```
import main
import matplotlib.pyplot as plt

mainNode = dict(boxstyle='sawtooth', fc='0.8')
secondNode = dict(boxstyle='round4', fc='0.8')
arrow_args = dict(arrowstyle='<-')


def drawNode(nodeTxt, centerPt, parentPt, nodeType):
    showDic.ax1.annotate(nodeTxt, xy=parentPt, xycoords='axes fraction',
                         xytext=centerPt, textcoords='axes fraction',
                         va='center', ha='center', bbox=nodeType, arrowprops=arrow_args)


def getLeafs(inputTree):
    '''
    子节点（叶节点）数量
    '''
    numLeafs = 0
    firstStr = list(inputTree.keys())[0]
    secondDict = inputTree[firstStr]
    for key in secondDict.keys():
        if type(secondDict[key]).__name__ == 'dict':
            numLeafs += getLeafs(secondDict[key])
        else:
            numLeafs += 1
    return numLeafs


def getDepth(inputTree):
    '''
    决策树最大深度
    '''
    maxDepth = 0
    firstStr = list(inputTree.keys())[0]
    secondDict = inputTree[firstStr]
    for key in secondDict.keys():
        if type(secondDict[key]).__name__ == 'dict':
            thisDepth = 1 + getDepth(secondDict[key])
        else:
            thisDepth = 1
        if thisDepth > maxDepth:
            maxDepth = thisDepth
    return maxDepth
```

```
def plotMidText(cntrPt, parentPt, txtStr):
    '''
    决策树箭头上的字
    '''
    xMid = (parentPt[0] - cntrPt[0]) / 2.0 + cntrPt[0]
    yMid = (parentPt[1] - cntrPt[1]) / 2.0 + cntrPt[1]
    showDic.ax1.text(xMid, yMid, txtStr)


def showTree(inputTree, parentPt, nodeTxt):
    numLeafs = getLeafs(inputTree)
    depth = getDepth(inputTree)
    firstStr = list(inputTree.keys())[0]
    cntrPt = (showTree.xOff + (1.0 + float(numLeafs))/2.0/showTree.totalW,
              showTree.yOff)
    plotMidText(cntrPt, parentPt, nodeTxt)
    drawNode(firstStr, cntrPt, parentPt, mainNode)
    secondDict = inputTree[firstStr]
    showTree.yOff = showTree.yOff - 1.0/showTree.totalD
    for key in secondDict.keys():
        if type(secondDict[key]).__name__ == 'dict':
            showTree(secondDict[key], cntrPt, str(key))
        else:
            showTree.xOff = showTree.xOff + 1.0/showTree.totalW
            drawNode(secondDict[key], (showTree.xOff,
                                       showTree.yOff), cntrPt, secondNode)
            plotMidText((showTree.xOff, showTree.yOff), cntrPt, str(key))
    showTree.yOff = showTree.yOff+1.0/showTree.totalD


def showDic():
    fig = plt.figure(1, facecolor='white')
    fig.clf()
    showDic.ax1 = plt.subplot(111, frameon=False)
    drawNode('node', (0.5, 0.1), (0.1, 0.5), mainNode)
    drawNode('leaf', (0.8, 0.1), (0.3, 0.8), secondNode)
    plt.show()


def showDic(inTree):
    '''
    将嵌套字典展示为决策树
    '''
    fig = plt.figure(1, facecolor='white')
    fig.clf()
    axprops = dict(xticks=[], yticks=[])
    showDic.ax1 = plt.subplot(111, frameon=False, **axprops)
    showTree.totalW = float(getLeafs(inTree))
    showTree.totalD = float(getDepth(inTree))
    showTree.xOff = -0.5/showTree.totalW
    showTree.yOff = 1.0
    showTree(inTree, (0.5, 1.0), '')
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.savefig('output.pdf')
    plt.show()


showDic(main.main())
```