation
    def translation2(self,P1):
        self.pos=self.pos.__add__(P1)
        #ou bien self.pos=self.pos+P1
        #ce qui est equivalent : self.pos+=P1
        #ou bien self.pos=P1.__add__(self.pos)



    def centreDeMasse(self):
        return self.pos

class Molecule:
    def __init__(self,L):
        self.listeAtomes=L

    def __repr__(self):
        #on va creer un dictionnaire: les clefs sont les atomes de la molecule et les valeurs
        #sont le nombre d'occurence de chaque atome dans la molecule
        D={}
        for atome in self.listeAtomes:
            if atome in D:
                D[atome]=D[atome]+1
            else:
                D[atome]=1

        #une liste qui va comporter seulement les cles c-a-d les atomes (sans redondance)
        #on recupere le nombre d'occurence a partir du dictionnaire D

        lesAtomes=list(D.keys())
```

```
    ch=''
    for i in lesAtomes:
        nb_occ=D[i]
        if nb_occ!=1:
            ch=ch+i.__repr__()+str(nb_occ)
        else:
            ch=ch+i.__repr__()
    return ch


def translate(self,P1):
    for i in self.listeAtomes:
        i.translate(P1)

def masse(self):
    m=0
    for i in self.listeAtomes:
        m=m+i.masse()
    return m

#Version 2 de la methode masse
def masse2(self):
    return np.sum([i.masse() for i in self.listeAtomes])



def masseDeCentre(self):
    S1 = self.masse()
    S2 = Position(0., 0., 0.)
    for i in self.listeAtomes:
        S2.tab=S2.tab+i.masse()*i.pos.tab

    P=S2.tab/S1
    return(Position(P[0],P[1],P[2]))



def ajouter(self,a):
    self.listeAtomes.append(a)

def __getitem__(self,index):
```

```
        self.listeAtomes[index]



###  Programme principal

hydrogene = Atome("H",Position(0., 5., 1.),1)
carbone = Atome("C",Position(1., 0., 1.),6)
oxygene = Atome("O",Position(0., 2., 1.),8)

dioxydeDeCarbone = Molecule([carbone,oxygene, oxygene])
butane=Molecule([carbone for i in range(4)]+[hydrogene for i in range(10)])
print(butane)
f=open('chimie.txt', 'w')
for m in [dioxydeDeCarbone,butane]:
    ligne = m.__repr__()+' Masse de Centre: ----> '+str(m.masseDeCentre())
    f.write(ligne +'\n')

f.close()
```