作业 11：自动轨迹绘制

要求：

读取 data.txt 文件，按文件数据绘制图片

提示：

1. 使用 turtle
2. 示例文件中：

300,0,144,1,0,0 依次代表行进距离、转向判断、转向角度，后三位代表 RGB 三个通道颜色

3. turtle.pencolor(data[i][3], data[i][4], data[i][5]) #使用后三位

作业 12：词云处理

要求：

1. 读取示例文件：新时代中国特色社会主义.txt。使用 wordcloud 库将文本变成词云图片。最大词数为 15。
2. 读取示例图片：fivestart.png。将其作为模板生成词云图片。

提示：

1. 生成词云前须用 jieba 库将词语切分开。
2. 词云使用代码：

```
w=wordcloud.WordCloud(font_path, mask, width, height, background_color)
w.generate(list)
w.to_file(name)
```

作业 13：模拟比赛

要求：

自己设计代码，输入球员 A 和球员 B 的胜率，比赛次数。进行比赛模拟，输出球员 A 和 B 的获胜次数和比值。

提示：

可以分多个函数分别实现获取球员胜率和比赛次数、比赛模拟、获胜次数计数和比值。

作业 14：第三方库安装脚本

要求：

编写代码，自动安装 numpy, matplotlib, pillow, sklearn, requests, jieba, beautifulsoup4 函数包。并在每次安装后输出安装结果。

提示：

1. 可将想要安装的包写在一个 List 里。依次安装。
2. os.system（命令）函数可在命令端调用命令。
3. 使用 try:
   except:
   函数来输出成功与否。

作业 15：玫瑰花绘制

按代码输出结果（代码附在文档后面）

作业 16：图像转手绘效果

要求：

将提供的图片转为手绘效果

提示：

利用像素之间的梯度值和虚拟深度值对图像进行重构

根据灰度变化来模拟人类视觉的远近程度

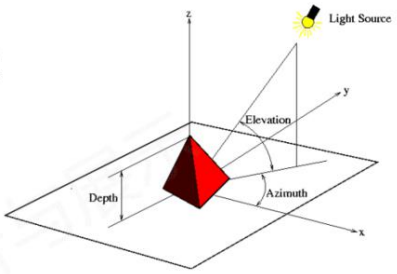
```
depth = 10.
grad = np.gradient(a)
grad_x, grad_y = grad
grad_x = grad_x*depth/100.
grad_y = grad_y*depth/100.
```

预设深度值为10　取值范围0-100

提取x和y方向的梯度值

根据深度调整x和y方向的梯度值

光源效果



np.cos(vec_el)为单位光线在地平面上的投影长度

```
vec_el = np.pi/2.2
vec_az = np.pi/4.
dx = np.cos(vec_el)*np.cos(vec_az)
dy = np.cos(vec_el)*np.sin(vec_az)
dz = np.sin(vec_el)
```

dx，dy，dz是光源对x/y/z三方向的影响程度

梯度归一化

构造x和y轴梯度的三维归一化单位坐标系

```
A = np.sqrt(grad_x**2 + grad_y**2 + 1.)
uni_x = grad_x/A
uni_y = grad_y/A
uni_z = 1./A
b = 255*(dx*uni_x + dy*uni_y + dz*uni_z)
```

梯度与光源相互作用，将梯度转化为灰度

图像生成

为避免数据越界，将生成的灰度值裁剪至0-255区间

```
b = b.clip(0,255)
im = Image.fromarray(b.astype('uint8'))
im.save('./beijingHD.jpg')
```

生成图像

作业 17：标量数据绘制

要求：

解压 1.3.15 plot3d_data.rar 文件，将里面的数据利用 tvtk 进行绘制

提示：

```
# 读入数据
plot3d = tvtk.MultiBlockPLOT3DReader(
        xyz_file_name="combxyz.bin",#网格文件
        q_file_name="combq.bin",#空气动力学结果文件
        scalar_function_number=100, vector_function_number=200
    )
plot3d.update()#让plot3D计算其输出数据
grid = plot3d.output.get_block(0)#获取读入的数据集对象
```

使用 tvtk 创建等值面对象，设置网格，并指定轮廓书数和数据范围。

```
con = tvtk.ContourFilter()#创建等值面对象
con.set_input_data(grid)
con.generate_values(10, grid.point_data.scalars.range)#指定轮廓数和数据范围
#设定映射器的变量范围属性
```

绘制数据：

```
m = tvtk.PolyDataMapper(scalar_range = grid.point_data.scalars.range,
                        input_connection=con.output_port)
a = tvtk.Actor(mapper = m)
a.property.opacity = 0.5#设定透明度为0.5
#窗口绘制
win = ivtk_scene(a)
```

作业 18：矢量数据可视化

要求：

使用作业 17 的数据，但是对数据进行随机选取，每 50 个点选择一个点。
在各点上放置箭头，箭头方向长度和颜色都有对应的矢量和标量数据决定。

提示：

tvtk.MaskPoints () 降采样

```
#对数据集中的数据进行随机选取，每50个点选择一个点
mask = tvtk.MaskPoints(random_mode=True, on_ratio=50)
mask.set_input_data(grid)
#创建表示箭头的PolyData数据集
glyph_source = tvtk.ArrowSource()
#在Mask采样后的PolyData数据集每个点上放置一个箭头
#箭头的方向、长度和颜色由于点对应的矢量和标量数据决定
glyph = tvtk.Glyph3D(input_connection=mask.output_port,
                     scale_factor=4)
glyph.set_source_connection(glyph_source.output_port)
```

tvtk.Glyph3D()　符号化技术

```
#对数据集中的数据进行随机选取，每50个点选择一个点
mask = tvtk.MaskPoints(random_mode=True, on_ratio=50)
mask.set_input_data(grid)
#创建表示箭头的PolyData数据集
glyph_source = tvtk.ArrowSource()
#在Mask采样后的PolyData数据集每个点上放置一个箭头
#箭头的方向、长度和颜色由于点对应的矢量和标量数据决定
glyph = tvtk.Glyph3D(input_connection=mask.output_port,
                     scale_factor=4)
glyph.set_source_connection(glyph_source.output_port)
```

作业 19：空间轮廓线可视化

要求：

对 17 中的数据进行处理，使只显示轮廓线

提示：

计算轮廓线

```
outline = tvtk.StructuredGridOutlineFilter()#计算表示外边框的PolyData对象
configure_input(outline, grid)#调用tvtk.common.configure_input()
m = tvtk.PolyDataMapper(input_connection=outline.output_port)
a = tvtk.Actor(mapper=m)
a.property.color = 0.3, 0.3, 0.3
```

作业 15 代码：

```
#RoseDraw.py
import turtle as t
#  定义一个曲线绘制函数
def DegreeCurve(n, r, d=1):
    for i in range(n):
        t.left(d)
        t.circle(r, abs(d))
#  初始位置设定
s = 0.2 # size
t.setup(450*5*s, 750*5*s)
t.pencolor("black")
t.fillcolor("red")
t.speed(100)
t.penup()
```

```
t.goto(0, 900*s)
t.pendown()
# 绘制花朵形状
t.begin_fill()
t.circle(200*s,30)
DegreeCurve(60, 50*s)
t.circle(200*s,30)
DegreeCurve(4, 100*s)
t.circle(200*s,50)
DegreeCurve(50, 50*s)
t.circle(350*s,65)
DegreeCurve(40, 70*s)
t.circle(150*s,50)
DegreeCurve(20, 50*s, -1)
t.circle(400*s,60)
DegreeCurve(18, 50*s)
t.fd(250*s)
t.right(150)
t.circle(-500*s,12)
t.left(140)
t.circle(550*s,110)
t.left(27)
t.circle(650*s,100)
t.left(130)
t.circle(-300*s,20)
t.right(123)
t.circle(220*s,57)
t.end_fill()
# 绘制花枝形状
t.left(120)
t.fd(280*s)
t.left(115)
t.circle(300*s,33)
t.left(180)
t.circle(-300*s,33)
DegreeCurve(70, 225*s, -1)
t.circle(350*s,104)
t.left(90)
t.circle(200*s,105)
t.circle(-500*s,63)
t.penup()
t.goto(170*s,-30*s)
t.pendown()
t.left(160)
```

```
DegreeCurve(20, 2500*s)
DegreeCurve(220, 250*s, -1)
# 绘制一个绿色叶子
t.fillcolor('green')
t.penup()
t.goto(670*s,-180*s)
t.pendown()
t.right(140)
t.begin_fill()
t.circle(300*s,120)
t.left(60)
t.circle(300*s,120)
t.end_fill()
t.penup()
t.goto(180*s,-550*s)
t.pendown()
t.right(85)
t.circle(600*s,40)
# 绘制另一个绿色叶子
t.penup()
t.goto(-150*s,-1000*s)
t.pendown()
t.begin_fill()
t.rt(120)
t.circle(300*s,115)
t.left(75)
t.circle(300*s,100)
t.end_fill()
t.penup()
t.goto(430*s,-1070*s)
t.pendown()
t.right(30)
t.circle(-600*s,35)
t.done()
```