

Python语言程序设计

# 第9章 Python计算生态概览

---





# 前课复习

# Python基础语法 (全体系)

## ① 基本数据类型

- 整数、浮点数、复数
- 字符串

## ③ 函数和代码复用

- 函数定义和使用
- 函数递归

## ⑤ 文件和数据格式化

- 文件的使用
- 一二维数据的表示存储和处理

## ② 程序的控制结构

- 分支结构与异常处理
- 遍历循环、无限循环

## ④ 组合数据类型

- 集合类型
- 序列类型：元组和列表
- 字典类型



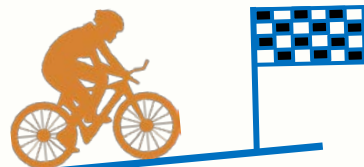
# Python程序设计思维

- 计算思维：抽象计算过程和自动化执行
- 计算生态：竞争发展、相互依存、快速更迭
- 用户体验：进度展示、异常处理等
- IPO、自顶向下、模块化、配置化、应用开发的四个步骤



# Python第三方库安装

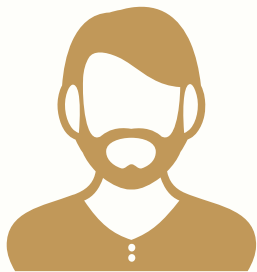
- **PyPI : Python Package Index**
- **pip命令的各种用法**
- **Anaconda集成开发工具及安装方法**
- **UCI页面的“补丁”安装方法**





# 本课概要

# 第9章 Python计算生态概览



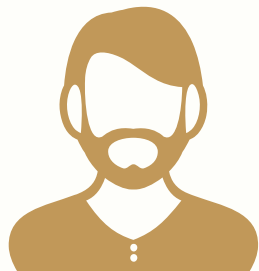
- 9.1 从数据处理到人工智能
- 9.2 实例15: 霍兰德人格分析雷达图
- 9.3 从Web解析到网络空间
- 9.4 从人机交互到艺术设计
- 9.5 实例16: 玫瑰花绘制



# 第9章 Python计算生态概览

## 方法论

- 纵览Python计算生态，看见更大的世界



## 实践能力

- 初步编写带有计算生态的复杂程序





# Python语言程序设计

## 9.1 从数据处理到人工智能

---



# 从数据处理到人工智能

数据表示->数据清洗->数据统计->数据可视化->数据挖掘->人工智能

- **数据表示**：采用合适方式用程序表达数据
- **数据清理**：数据归一化、数据转换、异常值处理
- **数据统计**：数据的概要理解，数量、分布、中位数等

# 从数据处理到人工智能

数据表示->数据清洗->数据统计->数据可视化->数据挖掘->人工智能

- **数据可视化**：直观展示数据内涵的方式
- **数据挖掘**：从数据分析获得知识，产生数据外的价值
- **人工智能**：数据/语言/图像/视觉等方面深度分析与决策

# 从数据处理到人工智能



- Python库之数据分析
- Python库之数据可视化
- Python库之文本处理
- Python库之机器学习





# Python库之数据分析

# Python库之数据分析

**Numpy:** 表达N维数组的最基础库

- Python接口使用，C语言实现，计算速度优异
- Python数据分析及科学计算的基础库，支撑Pandas等
- 提供直接的矩阵运算、广播函数、线性代数等功能

# Python库之数据分析

## Numpy: 表达N维数组的最基础库

```
def pySum():  
    a = [0, 1, 2, 3, 4]  
    b = [9, 8, 7, 6, 5]  
    c = []  
  
    for i in range(len(a)):  
        c.append(a[i]**2 + b[i]**3)  
  
    return c  
  
print(pySum())
```



```
import numpy as np  
  
def npSum():  
    a = np.array([0, 1, 2, 3, 4])  
    b = np.array([9, 8, 7, 6, 5])  
  
    c = a**2 + b**3  
  
    return c  
  
print(npSum())
```



<http://www.numpy.org>

# Python库之数据分析

## **Pandas:** Python数据分析高层次应用库

- 提供了简单易用的数据结构和数据分析工具
- 理解数据类型与索引的关系，操作索引即操作数据
- Python最主要的数据分析功能库，基于Numpy开发



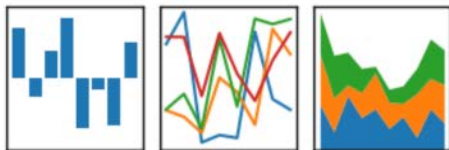
# Python库之数据分析

**Pandas:** Python数据分析高层次应用库

**Series = 索引 + 一维数据**

**DataFrame = 行列索引 + 二维数据**

pandas  
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



<http://pandas.pydata.org>

# Python库之数据分析

**SciPy: 数学、科学和工程计算功能库**

- 提供了一批数学算法及工程数据运算功能
- 类似Matlab，可用于如傅里叶变换、信号处理等应用
- Python最主要的科学计算功能库，基于Numpy开发

# Python库之数据分析

**SciPy: 数学、科学和工程相关功能库**



<http://www.scipy.org>



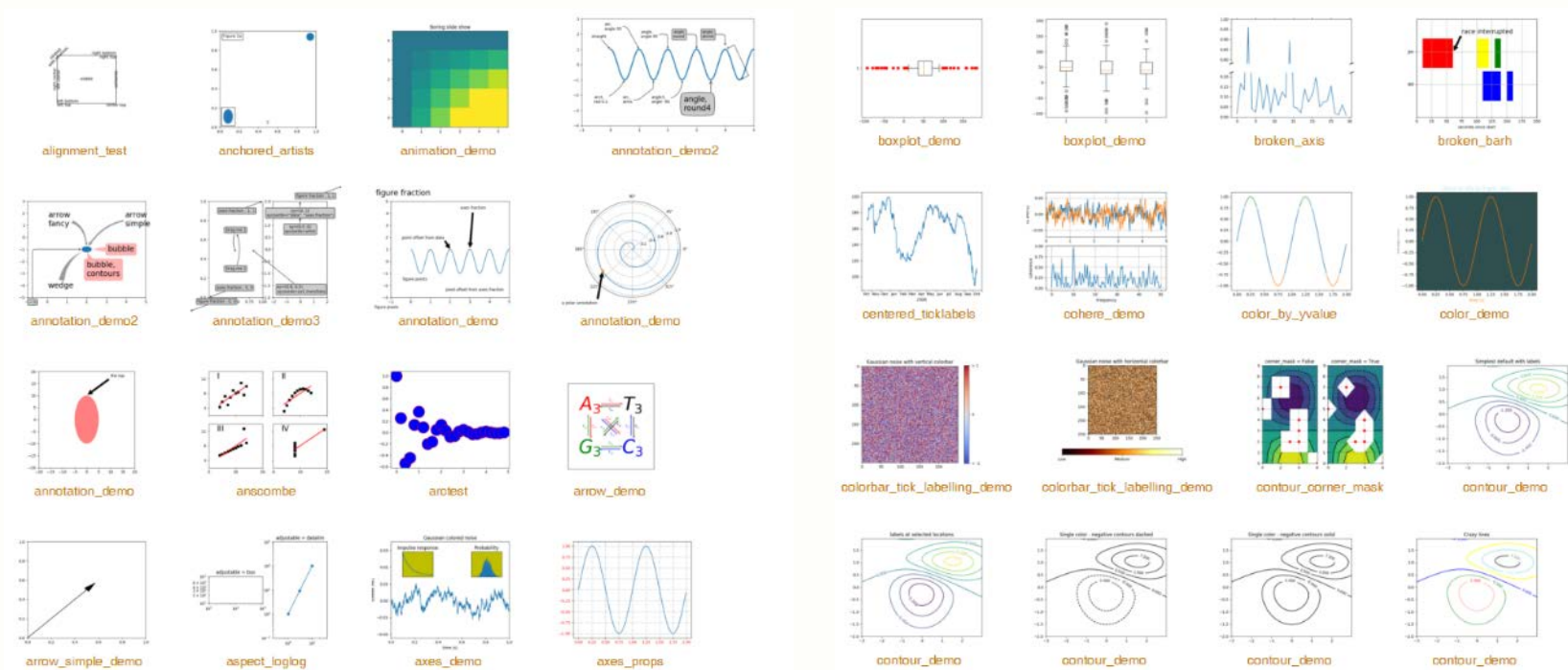
# Python库之数据可视化

# Python库之数据可视化

**Matplotlib:** 高质量的二维数据可视化功能库

- 提供了超过100种数据可视化展示效果
- 通过matplotlib.pyplot子库调用各可视化效果
- Python最主要的数据可视化功能库，基于Numpy开发

# Python库之数据可视化



<http://matplotlib.org>

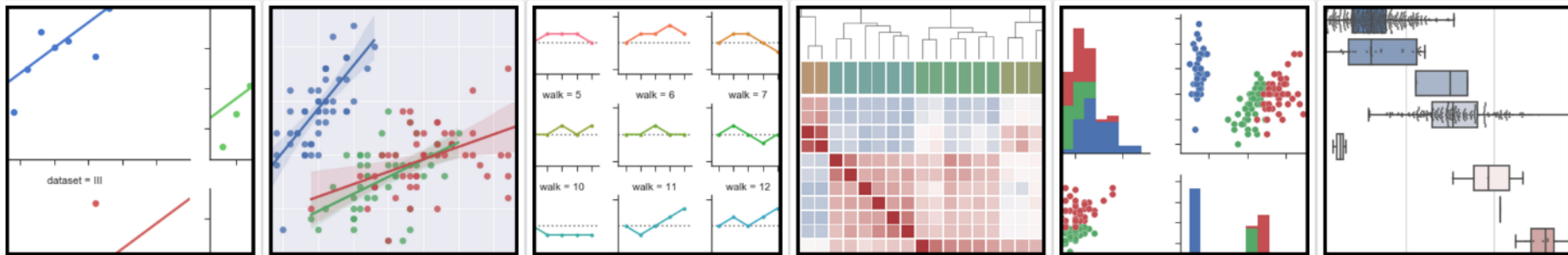
# Python库之数据可视化

## **Seaborn:** 统计类数据可视化功能库

- 提供了一批高层次的统计类数据可视化展示效果
- 主要展示数据间分布、分类和线性关系等内容
- 基于Matplotlib开发，支持Numpy和Pandas

# Python库之数据可视化

## Seaborn: 统计类数据可视化功能库



<http://seaborn.pydata.org/>



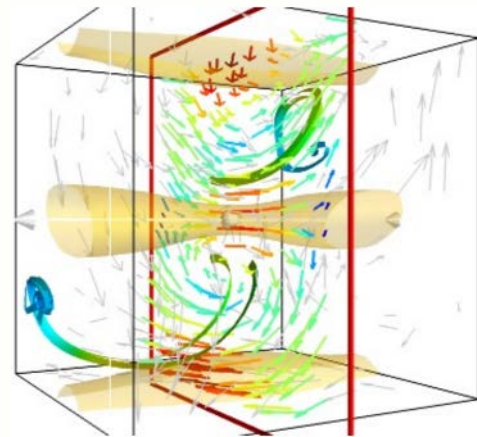
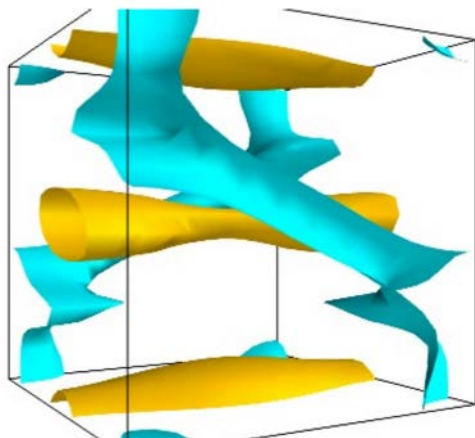
# Python之数据可视化

## **Mayavi** : 三维科学数据可视化功能库

- 提供了一批简单易用的3D科学计算数据可视化展示效果
- 目前版本是Mayavi2 , 三维可视化最主要的第三方库
- 支持Numpy、TVTK、Traits、Envisage等第三方库

# Python之数据可视化

**Mayavi** : 三维科学数据可视化功能库



<http://docs.enthought.com/mayavi/mayavi/>



# Python库之文本处理

# Python之文本处理

## **PyPDF2** : 用来处理pdf文件的工具集

- 提供了一批处理PDF文件的计算功能
- 支持获取信息、分隔/整合文件、加密解密等
- 完全Python语言实现，不需要额外依赖，功能稳定

# Python之文本处理

## PyPDF2 : 用来处理pdf文件的工具集

```
from PyPDF2 import PdfFileReader, PdfFileMerger
```

```
merger = PdfFileMerger()
```

```
input1 = open("document1.pdf", "rb")
```

```
input2 = open("document2.pdf", "rb")
```

```
merger.append(fileobj = input1, pages = (0,3))
```

```
merger.merge(position = 2, fileobj = input2, pages = (0,1))
```

```
output = open("document-output.pdf", "wb")
```

```
merger.write(output)
```

<http://mstamy2.github.io/PyPDF2>

# Python之文本处理

## **NLTK**：自然语言文本处理第三方库

- 提供了一批简单易用的自然语言文本处理功能
- 支持语言文本分类、标记、语法句法、语义分析等
- 最优秀的Python自然语言处理库

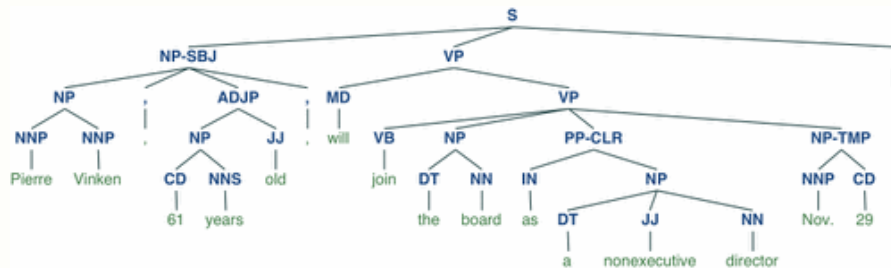
# Python之文本处理

## NLTK : 自然语言文本处理第三方库

```
from nltk.corpus import treebank
```

```
t = treebank.parsed_sents('wsj_0001.mrg')[0]
```

```
t.draw()
```



<http://www.nltk.org/>

# Python之文本处理

**Python-docx** : 创建或更新Microsoft Word文件的第三方库

- 提供创建或更新.doc .docx等文件的计算功能
- 增加并配置段落、图片、表格、文字等，功能全面



# Python之文本处理

**Python-docx** : 创建或更新Microsoft Word文件的第三方库

```
from docx import Document  
document = Document()  
document.add_heading('Document Title', 0)  
p = document.add_paragraph('A plain paragraph having some ')  
document.add_page_break()  
document.save('demo.docx')
```

<http://python-docx.readthedocs.io/en/latest/index.html>



# Python库之机器学习

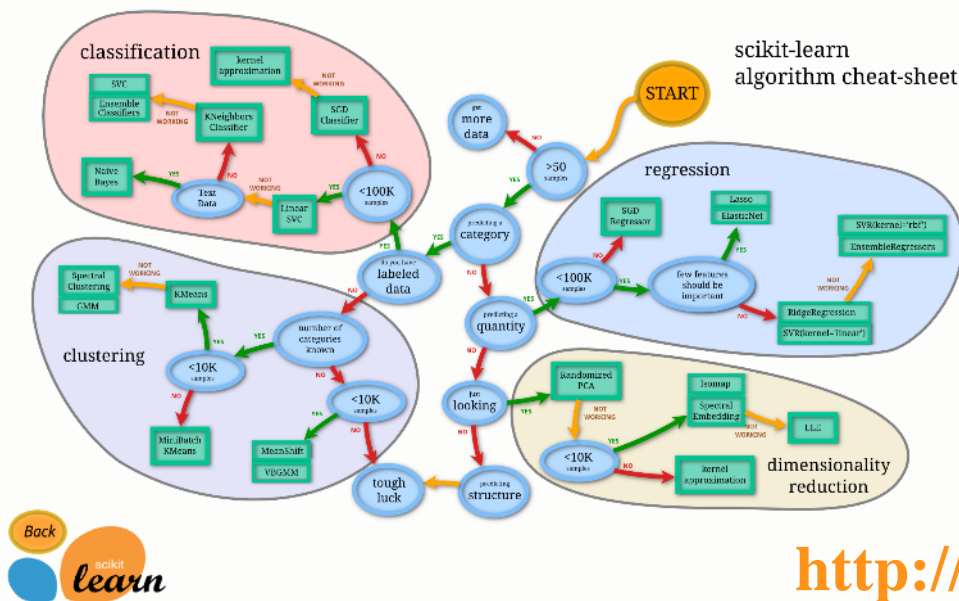
# Python之机器学习

## Scikit-learn : 机器学习方法工具集

- 提供一批统一化的机器学习方法功能接口
- 提供聚类、分类、回归、强化学习等计算功能
- 机器学习最基本且最优秀的Python第三方库

# Python之机器学习

## Scikit-learn : 与数据处理相关的第三方库



<http://scikit-learn.org/>

# Python之机器学习

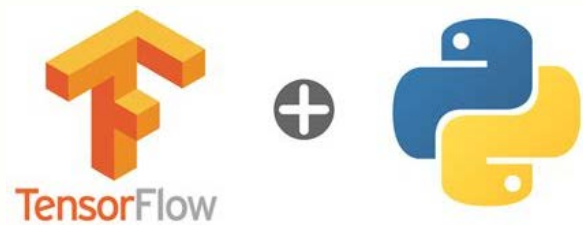
## **TensorFlow** : **AlphaGo**背后的机器学习计算框架

- 谷歌公司推动的开源机器学习框架
- 将数据流图作为基础，图节点代表运算，边代表张量
- 应用机器学习方法的一种方式，支撑谷歌人工智能应用

# Python之机器学习

## TensorFlow : AlphaGo背后的机器学习计算框架

```
import tensorflow as tf  
init = tf.global_variables_initializer()  
sess = tf.Session()  
sess.run(init)  
res = sess.run(result)  
print('result:', res)
```



<https://www.tensorflow.org/>

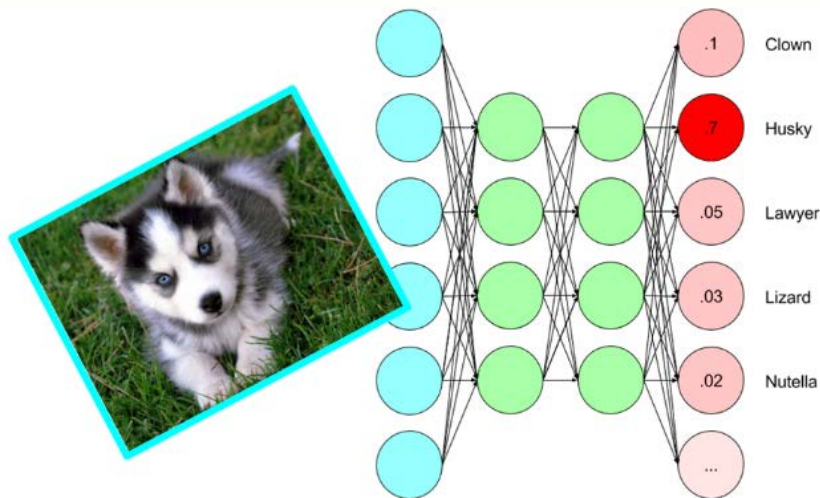
# Python之机器学习

**MXNet** : 基于神经网络的深度学习计算框架

- 提供可扩展的神经网络及深度学习计算功能
- 可用于自动驾驶、机器翻译、语音识别等众多领域
- Python最重要的深度学习计算框架

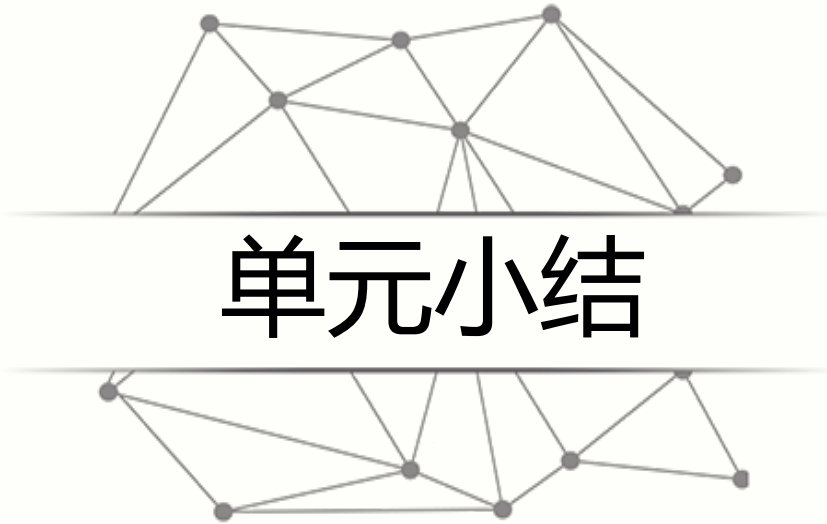
# Python之机器学习

**MXNet** : 基于神经网络的深度学习计算框架



<https://mxnet.incubator.apache.org/>





# 单元小结

# 从数据处理到人工智能

- Numpy、Pandas、SciPy
- Matplotlib、Seaborn、Mayavi
- PyPDF2、NLTK、python-docx
- Scikit-learn、TensorFlow、MXNet



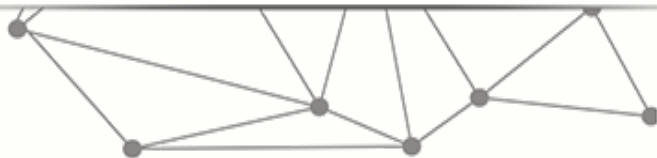
## 9.2 实例15: 霍兰德人格分析雷达图

---



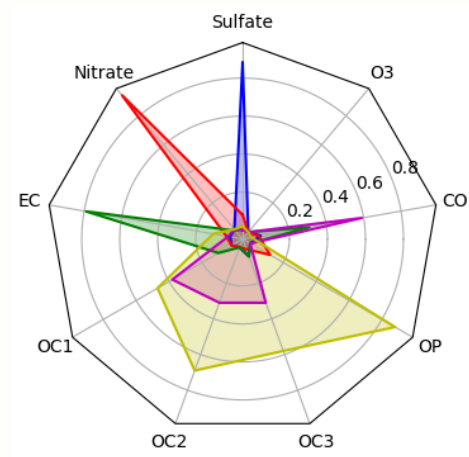
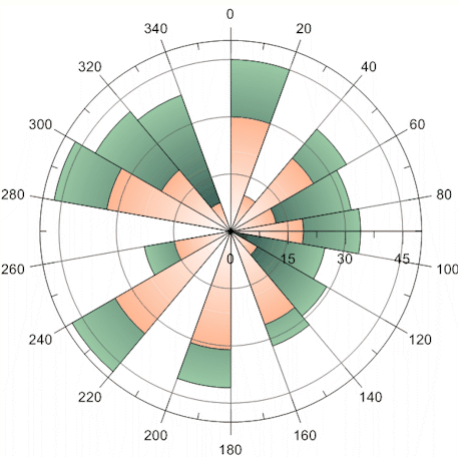


# "霍兰德人格分析雷达图"问题分析



# 问题分析

## 雷达图 Radar Chart



雷达图是多特性直观展示的重要方式

# 问题分析

## 霍兰德人格分析

- 霍兰德认为：人格兴趣与职业之间应有一种内在的对应关系
- 人格分类：研究型、艺术型、社会型、企业型、传统型、现实性
- 职业：工程师、实验员、艺术家、推销员、记事员、社会工作者

# 问题分析

## 霍兰德人格分析雷达图

- 需求：雷达图方式验证霍兰德人格分析
- 输入：各职业人群结合兴趣的调研数据
- 输出：雷达图

# 问题分析

## 霍兰德人格分析雷达图

- 通用雷达图绘制：matplotlib库
- 专业的多维数据表示：numpy库
- 输出：雷达图





# "霍兰德人格分析雷达图"实例展示



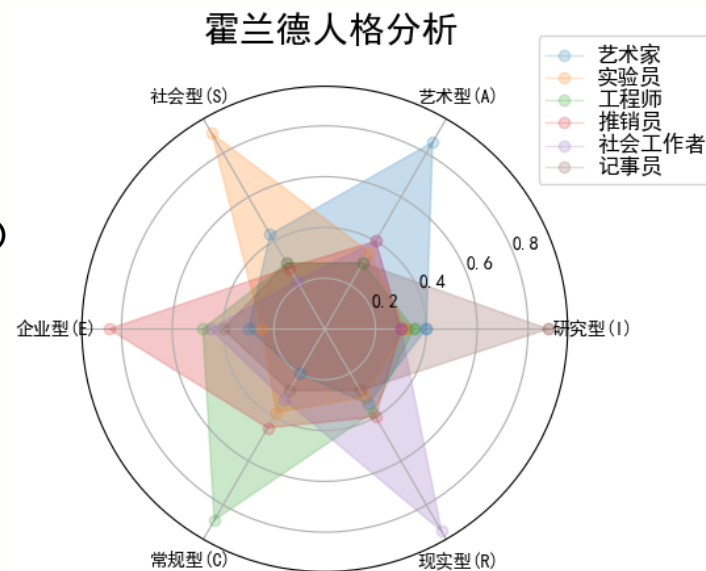
```

#HollandRadarDraw
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family']='SimHei'
radar_labels = np.array(['研究型(I)', '艺术型(A)', '社会型(S)', \
                          '企业型(E)', '常规型(C)', '现实型(R)'])

data = np.array([[0.40, 0.32, 0.35, 0.30, 0.30, 0.88],
                 [0.85, 0.35, 0.30, 0.40, 0.40, 0.30],
                 [0.43, 0.89, 0.30, 0.28, 0.22, 0.30],
                 [0.30, 0.25, 0.48, 0.85, 0.45, 0.40],
                 [0.20, 0.38, 0.87, 0.45, 0.32, 0.28],
                 [0.34, 0.31, 0.38, 0.40, 0.92, 0.28]]) #数据值

data_labels = ('艺术家', '实验员', '工程师', '推销员', '社会工作者', '记事员')
angles = np.linspace(0, 2*np.pi, 6, endpoint=False)
data = np.concatenate((data, [data[0]]))
angles = np.concatenate((angles, [angles[0]]))
fig = plt.figure(facecolor="white")
plt.subplot(111, polar=True)
plt.plot(angles, data, 'o-', linewidth=1, alpha=0.2)
plt.fill(angles, data, alpha=0.25)
plt.thetagrids(angles*180/np.pi, radar_labels, frac = 1.2)
plt.figtext(0.52, 0.95, '霍兰德人格分析', ha='center', size=20)
legend = plt.legend(data_labels, loc=(0.94, 0.80), labelspace=0.1)
plt.setp(legend.get_texts(), fontsize='large')
plt.grid(True)
plt.savefig('holland_radar.jpg')
plt.show()

```



```
#HollandRadarDraw
```

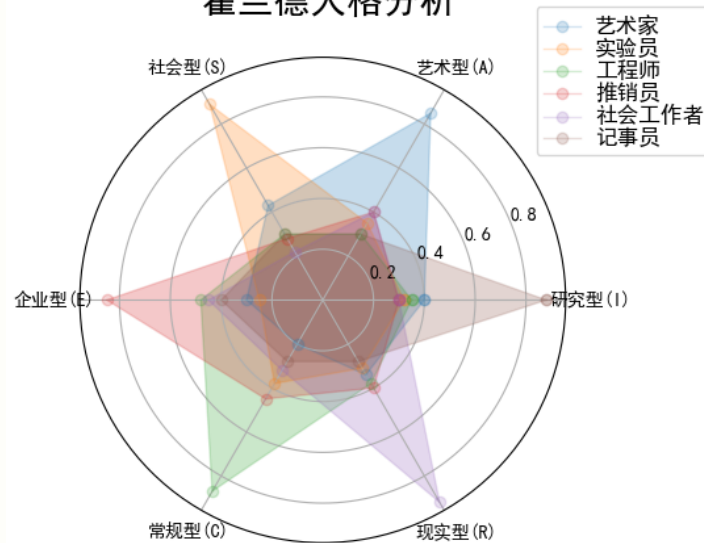
```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib
```

(略)

霍兰德人格分析



(略)

```
matplotlib.rcParams['font.family']='SimHei'
radar_labels = np.array(['研究型(I)', '艺术型(A)', '社会型(S)', \
                          '企业型(E)', '常规型(C)', '现实型(R)'])

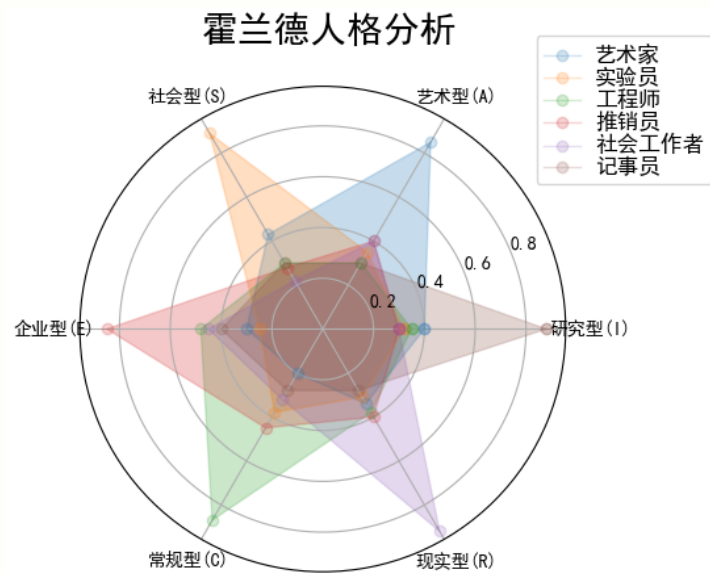
data = np.array([[0.40, 0.32, 0.35, 0.30, 0.30, 0.88],
                 [0.85, 0.35, 0.30, 0.40, 0.40, 0.30],
                 [0.43, 0.89, 0.30, 0.28, 0.22, 0.30],
                 [0.30, 0.25, 0.48, 0.85, 0.45, 0.40],
                 [0.20, 0.38, 0.87, 0.45, 0.32, 0.28],
                 [0.34, 0.31, 0.38, 0.40, 0.92, 0.28]]) #数据值

data_labels = ('艺术家', '实验员', '工程师', '推销员', '社会工作者', '记事员')
(略)
```

(略)

```
angles = np.linspace(0, 2*np.pi, 6, endpoint=False)
data = np.concatenate((data, [data[0]]))
angles = np.concatenate((angles, [angles[0]]))
fig = plt.figure(facecolor="white")
plt.subplot(111, polar=True)
plt.plot(angles,data,'o-', linewidth=1, alpha=0.2)
plt.fill(angles,data, alpha=0.25)
plt.thetagrids(angles*180/np.pi, radar_labels,frac = 1.2)
```

(略)



(略)

```
plt.figtext(0.52, 0.95, '霍兰德人格分析', ha='center', size=20)
legend = plt.legend(data_labels, loc=(0.94, 0.80), labelspace=0.1)
plt.setp(legend.get_texts(), fontsize='large')
plt.grid(True)
plt.savefig('holland_radar.jpg')
plt.show()
```

Python语言程序设计

## 9.3 从Web解析到网络空间

---



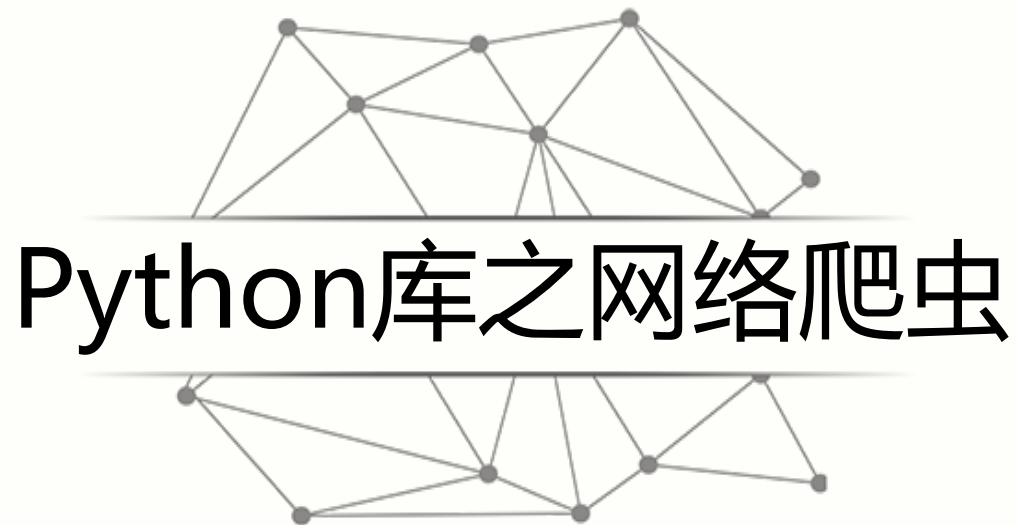
# 从Web解析到网络空间



- Python库之网络爬虫
- Python库之Web信息提取
- Python库之Web网站开发
- Python库之网络应用开发







# Python库之网络爬虫

# Python库之网络爬虫

**Requests: 最友好的网络爬虫功能库**

- 提供了简单易用的类HTTP协议网络爬虫功能
- 支持连接池、SSL、Cookies、HTTP(S)代理等
- Python最主要的页面级网络爬虫功能库

# Python库之网络爬虫

## Requests: 最友好的网络爬虫功能库

```
import requests  
  
r = requests.get('https://api.github.com/user',\  
                 auth=('user', 'pass'))  
  
r.status_code  
r.headers['content-type']  
r.encoding  
r.text
```



<http://www.python-requests.org/>

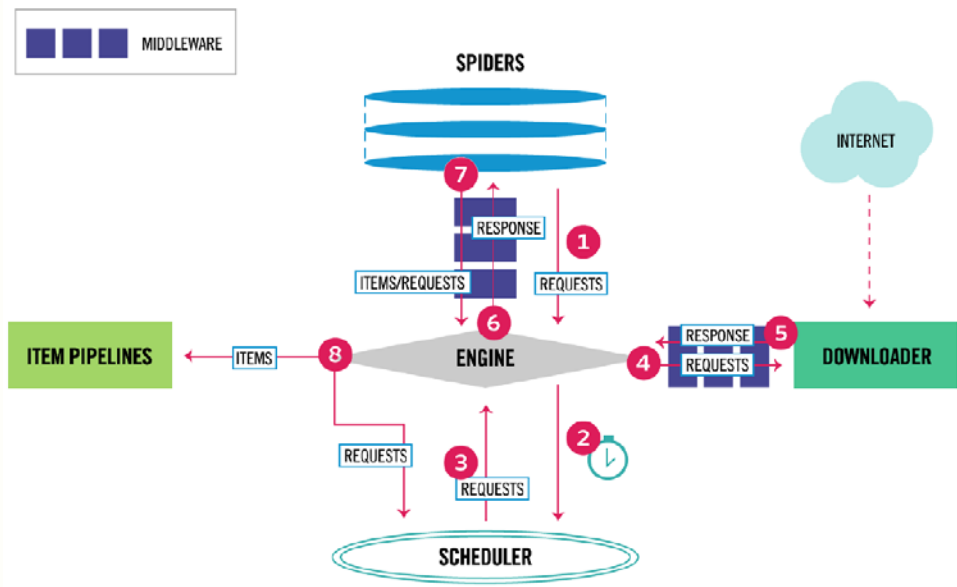
# Python库之网络爬虫

## **Scrapy:** 优秀的网络爬虫框架

- 提供了构建网络爬虫系统的框架功能，功能半成品
- 支持批量和定时网页爬取、提供数据处理流程等
- Python最主要且最专业的网络爬虫框架

# Python库之网络爬虫

**Scrapy:** Python数据分析高层次应用库



<https://scrapy.org>

# Python库之网络爬虫

**pyspider: 强大的Web页面爬取系统**

- 提供了完整的网页爬取系统构建功能
- 支持数据库后端、消息队列、优先级、分布式架构等
- Python重要的网络爬虫类第三方库

# Python库之网络爬虫

## pyspider: 强大的Web页面爬取系统

pyspider > js\_test\_sciencedirect

```
{
  "fetch": {
    "fetch_type": "js"
  },
  "process": {
    "callback": "detail_page"
  },
  "project": "js_test_sciencedirect",
  "taskId": "001d162322318ebba208ad9feb01d6ed",
  "url": "http://www.sciencedirect.com/science/article/pii/S0261560808000463"
}
```

run

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# vim: set et sw4 ts4 sts4 ff=unix fenc=utf8:
# Created on 2014-10-31 13:05:52

import re
from libs.base_handler import *

class Handler(BaseHandler):
    """
    this is a sample handler
    """
    def on_start(self):
        self.crawl('http://www.sciencedirect.com/science/article/pii/S1568494612005741',
                   callback=self.detail_page)

    def index_page(self, response):
        for each in response.doc('a').items():
            if re.match('http://www.sciencedirect.com/science/article/pii/\w+$',
                        each.attr.href):
                self.crawl(each.attr.href, callback=self.detail_page)

    @config(fetch_type="js")
    def detail_page(self, response):
        self.index_page(response)
        self.crawl(response.doc('HTML#BODY>DIV#page-
area>DIV#rightPanel>DIV#rightOuter>DIV#rightInner>DIV.InnerPadding>DIV#recommend_rela
ted_articles>OL#relatedArticles>LI>A.viewMoreArticles.click').attr.href,
                   callback=self.index_page)

    return {
        "url": response.url,
        "title": response.doc('HTML#BODY>DIV#page-
area>DIV#centerPanel>DIV#centerContent>DIV#centerInner>DIV#frag_1>H1.svTitle').text(),
        "authors": [{"name": x.text(), "url": x.attr.href} for x in
response.doc('HTML#BODY>DIV#page-
area>DIV#centerPanel>DIV#centerContent>DIV#centerInner>DIV#frag_1>UL.authorGroup.noCo
llab>LI.smh>A.authorName').items()],
        "abstract": response.doc('HTML#BODY>DIV#page-
area>DIV#centerPanel>DIV#centerContent>DIV#centerInner>DIV#frag_2>DIV.abstract.svAbst
ract').text(),
        "keywords": [x.text() for x in response.doc('HTML#BODY>DIV#page-
area>DIV#centerPanel>DIV#centerContent>DIV#centerInner>DIV#frag_2>UL.keyword>LI.svkey
word').items()],
        "url": 'http://www.sciencedirect.com/science/article/pii/S0261560808000463'
    }

    'keywords': [],
    'title': 'Editorial',
    'url': 'http://www.sciencedirect.com/science/article/pii/S0261560808000463'
}
```

Quickstart 脚本编写指南 save

ScienceDirect Journals Books Shopping cart

Purchase Export Search SciencDirect Advanced search

Journal of International Money and Finance  
Volume 21, Issue 6, November 2002, Pages 693  
International Financial Integration

Editorial  
J.R Lothian  
Show more

Choose an option to locate/access this article:

Check if you have access through your login credentials or your institution  
Purchase \$35.95 Get Full Text Elsewhere

enable css selector helper web html follow 21 messages

<http://docs.pyspider.org>



# Python库之Web信息提取



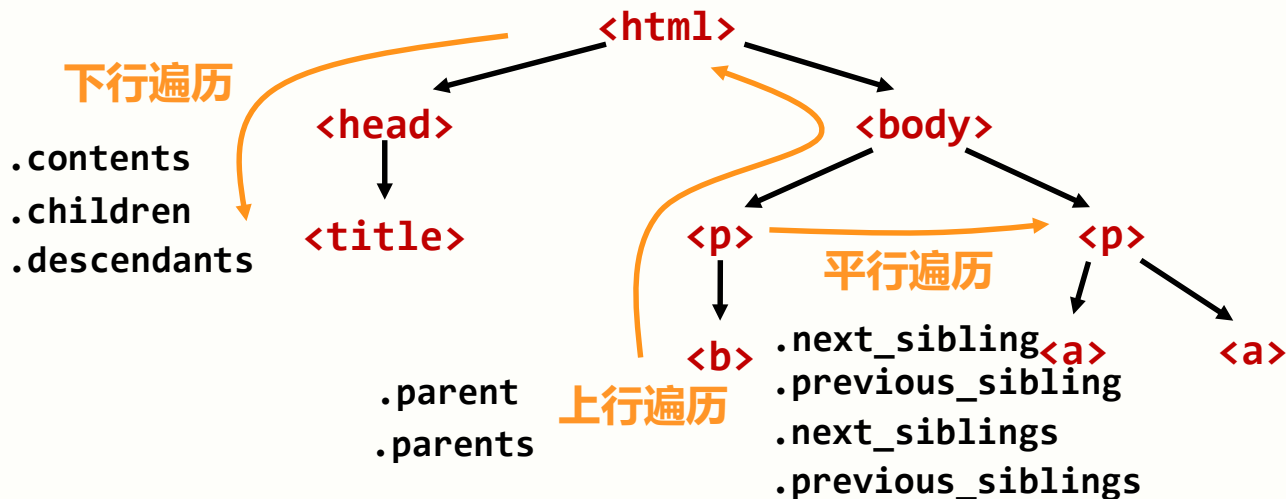
# Python库之Web信息提取

## **Beautiful Soup:** HTML和XML的解析库

- 提供了解析HTML和XML等Web信息的功能
- 又名beautifulsoup4或bs4，可以加载多种解析引擎
- 常与网络爬虫库搭配使用，如Scrapy、requests等

# Python库之Web信息提取

## Beautiful Soup: HTML和XML的解析库



<https://www.crummy.com/software/BeautifulSoup/bs4>

# Python库之Web信息提取

## **Re: 正则表达式解析和处理功能库**

- 提供了定义和解析正则表达式的一批通用功能
- 可用于各类场景，包括定点的Web信息提取
- Python最主要的标准库之一，无需安装

# Python库之Web信息提取

## Re: 正则表达式解析和处理功能库

`re.search()`

`re.split()`

`re.match()`

`r'\d{3}-\d{8}|\d{4}-\d{7}'`

`re.finditer()`

`re.findall()`

`re.sub()`

<https://docs.python.org/3.6/library/re.html>

# Python库之Web信息提取

**Python-Goose:** 提取文章类型Web页面的功能库

- 提供了对Web页面中文章信息/视频等元数据的提取功能
- 针对特定类型Web页面，应用覆盖面较广
- Python最主要的Web信息提取库

# Python库之Web信息提取

## Python-Goose: 提取文章类型Web页面的功能库

```
from goose import Goose

url = 'http://www.elmundo.es/elmundo/2012/10/28/espana/1351388909.html'

g = Goose({'use_meta_language': False, 'target_language': 'es'})

article = g.extract(url=url)

article.cleaned_text[:150]
```

<https://github.com/grangier/python-goose>



# Python库之Web网站开发

# Python库之Web网站开发

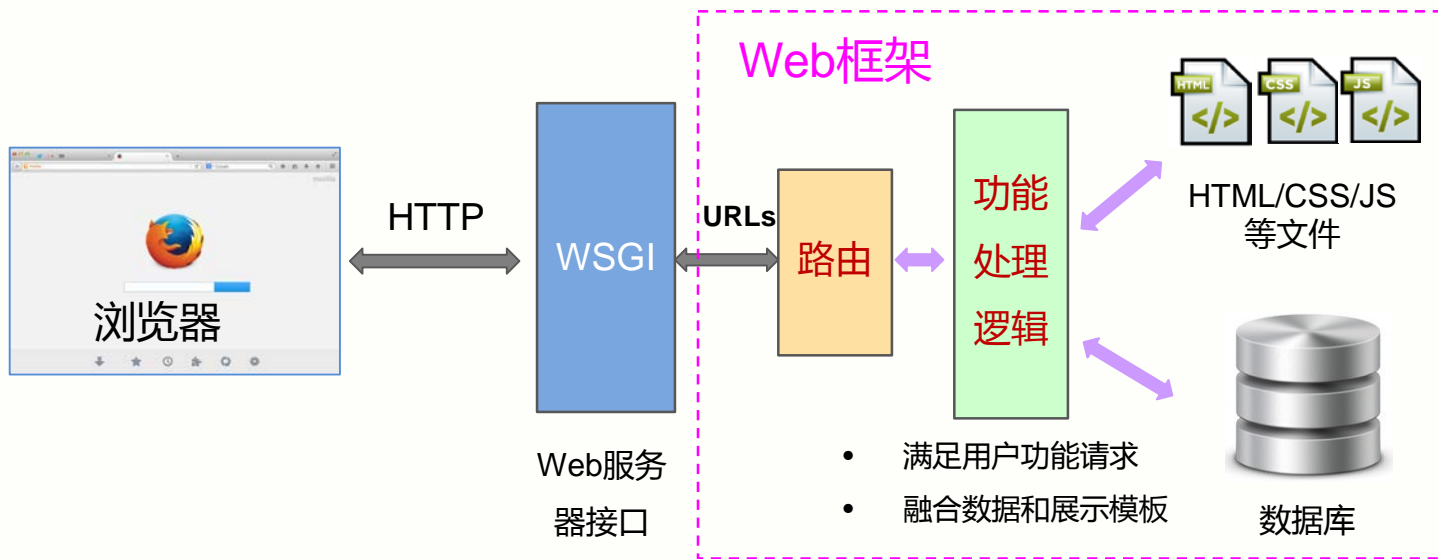
## **Django:** 最流行的Web应用框架

- 提供了构建Web系统的基本应用框架
- MTV模式：模型(model)、模板(Template)、视图(Views)
- Python最重要的Web应用框架，略微复杂的应用框架



# Python库之Web网站开发

## Django: 最流行的Web应用框架



<https://www.djangoproject.com>

# Python库之Web网站开发

## **Pyramid:** 规模适中的Web应用框架

- 提供了简单方便构建Web系统的应用框架
- 不大不小，规模适中，适合快速构建并适度扩展类应用
- Python产品级Web应用框架，起步简单可扩展性好

# Python库之Web网站开发

## Pyramid: 规模适中的Web应用框架

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
def hello_world(request):
    return Response('Hello World!')
if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

- 10行左右Hello Word程序

<https://trypyramid.com/>

# Python库之Web网站开发

## Flask: Web应用开发微框架

- 提供了最简单构建Web系统的应用框架
- 特点是：简单、规模小、快速
- Django > Pyramid > Flask

# Python库之Web网站开发

## Flask: Web应用开发微框架

```
from flask import Flask  
  
app = Flask(__name__)  
  
@app.route('/')  
def hello_world():  
    return 'Hello, World!'
```



<http://flask.pocoo.org>



# Python库之网络应用开发

# Python库之网络应用开发

## **WeRoBot:** 微信公众号开发框架

- 提供了解析微信服务器消息及反馈消息的功能
- 建立微信机器人的重要技术手段

# Python库之Web网站开发

## WeRoBot: 微信公众号开发框架

```
import werobot
```

```
robot = werobot.WeRoBot(token='tokenhere')
```

```
@robot.handler
```

```
def hello(message):
```

- 对微信每个消息反馈一个Hello World

```
    return 'Hello World!'
```

<https://github.com/offu/WeRoBot>



# Python库之网络应用开发

**aip: 百度AI开放平台接口**

- 提供了访问百度AI服务的Python功能接口
- 语音、人脸、OCR、NLP、知识图谱、图像搜索等领域
- Python百度AI应用的最主要方式

# Python库之Web网站开发

**aip:** 百度AI开放平台接口



<https://github.com/Baidu-AIP/python-sdk>

# Python库之网络应用开发

## **MyQR:** 二维码生成第三方库

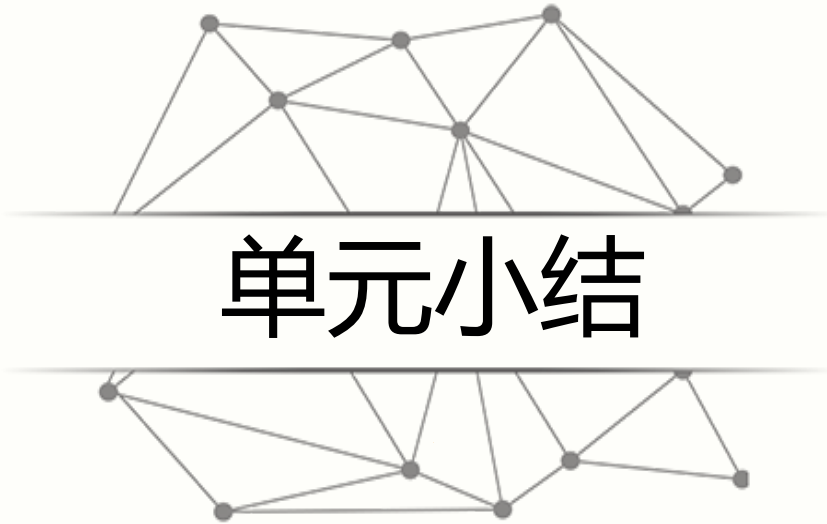
- 提供了生成二维码的系列功能
- 基本二维码、艺术二维码和动态二维码

# Python库之Web网站开发

**MyQR:** 二维码生成第三方库



<https://github.com/sylnsfar/qrcode>



# 单元小结

# 从Web解析到网络空间

- Requests、Scrapy、pyspider
- Beautiful Soup、Re、Python-Goose
- Django、Pyramid、Flask
- WeRobot、aip、MyQR



## Python语言程序设计

# 9.4 从人机交互到艺术设计

---



# 从人机交互到艺术设计



- Python库之图形用户界面
- Python库之游戏开发
- Python库之虚拟现实
- Python库之图形艺术







# Python库之图形用户界面

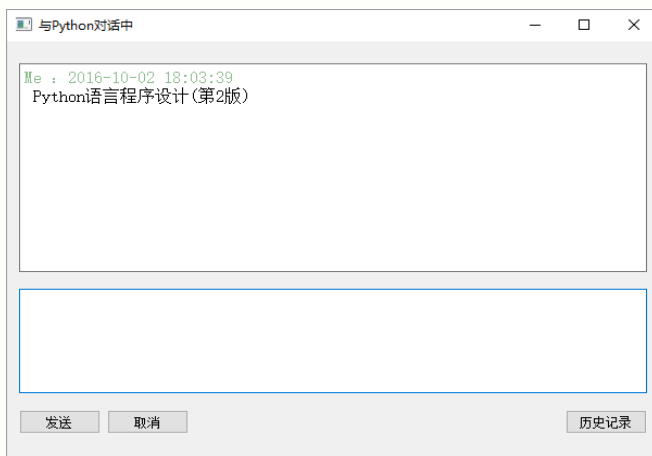
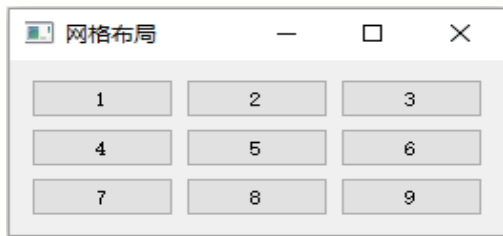
# Python库之图形用户界面

## PyQt5: Qt开发框架的Python接口

- 提供了创建Qt5程序的Python API接口
- Qt是非常成熟的跨平台桌面应用开发系统，完备GUI
- 推荐的Python GUI开发第三方库

# Python库之图形用户界面

## PyQt5: Qt开发框架的Python接口



<https://www.riverbankcomputing.com/software/pyqt>

# Python库之图形用户界面

**wxPython: 跨平台GUI开发框架**

- 提供了专用于Python的跨平台GUI开发框架
- 理解数据类型与索引的关系，操作索引即操作数据
- Python最主要的数据分析功能库，基于Numpy开发

# Python库之图形用户界面

## wxPython: 跨平台GUI开发框架

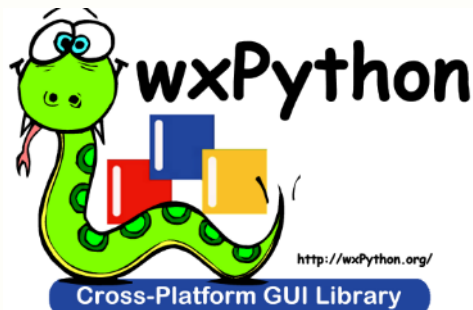
```
import wx
```

```
app = wx.App(False)
```

```
frame = wx.Frame(None, wx.ID_ANY, "Hello World")
```

```
frame.Show(True)
```

```
app.MainLoop()
```



<https://www.wxpython.org>

# Python库之图形用户界面

**PyGObject:** 使用GTK+开发GUI的功能库

- 提供了整合GTK+、WebKitGTK+等库的功能
- GTK+：跨平台的一种用户图形界面GUI框架
- 实例：Anaconda采用该库构建GUI

# Python库之图形用户界面

## PyGObject: 使用GTK+开发GUI的功能库

```
import gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
window = Gtk.Window(title="Hello World")
window.show()
window.connect("destroy", Gtk.main_quit)
Gtk.main()
```



PyGObject

<https://pygobject.readthedocs.io>



# Python库之游戏开发



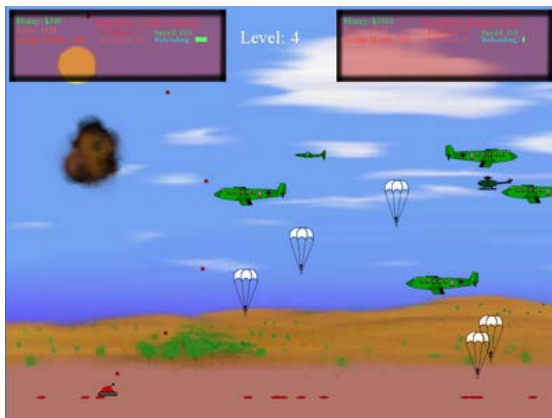
# Python库之游戏开发

## PyGame: 简单的游戏开发功能库

- 提供了基于SDL的简单游戏开发功能及实现引擎
- 理解游戏对外部输入的响应机制及角色构建和交互机制
- Python游戏入门最主要的第三方库

# Python库之游戏开发

**PyGame:** 简单的游戏开发功能库



<http://www.pygame.org>

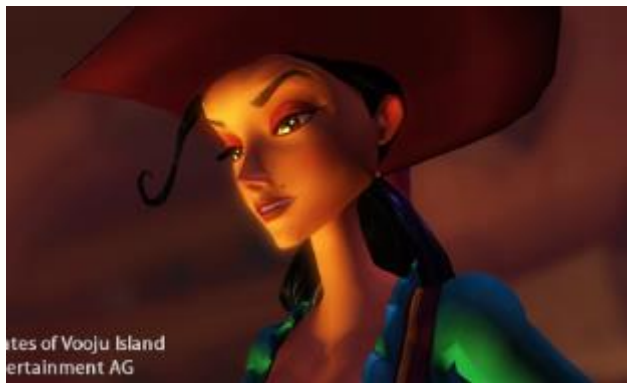
# Python库之游戏开发

## **Panda3D: 开源、跨平台的3D渲染和游戏开发库**

- 一个3D游戏引擎，提供Python和C++两种接口
- 支持很多先进特性：法线贴图、光泽贴图、卡通渲染等
- 由迪士尼和卡尼基梅隆大学共同开发

# Python库之游戏开发

**Panda3D: 开源、跨平台的3D渲染和游戏开发库**



<http://www.panda3d.org>

# Python库之游戏开发

**cocos2d: 构建2D游戏和图形界面交互式应用的框架**

- 提供了基于OpenGL的游戏开发图形渲染功能
- 支持GPU加速，采用树形结构分层管理游戏对象类型
- 适用于2D专业级游戏开发

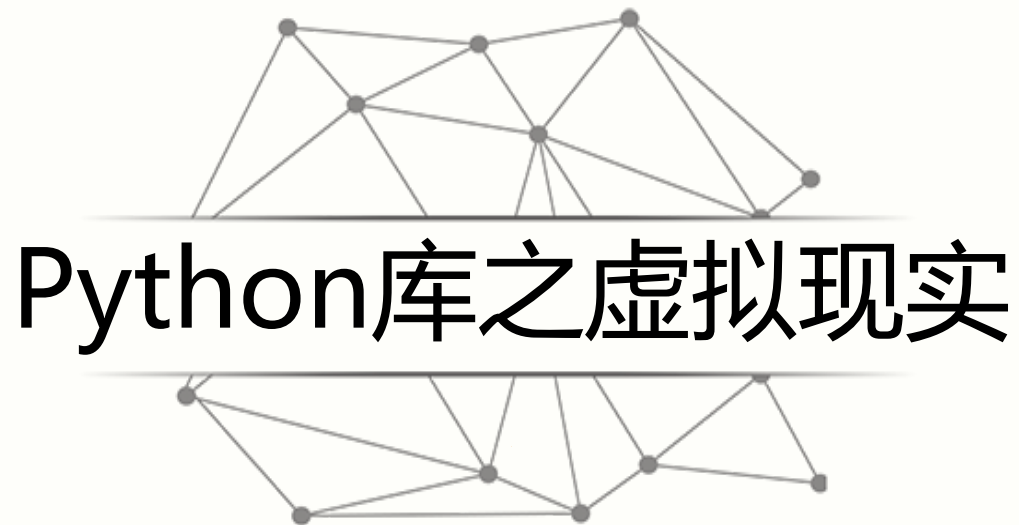
# Python库之游戏开发

**cocos2d:** 构建2D游戏和图形界面交互式应用的框架



**cocos2d**

<http://python.cocos2d.org/>



# Python库之虚拟现实

# Python库之虚拟现实

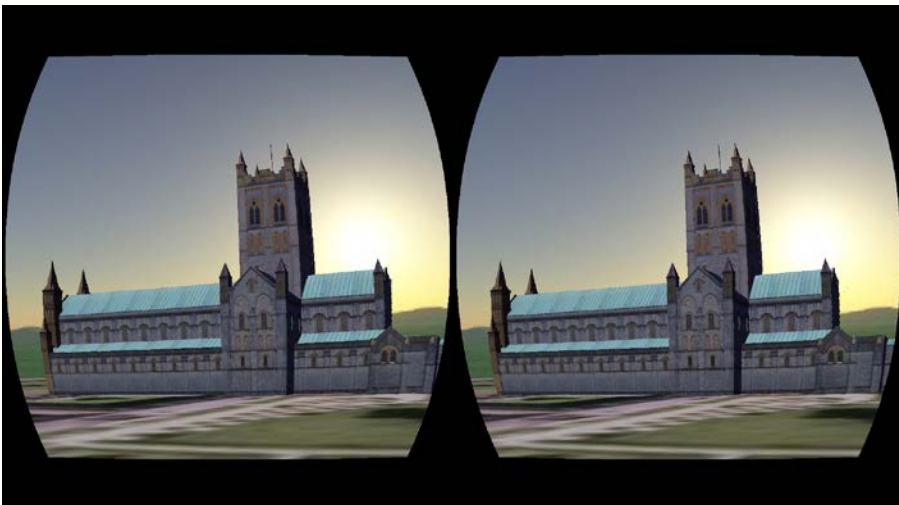
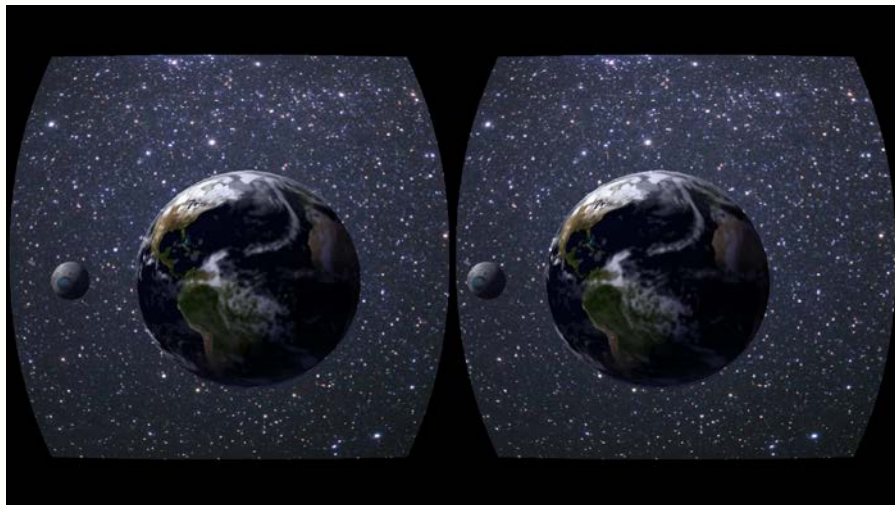
**VR Zero:** 在树莓派上开发VR应用的Python库

- 提供大量与VR开发相关的功能
- 针对树莓派的VR开发库，支持设备小型化，配置简单化
- 非常适合初学者实践VR开发及应用



# Python库之虚拟现实

**VR Zero:** 在树莓派上开发VR应用的Python库



<https://github.com/WayneKeenan/python-vrzero>

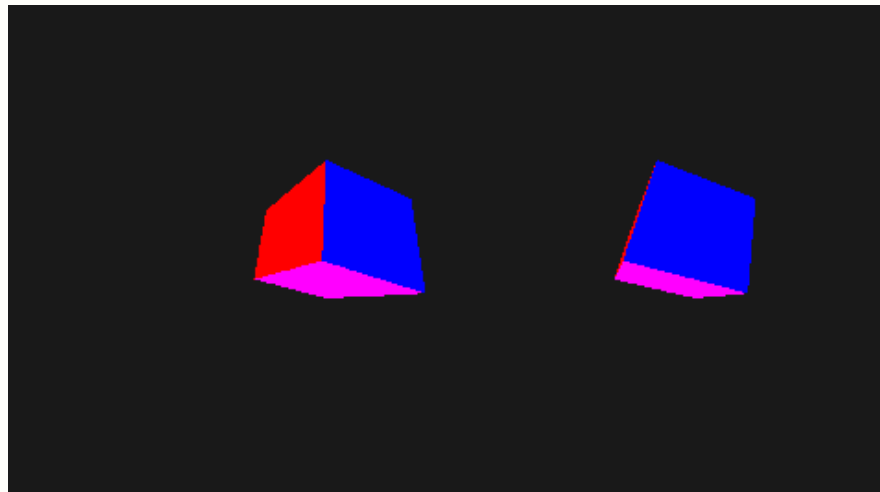
# Python库之虚拟现实

**pyovr: Oculus Rift的Python开发接口**

- 针对Oculus VR设备的Python开发库
- 基于成熟的VR设备，提供全套文档，工业级应用设备
- Python+虚拟现实领域探索的一种思路

# Python库之虚拟现实

**pyovr**: 开发Oculus Rift的Python库



<https://github.com/cmbruns/pyovr>

# Python库之虚拟现实

**Vizard:** 基于Python的通用VR开发引擎

- 专业的企业级虚拟现实开发引擎
- 提供详细的官方文档
- 支持多种主流的VR硬件设备，具有一定通用性

# Python库之虚拟现实

**Vizard:** 基于Python的通用VR开发引擎



<http://www.worldviz.com/vizard-virtual-reality-software>



# Python库之图形艺术

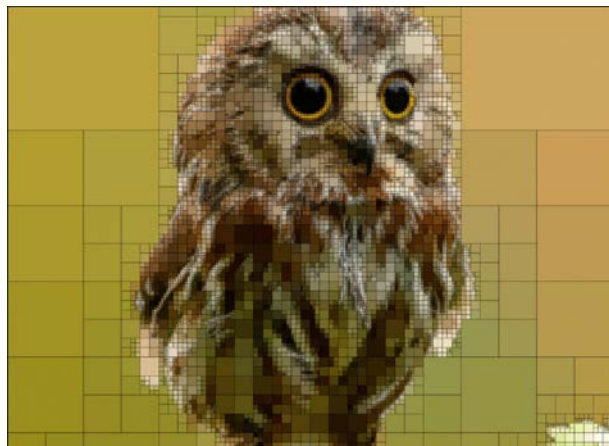
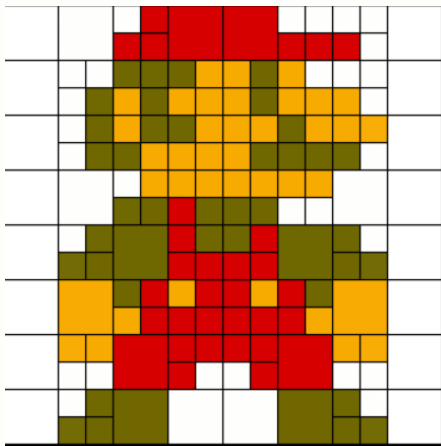
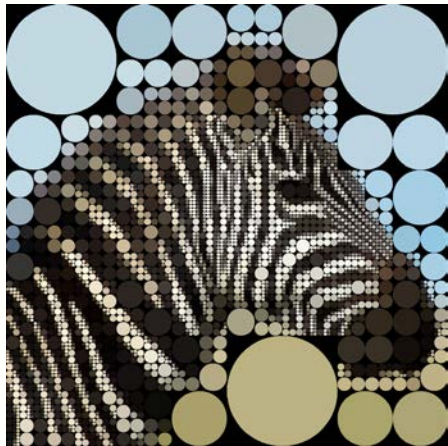
# Python库之图形艺术

## Quads: 迭代的艺术

- 对图片进行四分迭代，形成像素风
- 可以生成动图或静图图像
- 简单易用，具有很高展示度

# Python库之虚拟现实

## Quads: 迭代的艺术



<https://github.com/fogleman/Quads>



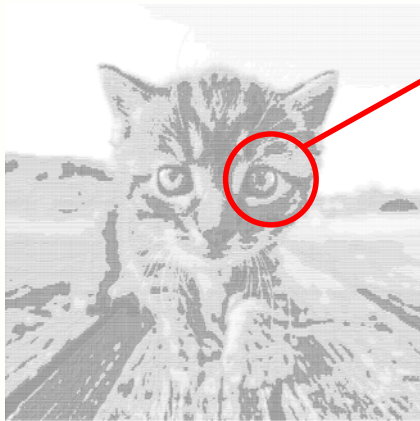
# Python库之图形艺术

**ascii\_art: ASCII艺术库**

- 将普通图片转为ASCII艺术风格
- 输出可以是纯文本或彩色文本
- 可采用图片格式输出

# Python库之虚拟现实

**ascii\_art: ASCII艺术库**



黑白

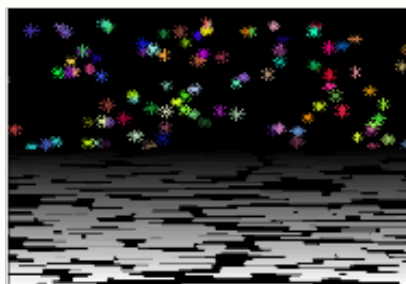
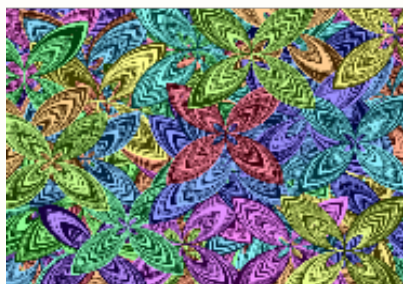
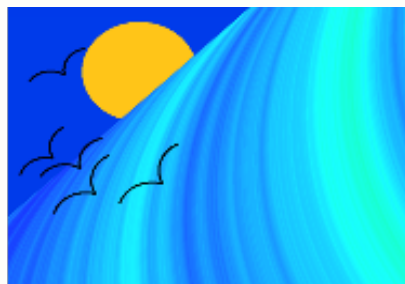
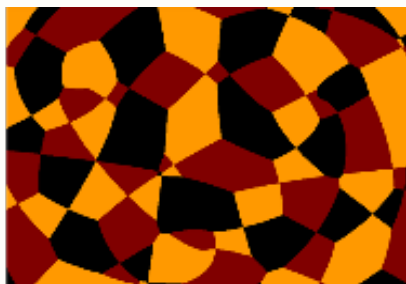


彩色

[https://github.com/jontonsoup4/ascii\\_art](https://github.com/jontonsoup4/ascii_art)

# Python库之图形艺术

**turtle:** 海龟绘图体系



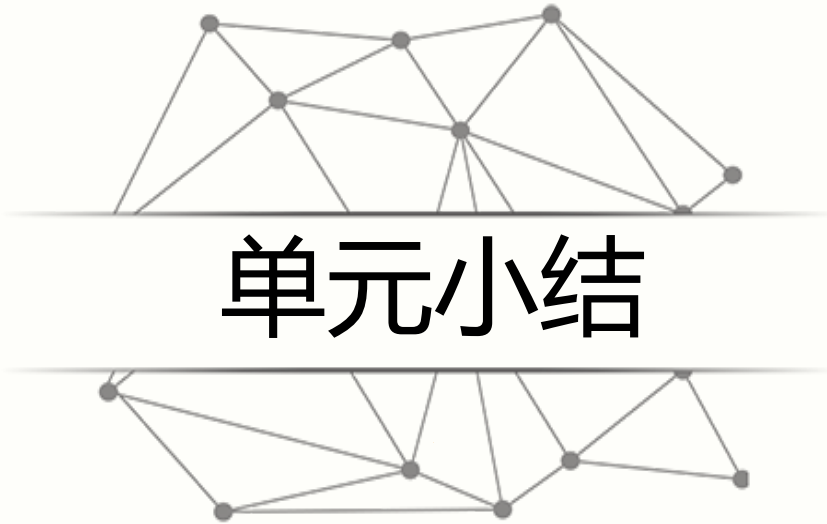
# Python库之图形艺术

**turtle:** 海龟绘图体系



– Random Art

<https://docs.python.org/3/library/turtle.html>



# 单元小结

# 从人机交互到艺术设计

- PyQt5、wxPython、PyGObject
- PyGame、Panda3D、cocos2d
- VR Zero、pyovr、Vizard
- Quads、ascii\_art、turtle



## Python语言程序设计

# 9.5 实例16: 玫瑰花绘制

---





# "玫瑰花绘制"问题分析



# 问题分析

## 玫瑰花绘制



# 问题分析

## 玫瑰花绘制

- **需求：用Python绘制一朵玫瑰花，献给所思所念**
- **输入：你的想象力！**
- **输出：玫瑰花**

# 问题分析

## 玫瑰花绘制

- 绘制机理：turtle基本图形绘制
- 绘制思想：因人而异
- 思想有多大、世界就有多大



# "玫瑰花绘制"实例展示

# 玫瑰花绘制

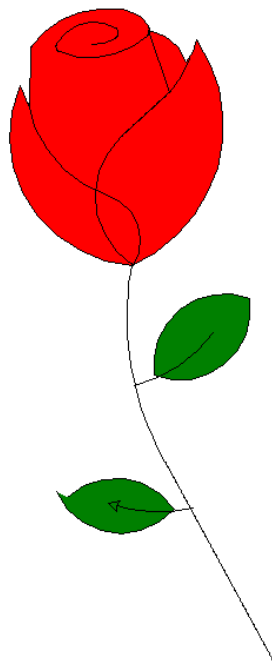
```
# RoseDraw.py
import turtle as t
# 定义一个曲线绘制函数
def DegreeCurve(n, r, d=1):
    for i in range(n):
        t.left(d)
        t.circle(r, abs(d))

# 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()
```



# 玫瑰花绘制

```
# 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))
```

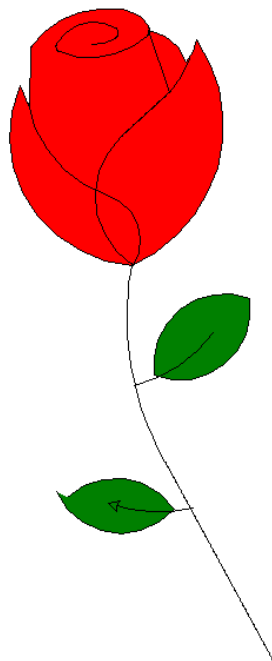
# 初始位置设定

```
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()
```



# 玫瑰花绘制

```
# 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()
```

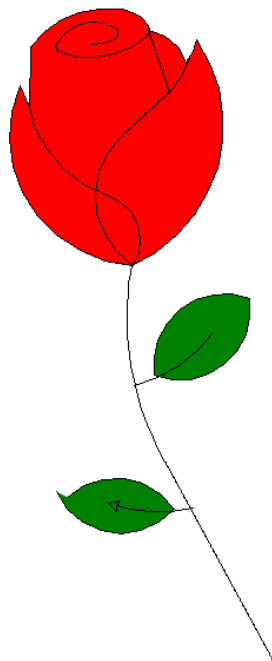
## # 绘制花朵形状

```
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()
```



# 玫瑰花绘制

```
# 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)
```

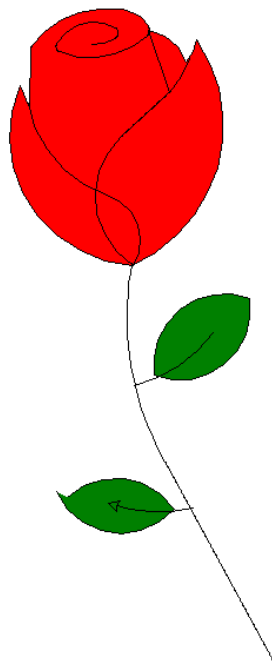
## # 绘制一个绿色叶子

```
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()
```





# 玫瑰花绘制

```
# 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)
```

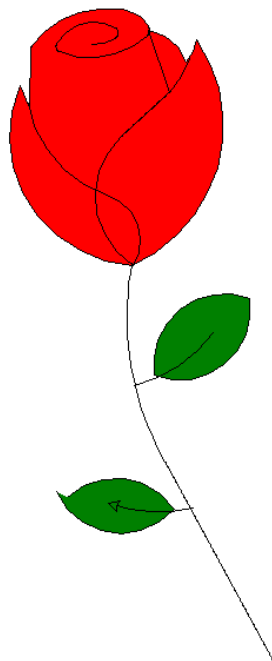
## # 绘制一个绿色叶子

```
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()
```



# 玫瑰花绘制

```
# 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)
```

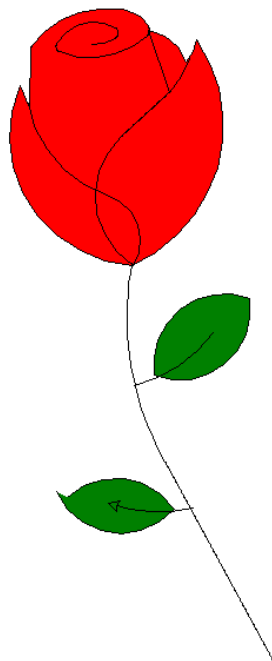
```
t.circle(-500*s, 12)
t.left(140)
t.circle(550*s, 110)
t.left(27)
t.circle(650*s, 100)
t.left(130)
t.circle(-300*s, 20)
t.right(123)
t.circle(220*s, 57)
t.end_fill()

# 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()
```

# 玫瑰花绘制

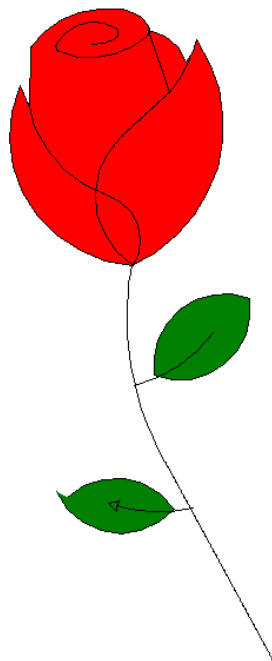
```
# RoseDraw.py
import turtle as t
# 定义一个曲线绘制函数
def DegreeCurve(n, r, d=1):
    for i in range(n):
        t.left(d)
        t.circle(r, abs(d))

# 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()
```





"玫瑰花绘制"举一反三

# 举一反三

## 艺术之于编程，设计之于编程

- 艺术：思想优先，编程是手段
- 设计：想法和编程同等重要
- 工程：编程优先，思想次之

# 举一反三

**编程不重要，思想才重要！**

- **认识自己：明确自己的目标，有自己的思想(想法)**
- **方式方法：编程只是手段，熟练之，未雨绸缪为思想服务**
- **为谁编程：将自身发展与祖国发展相结合，创造真正价值**



