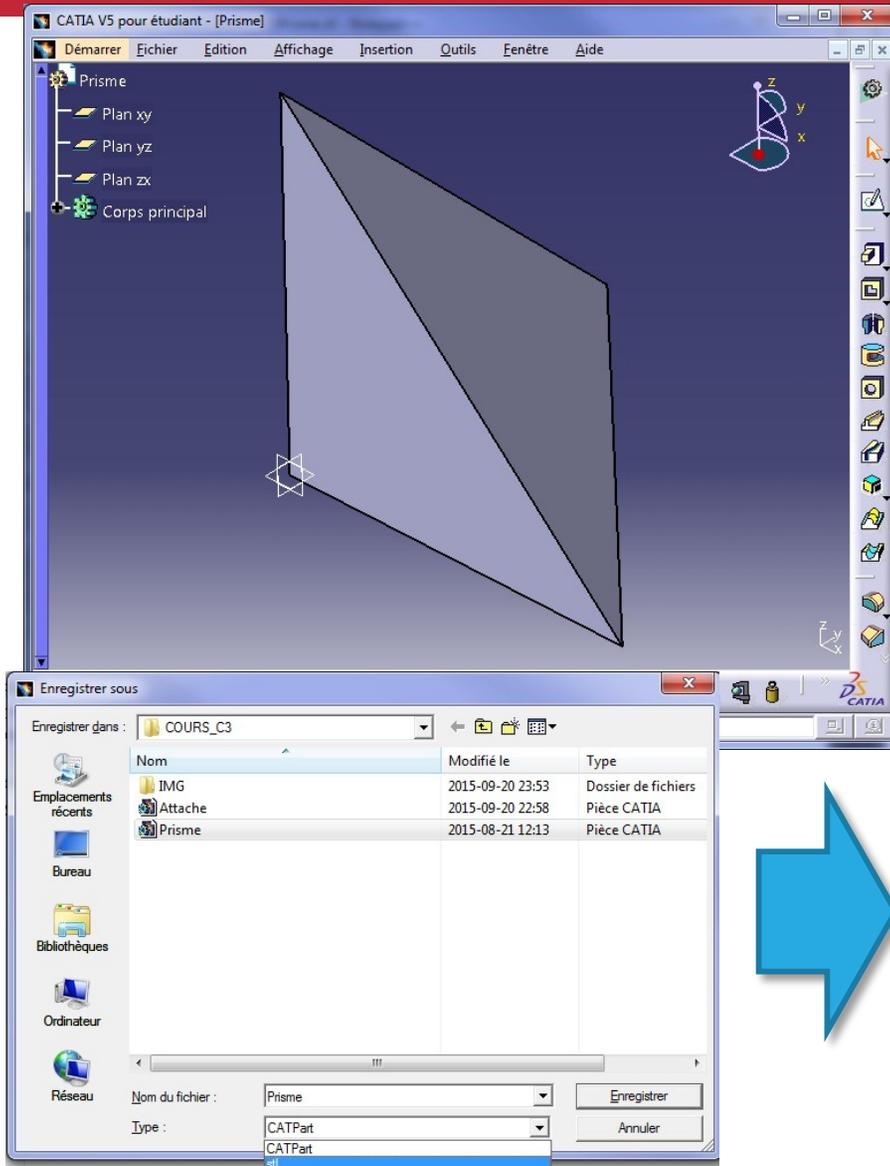


# COURS C3 : TRAITEMENT DE FICHER STL

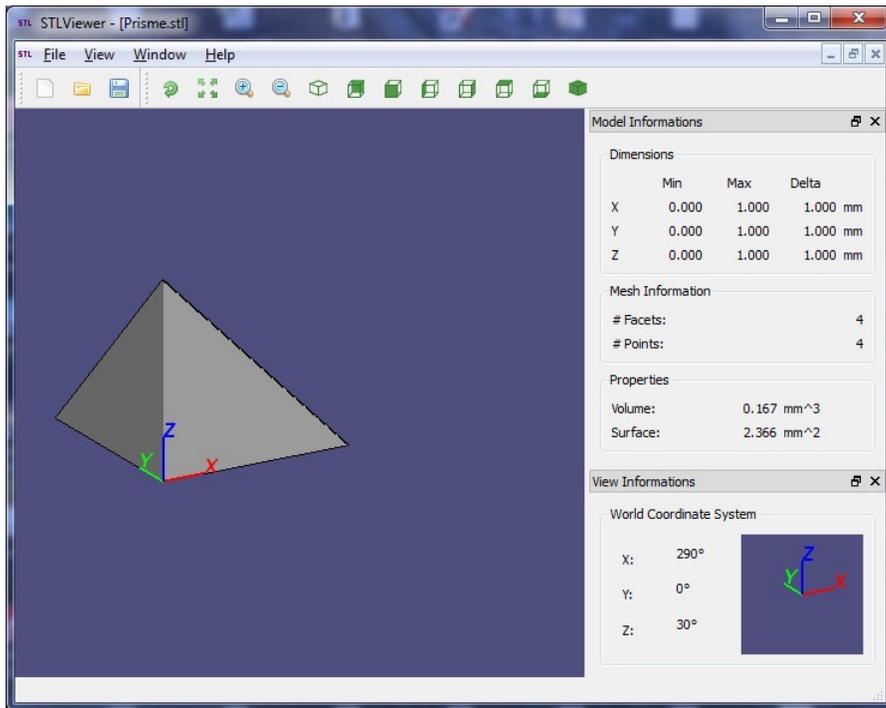
## Génération d'un fichier STL avec Catia



```
Prisme.stl |
1 solid CATIA STL
2   facet normal  0.000000e+000 -1.000000e+000  0.000000e+000
3     outer loop
4       vertex  0.000000e+000  0.000000e+000  1.000000e+000
5       vertex  0.000000e+000  0.000000e+000  0.000000e+000
6       vertex  1.000000e+000  0.000000e+000  0.000000e+000
7     endloop
8   endfacet
9   facet normal -1.000000e+000  0.000000e+000  0.000000e+000
10  outer loop
11    vertex  0.000000e+000  1.000000e+000  0.000000e+000
12    vertex  0.000000e+000  0.000000e+000  0.000000e+000
13    vertex  0.000000e+000  0.000000e+000  1.000000e+000
14  endloop
15 endfacet
16 facet normal  0.000000e+000  0.000000e+000 -1.000000e+000
17  outer loop
18    vertex  1.000000e+000  0.000000e+000  0.000000e+000
19    vertex  0.000000e+000  0.000000e+000  0.000000e+000
20    vertex  0.000000e+000  1.000000e+000  0.000000e+000
21  endloop
22 endfacet
23 facet normal  5.773503e-001  5.773503e-001  5.773503e-001
24  outer loop
25    vertex  0.000000e+000  0.000000e+000  1.000000e+000
26    vertex  1.000000e+000  0.000000e+000  0.000000e+000
27    vertex  0.000000e+000  1.000000e+000  0.000000e+000
28  endloop
29 endfacet
30 endsolid CATIA STL
```

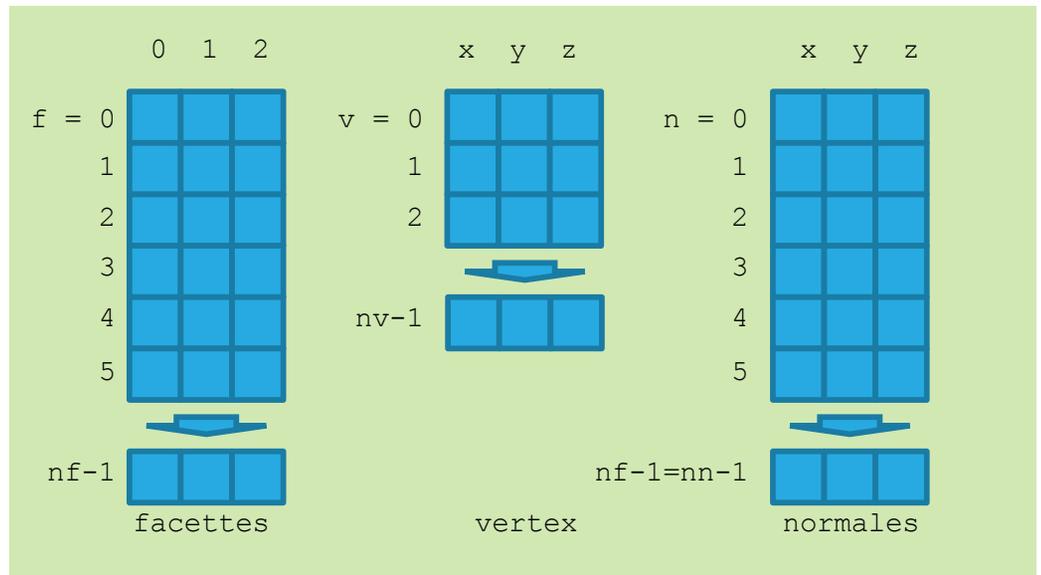
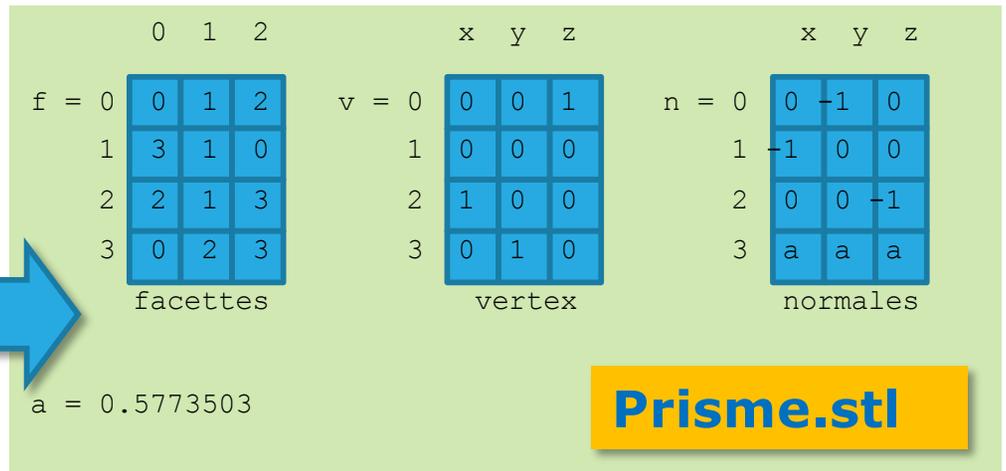
# COURS C3 : TRAITEMENT DE FICHER STL

## Lecture d'un fichier STL



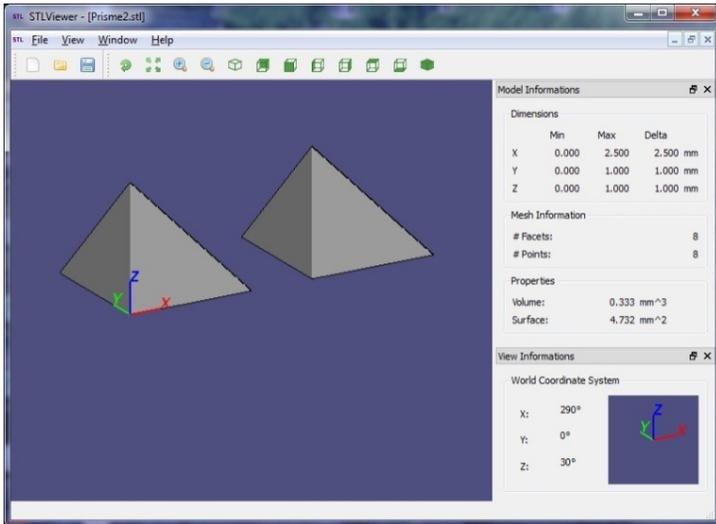
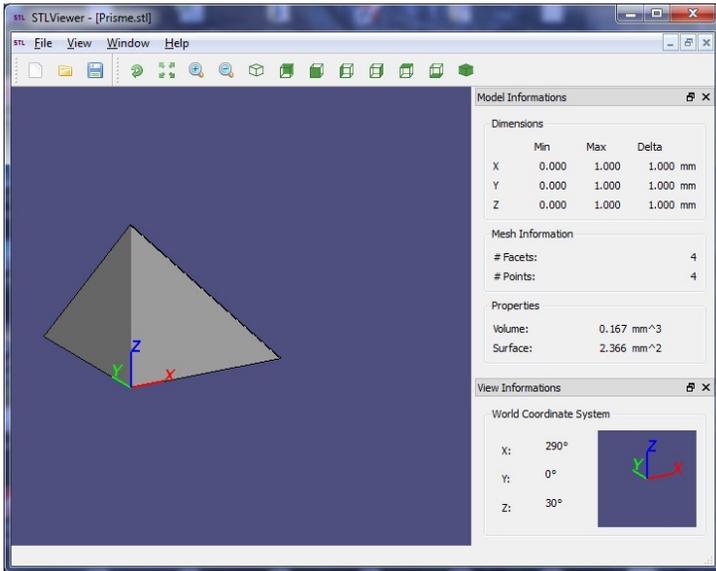
**En général**

`f , v , n = LireSTL(nom_fichier_in)`



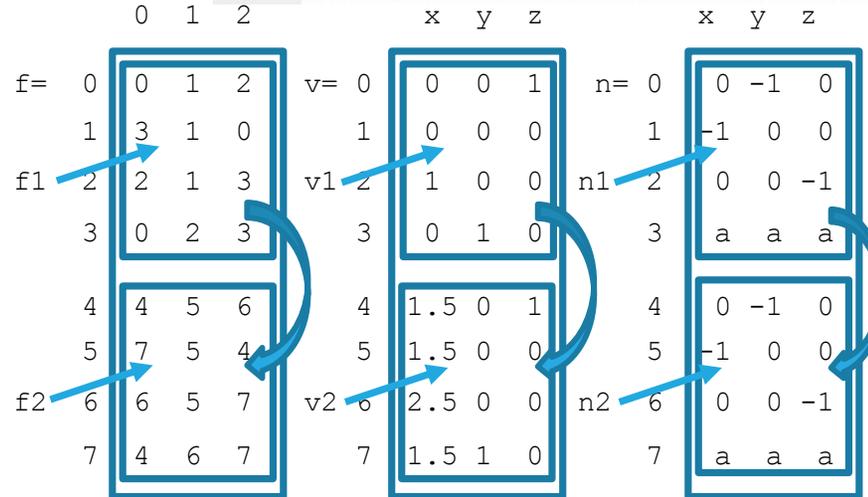
# COURS C3 : TRAITEMENT DE FICHER STL

## Ajout d'une copie à x=1.5



```

1 #Exemple C3_E1
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from MEC1315_STL import * #On importe tous les fonctions
5 nom_fichier_in="Prisme.stl"
6 f1,v1,n1=LireSTL(nom_fichier_in)
7 #=====
8 nv1=len(v1) #nombre de points dans v1
9 f2=f1+nv1 #facettes f1+nv1
10 #=====Translation en x=====
11 # v2=v1+[1.5,0,0]
12 #====Méthode 2 et 3 non fonctionnel=====
13 v2=np.array(v1) #fonctionne
14 #v2=v1 #ne fonctionne pas
15 v2[:,0]=v2[:,0]+1.5
16 n2=n1
17 #=====Fusion=====
18 f=np.vstack((f1,f2))
19 v=np.vstack((v1,v2))
20 n=np.vstack((n1,n2))
21 #=====Écriture=====
22 nom_fichier_out="Prisme2.stl"
23 EcrireSTLASCII(nom_fichier_out,f,v,n)
24
25 #=====Traçage du graphique=====
26 fig = plt.figure()
27 ax = fig.add_subplot(1, 1, 1, projection='3d')
28 ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
    
```



$f2=f1+4$

$v2=v1+\text{déplacement}$

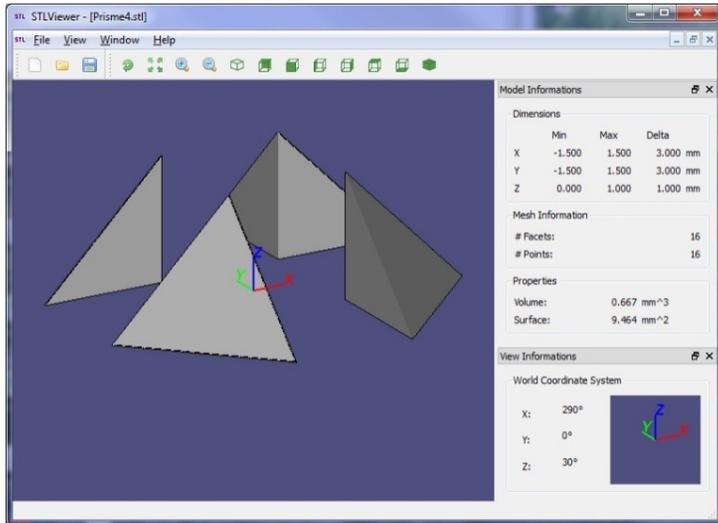
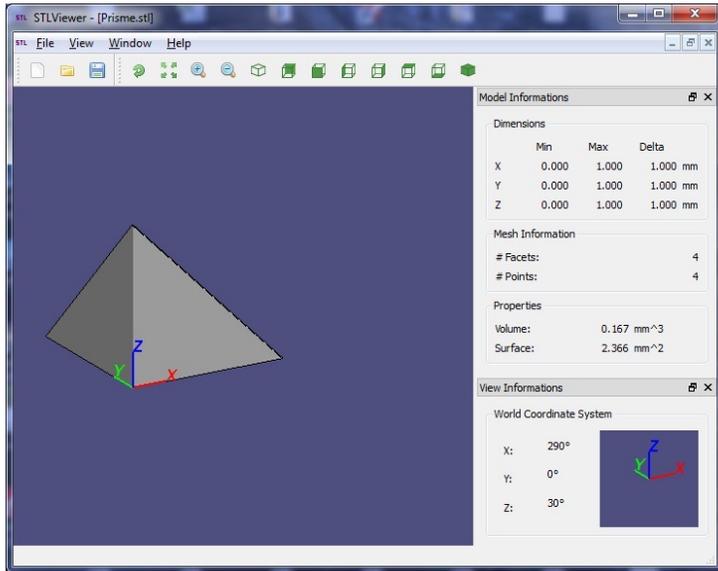
$n2=n1$  (pas de rotation)

Le génie en première classe



# COURS C3 : TRAITEMENT DE FICHER STL

## Translation à (0.5,0.5,0), puis 4 rotations en z de $\pi/2$



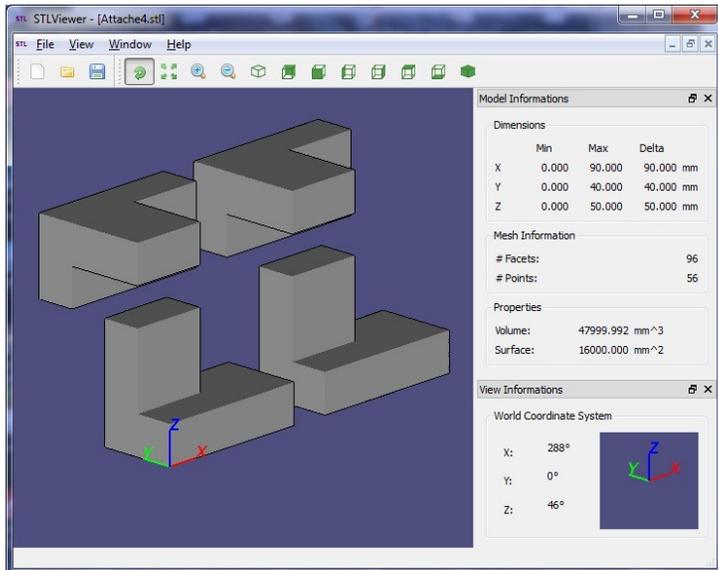
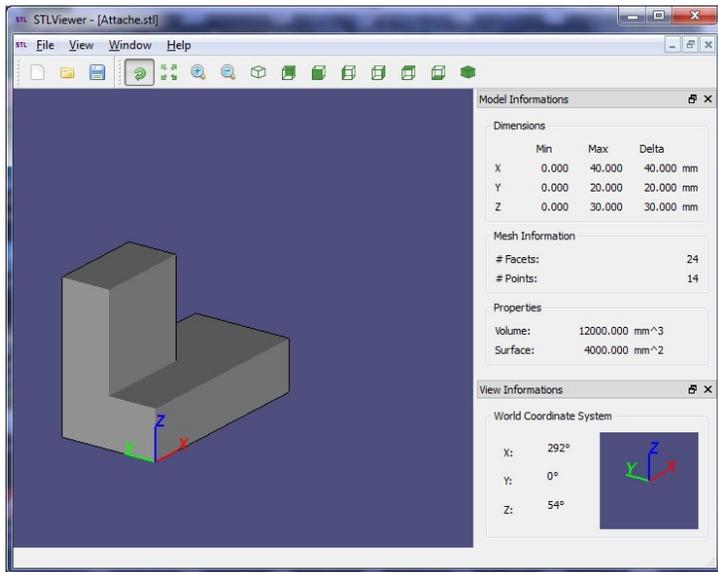
```
1 #Exemple C3_E1
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from MEC1315_STL import * #On importe tous les fonctions
5 nom_fichier_in="Prisme.stl"
6 f1,v1,n1=LireSTL(nom_fichier_in)
7 #=====
8 nv1=len(v1)
9 Rz=np.pi/2
10 v1[:,0]=v1[:,0]+0.5 #Translation en x
11 v1[:,1]=v1[:,1]+0.5 #Translation en y
12 f2, v2, n2= f1+nv1, v1.dot(R), n1.dot(R) #Copie de 1 dans 2 avec rotation
13 f3, v3, n3=f2+nv1, v2.dot(R), n2.dot(R) #Copie de 2 dans 3 avec rotation
14 f4, v4, n4=f3+nv1, v3.dot(R), n3.dot(R) #Copie de 3 dans 4 avec rotation
15
16 #=====Fusion=====
17 f=np.vstack((f1,f2,f3,f4))
18 v=np.vstack((v1,v2,v3,v4))
19 n=np.vstack((n1,n2,n3,n4))
20 #=====Écriture=====
21 nom_fichier_out="Prisme_rotation.stl"
22 EcrireSTLASCII(nom_fichier_out,f,v,n)
23
24 #=====Traçage du graphique=====
25 fig = plt.figure()
26 ax = fig.add_subplot(1, 1, 1, projection='3d')
27 ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
28
```

```
128 def Rz(angle):
129     Rz=np.array([[np.cos(angle), np.sin(angle), 0],
130                 [-np.sin(angle), np.cos(angle), 0],
131                 [0, 0, 1]])
132     return Rz
133
```



# COURS C3 : TRAITEMENT DE FICHER STL

## Copie en x, puis rotation et translation (0,40,30) du groupe



```
1 #Exemple C3_E1
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from MEC1315_STL import * #On importe tous les fonctions
5 nom_fichier_in="Bloc.stl"
6 f1,v1,n1=LireSTL(nom_fichier_in)
7 #=====
8 nv1=len(v1)
9 f2=f1+nv1 #Copie de 1 dans 2
10 v2=np.array(v1)
11 v2[:,0]=v2[:,0]+50 #Translation en x de 50mm
12 n2=n1
13 #=====Fusion intermédiaire=====
14 f , v , n =np.vstack((f1,f2)) , np.vstack((v1,v2)) , np.vstack((n1,n2))
15 nv=len(v)
16 R=Rx(np.pi/2) #Rotation autour de x de pi/2
17 f2 , v2 , n2 = f+nv , v.dot(R) , n.dot(R) #Copie de 1 dans 2
18 v2[:,1]=v2[:,1]+40 #Translation de 40 unités en y
19 v2[:,2]=v2[:,2]+30 #Translation de 30 unités en z
20 #=====Fusion final=====
21 f=np.vstack((f,f2))
22 v=np.vstack((v,v2))
23 n=np.vstack((n,n2))
24 #=====Écriture=====
25 nom_fichier_out="Bloc_modifie.stl"
26 EcrireSTLASCII(nom_fichier_out,f,v,n)
27 #=====Tracage du graphique=====
28 fig = plt.figure()
29 ax = fig.add_subplot(1, 1, 1, projection='3d')
30 ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
```

```
def Rx(angle):
    Rx=np.array([[1, 0, 0],
                 [0, np.cos(angle), np.sin(angle)],
                 [0, -np.sin(angle), np.cos(angle)]]
    return Rx

def Ry(angle):
    Ry=np.array([[np.cos(angle), 0, -np.sin(angle)],
                 [0, 1, 0],
                 [np.sin(angle), 0, np.cos(angle)]] )
    return Ry
```



# COURS C3 : TRAITEMENT DE FICHER STL

## Lecture d'un fichier STL

Fichier de fonction MEC1315\_STL fournit, doit être au même répertoire que le scripte de python ainsi que les fichiers STL

```
141 def EcrireSTLASCII(output_file_name,face_fvn,vertex_fvn,normal_fvn):
142     face_fvn=face_fvn.astype(int)
143     f=open(output_file_name,"w")
144     f.write("solid python")
145     for j in range(len(face_fvn)):
146         t1="facet normal"
147         t2="outer loop"
148         t3="endloop"
149         t4="endfacet"
150         f.write("\n %s %7.6e %7.6e %7.6e" % (t1, normal_fvn[j,0],normal_fvn[j,
151         f.write("\n %s" % t2) #Écrire outerloop
152         f.write("\n     vertex %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,0],0],
153         f.write("\n     vertex %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,1],0],
154         f.write("\n     vertex %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,2],0],
155         f.write("\n %s" % t3) #ecrire endloop
156         f.write("\n %s" % t4) #ecrire endfacet
157     f.write("\n%s" % "Endsolid Python")
158     f.close()
159
```

Fonction Écriture, formatage d'un fichier STL

```
116 def Rx(angle):
117     Rx=np.array([[1, 0, 0],
118                 [0, np.cos(angle), np.sin(angle)],
119                 [0, -np.sin(angle), np.cos(angle)]])
120     return Rx
121
122 def Ry(angle):
123     Ry=np.array([[np.cos(angle), 0, -np.sin(angle)],
124                 [0, 1, 0],
125                 [np.sin(angle), 0, np.cos(angle)] ] )
126     return Ry
```

Fonction Rotation

```
160 def CalculNormal(face_fvn,vertex_fvn):
161     face_fvn=face_fvn.astype(int)
162     normal_fvn=np.empty(np.shape(face_fvn))
163     for i in range(len(face_fvn)):
164         # print(f[i])
165         vecteurA=vertex_fvn[face_fvn[i],0]
166         vecteurB=vertex_fvn[face_fvn[i],1]
167         vecteurC=vertex_fvn[face_fvn[i],2]
168         vecteur_normal=np.cross((vecteurB-vecteurA), (vecteurC-vecteurA))
169         normal_fvn[i]=vecteur_normal/np.linalg.norm(vecteur_normal)
170     return normal_fvn
171
```

Fonction qui calcul la normale, utilisé pour les changements d'échelle par affinité vectorielle



# COURS C3 : TRAITEMENT DE FICHER STL

## Écriture d'un fichier STL

```
110 def LireSTL(nom_fichier):
111     vertex_fvn, face_fvn =ReadSTL(nom_fichier)
112     normal_fvn=GetNormals(vertex_fvn, face_fvn)
113     return face_fvn, vertex_fvn, normal_fvn

53 def ReadSTL(filename):
54     """ Returns numpy arrays for vertices and facet indexing """
55     def GetListFromASCII(filename):
56         """ Returns vertex listing from ASCII STL file """
57         outputList=[]
58
59         with open(filename,'r') as f:
60             lines=[line.split() for line in f.readlines()]
61         for line in lines:
62             if line[0] == 'vertex':
63                 outputList.append(tuple([float(x) for x in line[1:])))
64         return outputList
65
66     def GetListFromBinary(filename):
67         """ Returns vertex listing from binary STL file """
68         outputList=[]
69         with open(filename,'rb') as f:
70             f.seek(80) # skip header
71             nFacets=struct.unpack('I',f.read(4))[0] # number of facets in piece
72
73             for i in range(nFacets):
74                 f.seek(12,1) # skip normal
75                 outputList.append(struct.unpack('fff',f.read(12))) # append each
76                 outputList.append(struct.unpack('fff',f.read(12)))
77                 outputList.append(struct.unpack('fff',f.read(12)))
78                 f.seek(2,1) # skip attribute
79         return outputList
80
81     if IsBinarySTL(filename):
82         vertexList = GetListFromBinary(filename)
83     else:
84         vertexList = GetListFromASCII(filename)
85
86     coords, tempindx = Unique(vertexList)
87
88     indx = list()
89     templist = list()
90     for i in range(len(tempindx)):
91         if (i > 0 ) and not (i % 3):
92             indx.append(templist)
93             templist = list()
94         templist.append(tempindx[i])
95     indx.append(templist)
96
97     return np.array(coords), np.array(indxs)
```

Fonction LireSTL adapté pour le cours  
MEC1315

Fonction ReadSTL compatible pour fichier  
binaire et ascii. Code libre et tiré en ligne

