作业 19：Dragon 绘制

要求：

根据给出的 dragon.tar.zip 文件，按文件数据绘制图片

提示：

1. 使用 tarfile 包进行解压相关操作
2. 打开后传入文件路径，可用代码 ：
   os.path.join()
3. 使用 mayavi.mlab 中的函数读取渲染 dragon ply 文件：
   mlab.pipeline.surface(mlab.pipeline.open(filename))
   mlab.show()

作业 20：地形可视化

要求：

读取示例文件：N36W113.hgt.zip。使用 mayavi.mlab 进行可视化

提示：

1. 使用 zipfile 进行读取：
   zipfile.ZipFile(zipfilename).rea(filename)
2. 使用 mlab 渲染地形 hgt 的数据
   mlab.figure(size=(400, 320), bgcolor=(0.16, 0.28, 0.46))
   mlab.surf(data, colormap='gist_earth', warp_scale=0.2, vmin=1200, vmax=1610)
3. 创建交互式可视化窗口
   mlab.view(-5.9, 83, 570, [5.3, 20, 238])
   mlab.show()

作业 21：建立简单的 mayavi 窗口

要求：

自己设立函数，使用 mayavi 进行演示

提示：

1. 建立 mayavi 窗口代码示例：

```
from numpy import sqrt, sin, mgrid
from traits.api import HasTraits, Instance
from traitsui.api import View, Item
from tvtk.pyface.scene_editor import SceneEditor
from mayavi.tools.mlab_scene_model import MlabSceneModel
from mayavi.core.ui.mayavi_scene import MayaviScene


class ActorViewer(HasTraits):
    # 场景模型
    scene = Instance(MlabSceneModel, ())   # 建立场景实例
    # 建立视图
    view = View(Item(name='scene',   # 提供 Mayavi 视图窗口
                     editor=SceneEditor(scene_class=MayaviScene),
                     show_label=False,
                     resizable=True,
                     height=500,
                     width=500),
                resizable=True)

    def __init__(self, **traits):
        HasTraits.__init__(self, **traits)
        self.generate_data()
```

2. 在该类中自己定义一个函数来生成三维数据：

```
    def generate_data(self):
        ...
        self.scene.mlab.surf(x, y, z, colormap='cool')
```

3. 定义完类后进行调用：

```
a = ActorViewer()
a.configure_traits()
```

作业 22：基于交互控制的 Mayavi 窗口

要求：

按照提示自行编写代码实现

提示：

1. 定义从 HasTraits 继承的类

```
# 1.定义从HasTraits继承类
class MyModel(HasTraits):
    # 1.1定义窗口中的变量
    n_meridional
    n_longitudinal
    scene
    # 1.2更新视图绘制
    @on_trait_change()
    def update_plot(self):
        ... ...
    # 1.3建立视图布局
    view = View()
```

1.1 定义窗口中的变量

```
from traits.api import HasTraits, Range, Instance
from mayavi.core.ui.api import MlabSceneModel

class MyModel(HasTraits):
    # 1.1定义窗口中的变量
    n_meridional    = Range(0, 30, 6) #滑动条控件
    n_longitudinal  = Range(0, 30, 11) #滑动条控件
    scene = Instance(MlabSceneModel, ())# 场景模型实例
```

1.2 定义监听函数、更新视图绘制

```
from traits.api import on_trait_change

@on_trait_change('n_meridional,n_longitudinal,scene.activated')
def update_plot(self):
    ... ...#生成数据，并更新视图

    x, y, z, t = curve(self.n_meridional, self.n_longitudinal)
    if self.plot is None:
        self.plot = self.scene.mlab.plot3d(x, y, z, t,
                    tube_radius=0.025, colormap='Spectral')
    else:
        self.plot.mlab_source.set(x=x, y=y, z=z, scalars=t)
```

注意 curve 函数可自己生成

1.3 定义视图的布局

```
from traitsui.api import View, Item, Group
from mayavi.core.ui.api import MayaviScene, SceneEditor

# 1.3建立视图布局
view = View(
    Item('scene', editor=SceneEditor(scene_class=MayaviScene),
         height=250, width=300, show_label=False),
    Group('_', 'n_meridional', 'n_longitudinal'),
    resizable=True
    )
```

2. 调用 configure_traits()：

```
model = MyModel()
model.configure_traits()
```

作业 23：Spatial-ConvexHull 三维凸包可视化

要求：

绘制 Spatial-ConvexHull 三维凸包

提示：

1. 三维空间中的凸包是一个凸多面体，每个面都是一个三角形。
2. 在 scipy 中可以直接用 spatial.ConvexHull(points3d)生成凸包数组。其中 points3d 是三维矩阵，可自行随机生成。
3. 定义生成绘制凸包的函数

```
def convexhull(ch3d):
    #1 定义凸多面体 tvtk 的 Polydata() 对象
    poly = tvtk.PolyData()
    poly.points = ch3d.points
    poly.polys = ch3d.simplices
    #2 定义凸多面体顶点的小球
    sphere = tvtk.SphereSource(radius = 0.02)
    points3d = tvtk.Glyph3D()
    points3d.set_source_connection(sphere.output_port)
    points3d.set_input_data(poly)
    #3 绘制凸多面体的面，设置半透明度
    m1 = tvtk.PolyDataMapper()
    m1.set_input_data(poly)
    a1 = tvtk.Actor(mapper=m1)
    a1.property.opacity = 0.3
    #4 绘制凸多面体的边，设置为红色
```

```
    m2 = tvtk.PolyDataMapper()
    m2.set_input_data(poly)
    a2 = tvtk.Actor(mapper=m2)
    a2.property.representation = 'wireframe'
    a2.property.line_width = 2.0
    a2.property.color = (1.0,0,0)
    #5 绘制凸多面体的顶点，设置为绿色
    m3 = tvtk.PolyDataMapper(input_connection=points3d.output_port)
    a3 = tvtk.Actor(mapper = m3)
    a3.property.color = (0.0,1.0,0.0)
    return [a1,a2,a3]
```

4. 定义绘制场景用的函数

```
def ivtk_scene(actors):
    from tvtk.tools import ivtk
    win = ivtk.IVTKWithCrustAndBrowser()      #创建 crust 窗口
    win.open()
    win.scene.add_actor(actors)
    dialog = win.control.centralWidget().widget(0).widget(0)   #窗口错误修正
    from pyface.qt import QtCore
    dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
    dialog.show()
    return win
```

作业 24：K-means 聚类

要求：

通过聚类，了解 1999 年各个省份的消费水平在国内的情况。使用文件 city.txt

提示：

1. 建立工程，导入 sklearn 相关包

```
import numpy as np
from sklearn.cluster import KMeans
```

2. 加载数据，创建 K-means 算法实例，并进行训练，获得标签：
3. 加载 city.txt 中的数据，并返回数据和城市名。
4. 将城市按 label 分成设定的簇
5. 将每个簇的城市输出
6. 将每个簇的平均花费输出

作业 25：DBSCAN 密度聚类

要求：

通过 DBSCAN 聚类，分析学生上网时间和上网时长的模式。读取文件：‘学生月上网时间分布-TestData.txt’

提示：

1. 读取数据

```
mac2id=dict()
onlinetimes=[]
f=open('学生月上网时间分布-TestData.txt',encoding='utf-8')                    #加载数据
for line in f:
    mac=line.split(',')[2]
    onlinetime=int(line.split(',')[6])
    starttime=int(line.split(',')[4].split(' ')[1].split(':')[0])          #读取每条数据的 MAC 地址，开始上网时间，上网时长
    if mac not in mac2id:                                                  #mac2id 是一个字典：
        mac2id[mac]=len(onlinetimes)                                       #key 是 mac 地址
        onlinetimes.append((starttime,onlinetime))                         #value 是对应 mac 地址的
                                                                           #上网时长以及开始上网时
                                                                           #间

    else:
        onlinetimes[mac2id[mac]]=[(starttime,onlinetime)]
```

2. 调用 DBSCAN 方法进行训练 ，labels 为每个数据的簇标签
3. 打印数据被记上的标签，计算标签为-1，即噪声数据的比例。
4. 计算簇的个数并打印，评价聚类效果
5. 打印各簇标号以及各簇内数据