第一次上机作业

**18029100040**

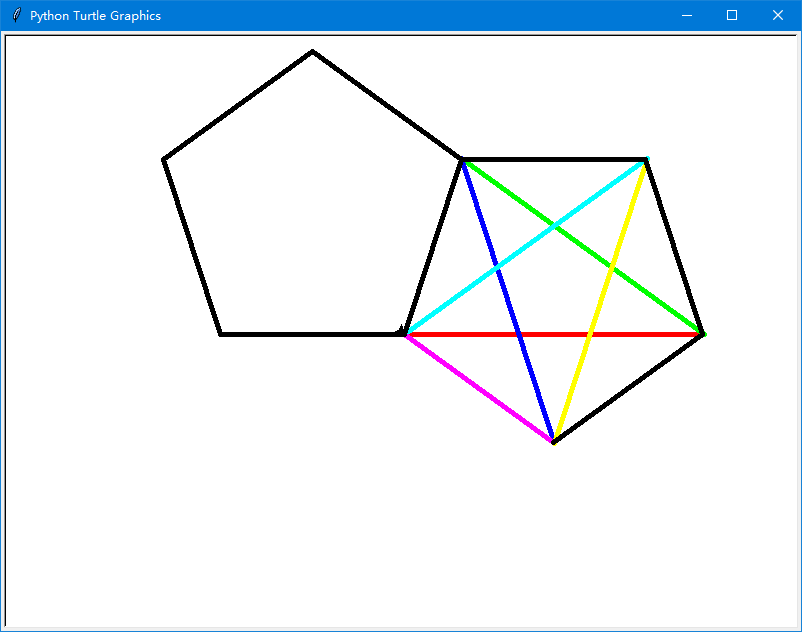
**吴程锴**

# 作业 11：自动轨迹绘制

## 代码

1. #AutoTrace.py
2. **import** turtle
3. turtle.setup(800,600)
4. turtle.pencolor("red")
5. turtle.pensize(5)
6. turtle.speed(5)
8. data = []
9. f = open("data.txt","r") #打开文件
10. **for** line **in** f:
11. line = line.replace('\n','')
12. data.append(list(map(eval,line.split(','))))
13. f.close()
15. **for** i **in** range(len(data)):
16. turtle.pencolor(data[i][3],data[i][4],data[i][5])
17. turtle.fd(data[i][0])
18. turtle.left(data[i][2]) **if** data[i][1] == 0 **else** turtle.right(data[i][2])
19. turtle.done

## 结果

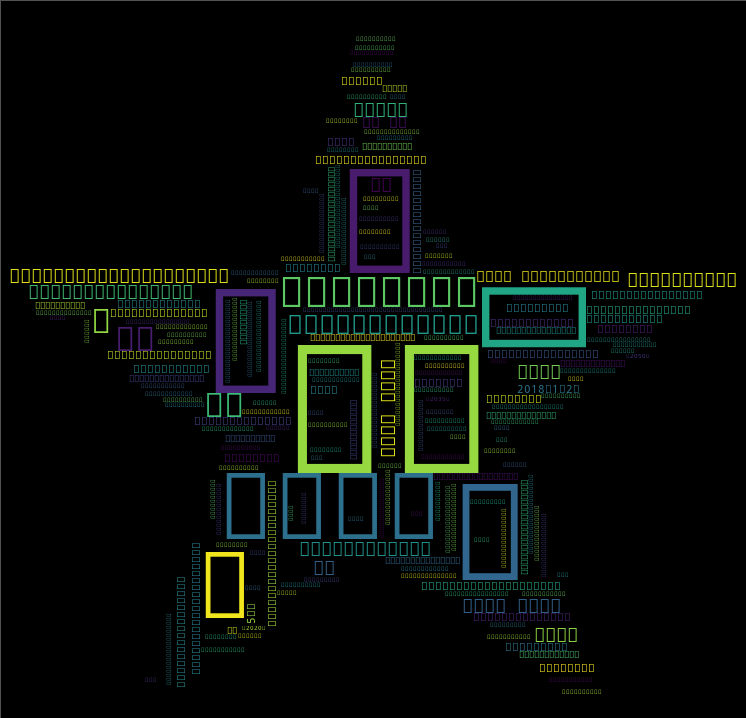


# 作业 12：词云处理

## 代码

1. **from** wordcloud **import** WordCloud
2. **import** PIL.Image as image
3. **import** numpy
5. with open("关于实施乡村振兴战略的意见.txt",encoding='utf-8') as fp:
6. text = fp.read()
7. mask = numpy.array(image.open("fivestart.png"))
8. wordcloud = WordCloud(mask = mask).generate(text)
9. image\_produce = wordcloud.to\_image()
10. image\_produce.show()

## 结果

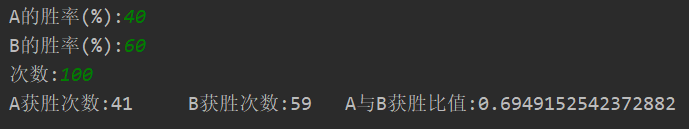


# 作业 13：模拟比赛

## 代码

1. **from** random **import** random
2. **import** math
4. rate\_A = eval(input("A的胜率(%):"))
5. rate\_B = eval(input("B的胜率(%):"))
6. times = eval(input("次数:"))
7. win\_A = 0
8. win\_B = 0
10. **for** i **in** range(times):
11. rate = random() \* 100
12. **if** rate < rate\_A:
13. win\_A = win\_A + 1
14. **else**:
15. win\_B = win\_B + 1
16. **print**("A获胜次数:{0}     B获胜次数:{1}   A与B获胜比值:{2}".format(win\_A,win\_B,win\_A/win\_B))

## 结果



# 作业 14：第三方库安装脚本

## 代码

1. **import** os
2. list = ["numpy","matplotlib","pillow","sklearn","requests","jieba","beautifulsoup4"]
4. **try**:
5. **for** lib **in** list:
6. os.system("pip install" + lib)
7. **print**("Successful!")
8. **except**:
9. **print**("Failed!")

## 结果

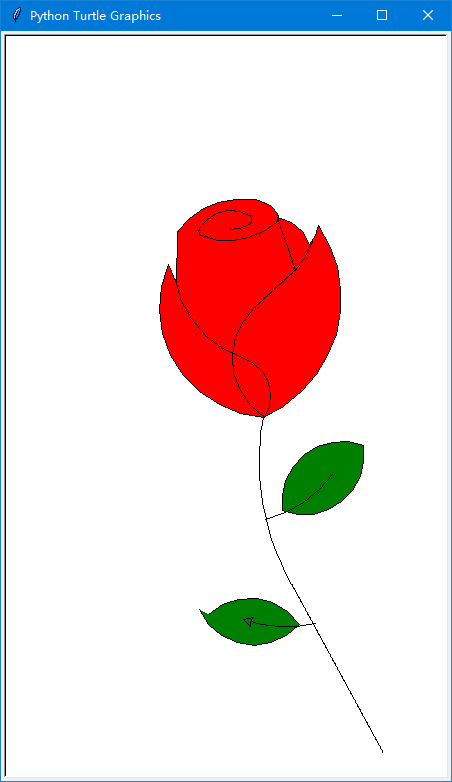


# 作业 15：玫瑰花绘制

## 代码

1. #RoseDraw.py
2. **import** turtle as t
3. # 定义一个曲线绘制函数
4. **def** DegreeCurve(n, r, d=1):
5. **for** i **in** range(n):
6. t.left(d)
7. t.circle(r, abs(d))
8. # 初始位置设定
9. s = 0.2 # size
10. t.setup(450\*5\*s, 750\*5\*s)
11. t.pencolor("black")
12. t.fillcolor("red")
13. t.speed(100)
14. t.penup()
15. t.goto(0, 900\*s)
16. t.pendown()
17. # 绘制花朵形状
18. t.begin\_fill()
19. t.circle(200\*s,30)
20. DegreeCurve(60, 50\*s)
21. t.circle(200\*s,30)
22. DegreeCurve(4, 100\*s)
23. t.circle(200\*s,50)
24. DegreeCurve(50, 50\*s)
25. t.circle(350\*s,65)
26. DegreeCurve(40, 70\*s)
27. t.circle(150\*s,50)
28. DegreeCurve(20, 50\*s, -1)
29. t.circle(400\*s,60)
30. DegreeCurve(18, 50\*s)
31. t.fd(250\*s)
32. t.right(150)
33. t.circle(-500\*s,12)
34. t.left(140)
35. t.circle(550\*s,110)
36. t.left(27)
37. t.circle(650\*s,100)
38. t.left(130)
39. t.circle(-300\*s,20)
40. t.right(123)
41. t.circle(220\*s,57)
42. t.end\_fill()
43. # 绘制花枝形状
44. t.left(120)
45. t.fd(280\*s)
46. t.left(115)
47. t.circle(300\*s,33)
48. t.left(180)
49. t.circle(-300\*s,33)
50. DegreeCurve(70, 225\*s, -1)
51. t.circle(350\*s,104)
52. t.left(90)
53. t.circle(200\*s,105)
54. t.circle(-500\*s,63)
55. t.penup()
56. t.goto(170\*s,-30\*s)
57. t.pendown()
58. t.left(160)
59. DegreeCurve(20, 2500\*s)
60. DegreeCurve(220, 250\*s, -1)
61. # 绘制一个绿色叶子
62. t.fillcolor('green')
63. t.penup()
64. t.goto(670\*s,-180\*s)
65. t.pendown()
66. t.right(140)
67. t.begin\_fill()
68. t.circle(300\*s,120)
69. t.left(60)
70. t.circle(300\*s,120)
71. t.end\_fill()
72. t.penup()
73. t.goto(180\*s,-550\*s)
74. t.pendown()
75. t.right(85)
76. t.circle(600\*s,40)
77. # 绘制另一个绿色叶子
78. t.penup()
79. t.goto(-150\*s,-1000\*s)
80. t.pendown()
81. t.begin\_fill()
82. t.rt(120)
83. t.circle(300\*s,115)
84. t.left(75)
85. t.circle(300\*s,100)
86. t.end\_fill()
87. t.penup()
88. t.goto(430\*s,-1070\*s)
89. t.pendown()
90. t.right(30)
91. t.circle(-600\*s,35)
92. t.done()

## 结果

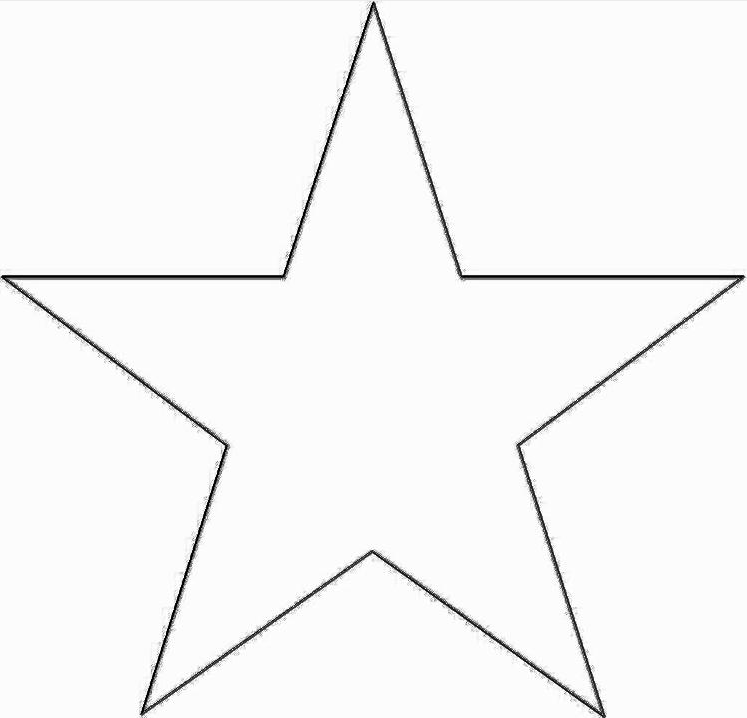


# 作业 16：图像转手绘效果

## 代码

1. **from** PIL **import** Image
2. **import** numpy as np
4. a = np.asarray(Image.open(r"fivestart.png").convert('L')).astype('float')
6. depth = 10.
7. grad = np.gradient(a)  # 梯度值，
8. grad\_x, grad\_y = grad
9. grad\_x = grad\_x \* depth / 100.  # 列梯度值\*0.1
10. grad\_y = grad\_y \* depth / 100.
12. A = np.sqrt(grad\_x \*\* 2 + grad\_y \*\* 2 + 1.)  # 相当于grad\_z=1
13. uni\_x = grad\_x / A
14. uni\_y = grad\_y / A
15. uni\_z = 1. / A  # 梯度归一化
17. vec\_el = np.pi / 2.2
18. vec\_az = np.pi / 4.
20. dx = np.cos(vec\_el) \* np.cos(vec\_az)
21. dy = np.cos(vec\_el) \* np.sin(vec\_az)
22. dz = np.sin(vec\_el)  # 长度为1，投影x,y,z长度
24. b = 255 \* (dx \* uni\_x + dy \* uni\_y + dz \* uni\_z)
25. b = b.clip(0, 255)
27. im = Image.fromarray(b.astype('uint8'))
28. im.save(r"fivestart\_Draw.jpg")

## 结果

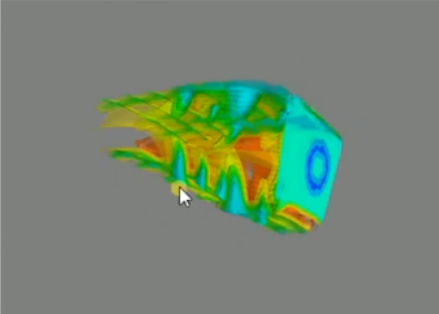


# 作业 17：标量数据绘制

## 代码

1. **from** tvtk.api **import** tvtk
2. **from** tvtkfunc **import** ivtk\_scene,event\_loop
4. plot3d = tvtk.MultiblockPLOT3DReader(
5. xyz\_file\_name = "combxyz.bin",
6. q\_file\_name = "combq.bin",
7. scalar\_function\_number = 100,
8. vector\_function\_number = 200
9. )
10. plot3d.update()
11. grid = plot3d.output.get\_block(0)
13. con = tvtk.ContourFilter()
14. con.set\_input\_data(grid)
15. con.generate\_values(10,grid.point\_data.scalars.range)
16. m = tvtk.PolyDataMapper(scalar\_range = grid.point\_data.scalars.range,
17. input\_connection = con.output\_port)
18. a = tvtk.Actor(mapper = m)
19. a.property.opacity = 0.5
20. win = ivtk\_scene(a)
21. win.scene.isometric\_view()
22. event\_loop()

## 结果



# 作业 18：矢量数据可视化

## 代码

1. **from** tvtk.api **import** tvtk
2. **from** Tvtkfunc **import** ivtk\_scene,event\_loop
4. **def** read\_data():    #导入数据
5. plot3d = tvtk.MultiBlockPLOT3DReader(
6. xyz\_file\_name="comxyz.bin", #网格文件
7. q\_file\_name="combq.bin", #开启动力学结果文件
8. scalar\_function\_number = 100, #设置标量数据数量
9. vector\_function\_number=200, #设置矢量数据数量
10. )   #读入Plot3D数据
11. plot3d.update() #让plot3D计算器输出数据
12. **return** plot3d

15. plot3d = read\_data()
16. grid = plot3d.output.get\_block(0)   #获取读入的数据集对象
18. #对数据集中的数据进行随机选取，每50个点选择一个点,是对数据进行降采样
19. mask = tvtk.MaskPoints(random\_mode=True,on\_ratio=50)
20. mask.set\_input\_data(grid)   #将grid和mask相连
21. #创建表示箭头的PolyData数据集
22. glyph\_source = tvtk.ArrowSource()
23. #在Mask采样后的PolyData数据集每个点上放置一个箭头
24. #箭头的方向（速度方向），长度<箭头越大，表示标量越大>和颜色<也表示标量大小，红色小，蓝色大>（两个都表示密度）由于点对应的矢量和标量数据决定
26. #将上面的降采样数据与箭头符号化相关联
27. glyph = tvtk.Glyph3D(input\_connection=mask.output\_port,
28. scale\_factor=4)    #scale\_factor符号的共同放缩系数
29. glyph.set\_source\_connection(glyph\_source.output\_port)
31. m = tvtk.PolyDataMapper(scalar\_range=grid.point\_data.scalars.range, #设置映射器的变量范围属性
32. input\_connection=glyph.output\_port)
33. a = tvtk.Actor(mapper=m)
34. a.property.opacity = 0.5    #设置透明度为0.5
36. win = ivtk\_scene(a)
37. win.scene.isometric\_view()
38. event\_loop()

## 结果

