计算机计视觉编程基础

Programming Basics

目录

- Python编程基础
- Numpy数值计算函数库
- Matplotlib绘图库
- OpenCV函数库

IEEE编程语言流行度排名

Rank	Language	Type	Score
1	Python	web desktop system	100.0
2	Java	web mobile desktop	96.3
3	C	mobile desktop system	94.4
4	C++	mobile desktop system	87.5
5	R	desktop	81.5
6	JavaScript	web	79.4
7	C#	web mobile desktop system	74.5
8	Matlab	desktop	70.6
9	Swift	mobile desktop	69.1
10	Go	web desktop	68.0

Python编程基础

- Python语言简介
- 变量与数据类型
- 数据结构
- 表达式
- 函数
- 类

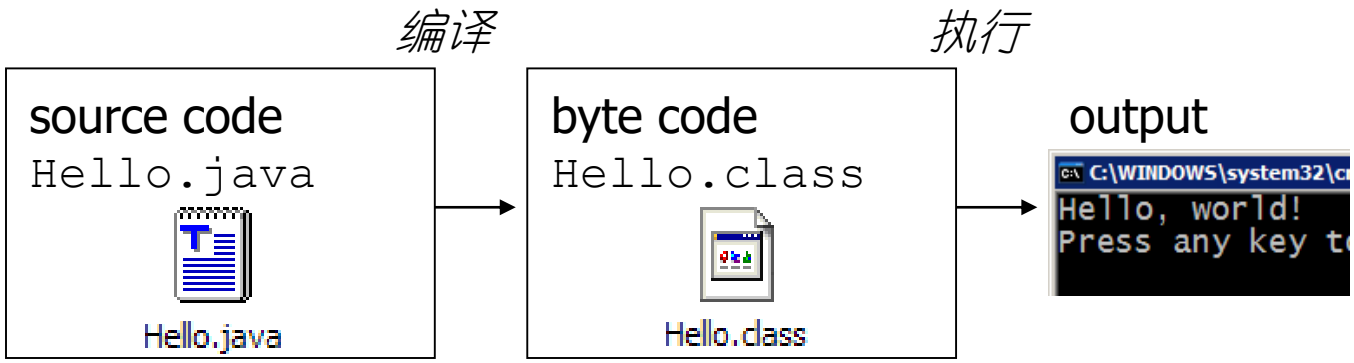
Python语言的历史

- 由荷兰软件工程师Guido van Rossum于1990年代开发

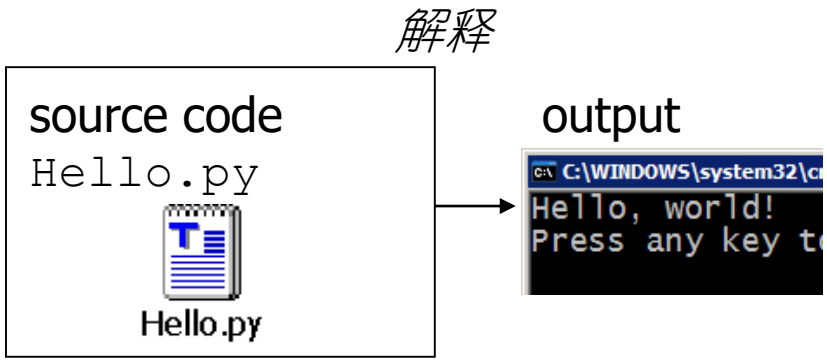
- 根据英国“喜剧界的披头士”喜剧团体Monty Python的名字命名

Python语言的特点

- 传统编程语言如Java，C++需要先编译，再运行



- 和Matlab等脚本语言类似，Python语言可以直接从源文件运行



Python的开发环境

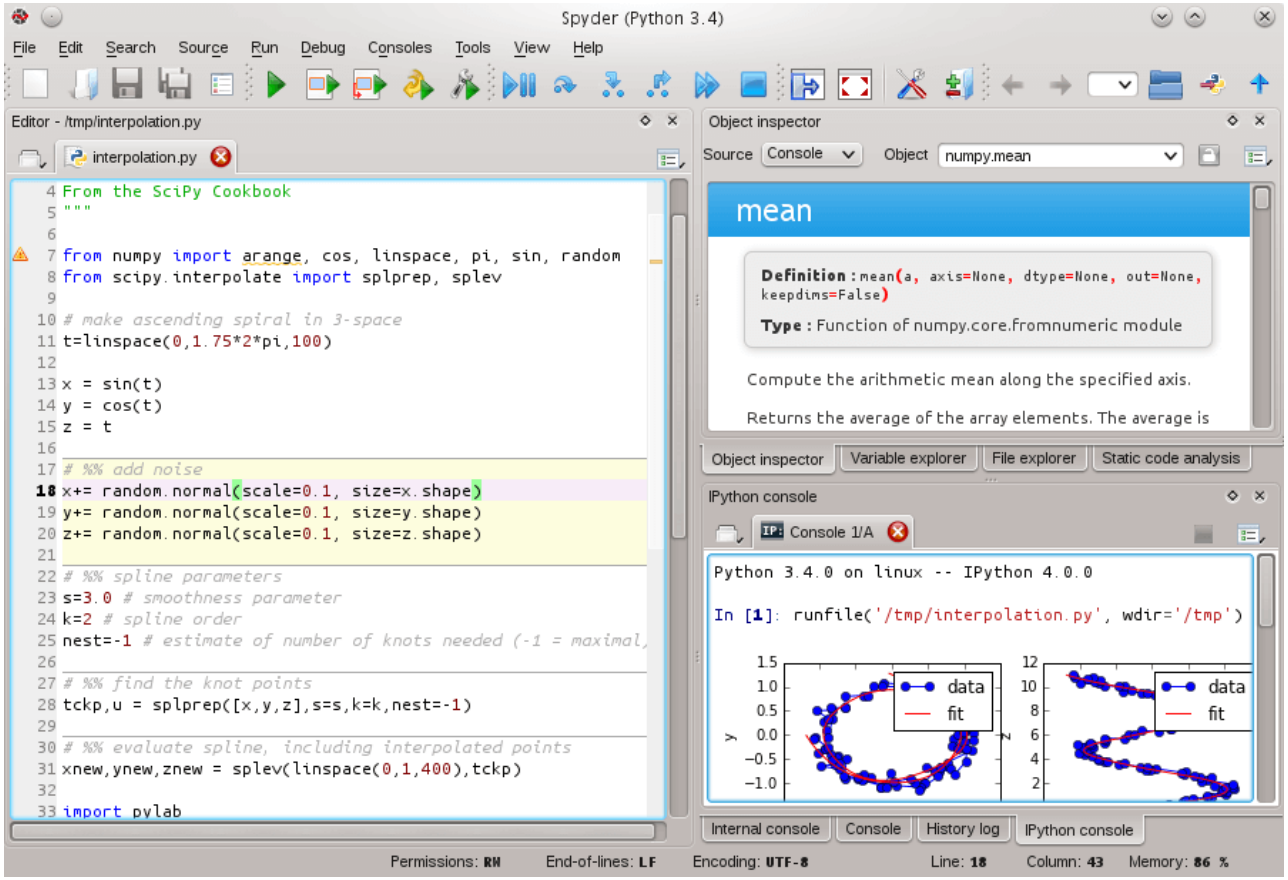
交互式开发环境

- 在命令行输入python进入交互运行模式
- 在>>>后输入要运行的python语句
- 按组合键CONTROL+C退出交互运行模式

```
% python

>>> 3+3
6
```

Python的开发环境

集成开发环境(例如Spyder)：集代码编写、运行、调试于一体



也可以通过命令行运行编写好的python文件

```
% python filename.py
```

Python语言的句法特征

代码的缩进非常重要
- 使用缩进来指示代码块，如同C++中的花括号{}

对变量的第一次赋值创建了该变量
- 不需要像C++那样指定变量的数据类型

=, ==, +, -, *, /, %等数学运算符与C++相同
- +可用于字符串的串连

逻辑运算符为（and, or, not）
- C++中的逻辑运算符（&&, ||, !）

使用#来注释代码
- #之后的同一行代码不被执行

一段简短的python代码

```
x = 34 - 23                  # A comment.
y = “Hello”                  # Another one.
z = 3.45
if z == 3.45 or y == “Hello”:
    x = x + 1
    y = y + “ World”         # String concat.
print x
print y
```

变量与数据类型

变量的命名规则：不能以数字开头

```
bob   Bob   _bob   _2_bob_   bob_2   BoB
```

Python语言的保留字：

```
and, assert, break, class, continue, def, del,
elif, else, except, exec, finally, for, from,
global, if, import, in, is, lambda, not, or, pass,
print, raise, return, try, while
```

基本的数据类型为整型，浮点型，字符串

整型：赋值为整数的变量类型

浮点型：赋值为带小数点的数字的变量类型

字符串：赋值为带“”或‘’的变量类型

Z = 5 / 2　　# Z = ?

Python vs Java/C++

Python

```
x = 5
y = x + 7
z = 3.14

name = “Rishi”


1 == 1 # => True
5 > 10 # => False

True and False # => False
not False # => True
```

变量不需要指定类型

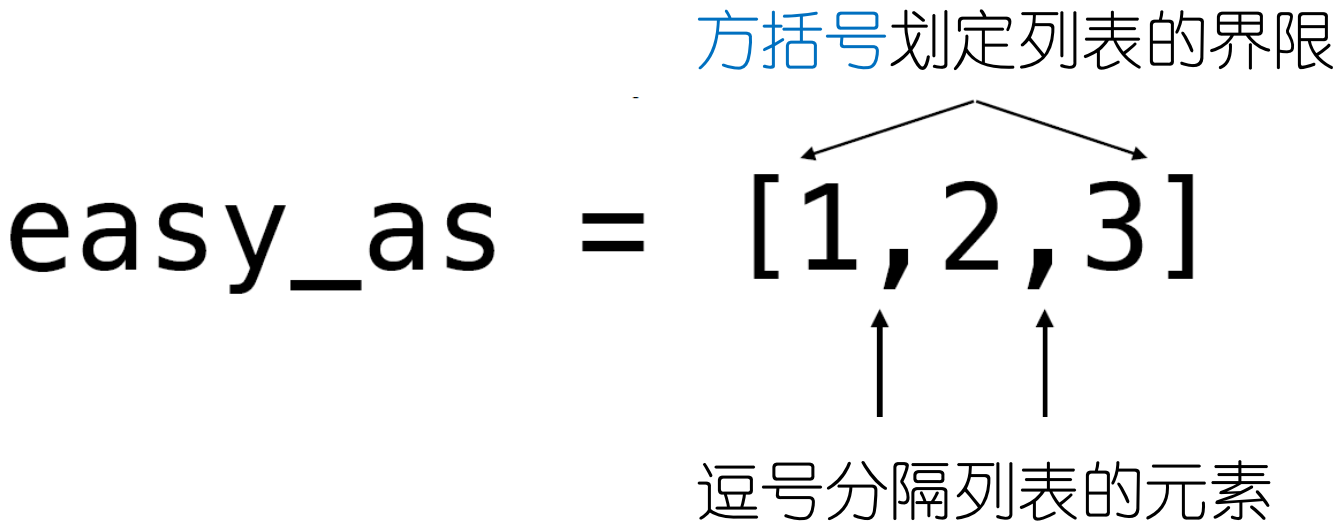
Java/C++

```
int x = 5;
int y = x + 7;
double z = 3.14;

String name = “Rishi”; // Java
string name(“Rishi”);  // C++

1 == 1 # => true
5 > 10 # => false

true && false # => false
!(false) # => true
```

数据结构

- 列表（List）
- 元组（Tuple）
- 字典（Dictionary）

列表（List）



列表的创建

```
# Create a new list   # 创建列表
empty = []
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]
                                  # 列表元素的混合类型
# Lists can contain elements of different types
mixed = [4, 5, "seconds"]

# Append elements to the end of a list# 添加列表元素
numbers.append(7)    # numbers == [2, 3, 5, 7]
numbers.append(11)   # numbers == [2, 3, 5, 7, 11]
```

```
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]
```

列表的访问

```
# Access elements at a particular index
numbers[0]  # => 2
numbers[-1]  # => 11

# You can also slice lists - the same rules apply
letters[:3]  # => ['a', 'b', 'c']
numbers[1:-1]  # => [3, 5, 7]

# Lists really can contain anything - even other lists!
x = [letters, numbers]
x  # => [['a', 'b', 'c', 'd'], [2, 3, 5, 7, 11]]
x[0]  # => ['a', 'b', 'c', 'd']
x[0][1]  # => 'b'
x[1][2:]  # => [5, 7, 11]
```

访问单个元素

访问多个元素

访问多层列表

列表的灵活创建

例子：创建一个列表是已知列表对应元素的平方值

Input: nums = [1, 2, 3, 4, 5]

Goal: sq_nums = [1, 4, 9, 16, 25]

```
# 基本创建方法
sq_nums = []
for n in nums:
    sq_nums.append(n**2)

# 灵活创建方法
sq_nums = [n ** 2 for n in nums]
```

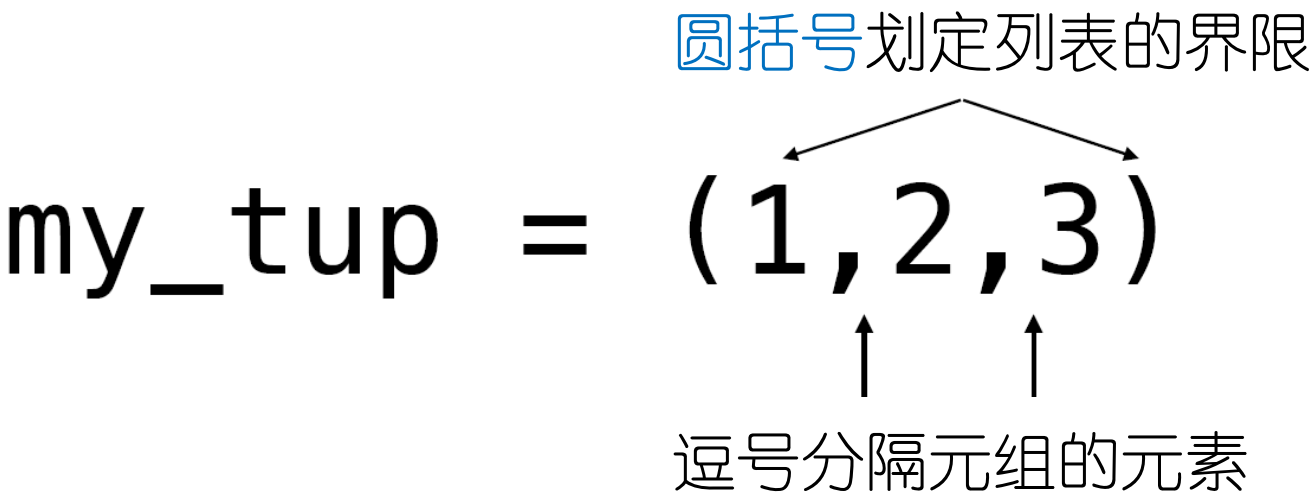
列表的灵活创建

```
Template:
    new_list = [f(x) for x in iterable]
```

将原列表的所有单词转换成大写

```
words = ['hello', 'this', 'is', 'python']
caps  = [word.upper() for word in words]
```

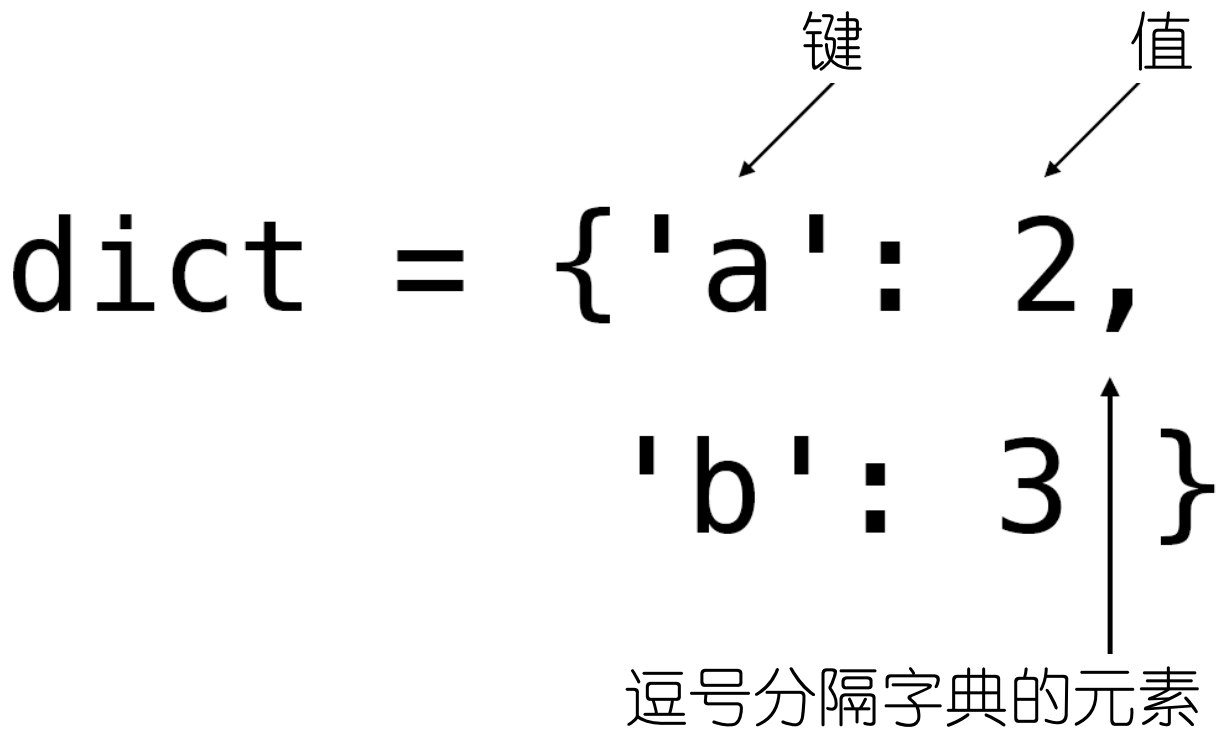
元组(Tuple)



与列表主要的区别在于元组的元素不可更改

```
my_tup[0] = 5 => Error!
```

字典（Dictionary）



字典

```
# 字典的创建
d = {"one": 1, "two": 2, "three": 3}
# 字典的大小
len(d.keys())  # => 3

# 通过键 访问值
print d['one']   # => 1
print d['five']  # => ERROR!
# 字典元素的添加
d['five'] = 5    # => OK, creates new key

d.keys()        # iterator over k # 字典的键列表
d.values()      # iterator over v # 字典的值列表
d.items()       # iterator over (k, v) pairs
```

表达式

- 逻辑表达式
- if…else条件表达式
- for迭代表达式

逻辑表达式

布尔值 True：真值 False：假值

- 生成布尔值的关系运算符：

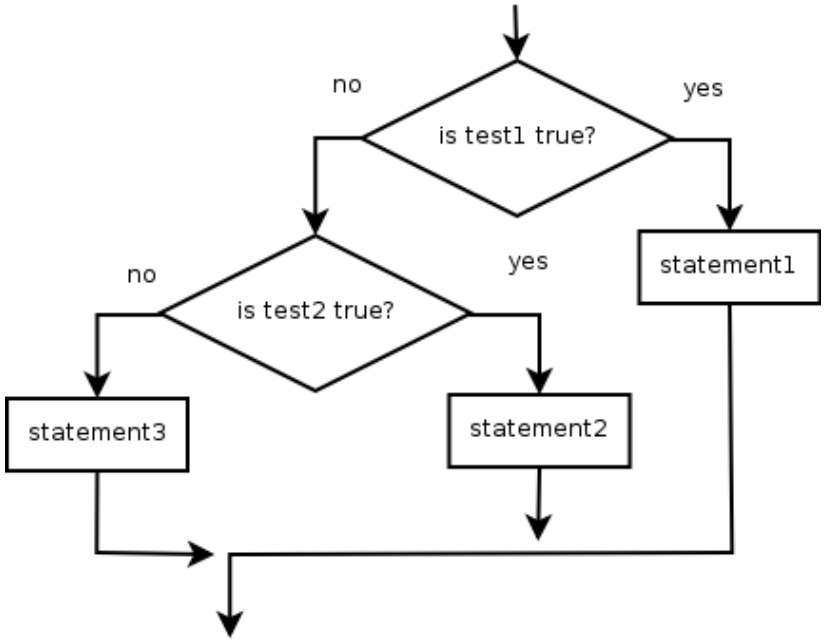
Operator	Meaning	Example	Result
==	相等	1 + 1 == 2	True
!=	不相等	3.2 != 2.5	True
<	小于	10 < 5	False
>	大于	10 > 5	True
<=	小于等于	126 <= 100	False
>=	大于等于	5.0 >= 5.0	True

- 组合布尔值的逻辑运算符：

Operator	Example	Result
and （与）	9 != 6 and 2 < 3	True
or （或）	2 == 3 or -1 < 5	True
not （非）	not 7 > 0	False

if...else条件表达式

```
if condition:
    statement1
elif condition:
    statement2
else:
    statement3
```




```
if x == 3:
    print "X equals 3."
elif x == 2:
    print "X equals 2."
else:
    print "X equals something else."
print "This is outside the 'if'."
```

注意事项：

- 每个条件分支后有一个冒号(:)
- 使用缩进标识条件分支下的执行代码块

for循环迭代

```
for <item> in <collection>:
    <statements>
```

对列表、字典等数据结构的元素直接进行迭代

- 不需要像C++那样使用索引
- 对字典进行迭代时，获取的是键(key)

for迭代访问列表

```
mylist = ['a', 'b', 'c']
for item in mylist:
    print item
```

for迭代访问字典

```
dict = {'a': 10, 'b': 15}
for key in dict:
    print key, dict[key]
```

for循环迭代

对索引进行迭代

- 通过函数range()指定索引的列表

```
# 函数range()的简单使用方法
for i in range(4):
    print i,                                    # 0 1 2 3

# 函数range()的复杂使用方法
for i in range(1, 10, 2):
    print i,                                    # 1 3 5 7 9

# 通过索引对列表进行迭代
mylist = ['a', 'b', 'c']
for i in range(len(mylist)):
    print i, mylist[i]                          #  0 'a'
                                                #  1 'b'
                                                #  2 'c'
```

函数(function)

```
def fn_name(param1, param2):
    value = do_something()
    return value
```

- def开启了一个函数的定义
- return是可选的
- 参数没有明确的类型

```
def isEven(num):
    return (num % 2 == 0)

myNum = 100
if isEven(myNum):
    print str(myNum) + " is even"
```

函数与列表的组合

```
def isEven(num):
    return (num % 2 == 0)
```

```
Template:
    new_list = [f(x) for x in iterable]
```

例子：

```
numbers  = [5, 18, 7, 9, 2, 4, 0]
isEvens  = [isEven(num) for num in numbers]
```

输出结果：

```
## isEvens = [False, True, False, False, True, True, True]
```

文件读取

- 读取整个文件:

variableName = open("***filename***").read()

例:

```
file_text = open("bankaccount.txt").read()
```

- 按行读取文件:

```
for line in open("filename").readlines():
    statements
```

例:

```
count = 0
for line in
open("bankaccount.txt").readlines():
    count = count + 1
print "The file contains", count, "lines."
```

类（class）

```
class Predictor(object):
    def __init__(self, nIters, name):
        self.nIters = nIters
        self.name = name

    def predict(self, start):
        raise NotImplementedError("Predictor should not be instantiated")
```

成员变量

成员函数

__init__在创建类的对象时进行初始化

类的继承与使用

```
class MonteCarloPredictor(Predictor):
    def predict(self, start):
        ## Do some stuff
        for x in self.nIters:
            # do some stuff
            pass
```

父类

使用注意事项：

- 类的内部访问成员变量要添加self.的前缀
- 类的外部访问成员变量或函数则使用类对象的名字作为前缀

```
myPredictor = MonteCarloPredictor(nIters=1000, name='myPred')
myPredictor.predict(pose)
```

导入模块(import module)

- 模块本身也是python文件，包含事先定义好的函数、类等信息
- 通过import导入

```
import math
import random
```


模块名

- 导入后可以通过模块名访问模块中的内容

```
math.exp(0.5)
random.random()
```

- 从模块中导入指定的函数，可以直接访问

```
from math import exp
from random import random
```

数学模块

```
from math import *
```

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

Numpy数值计算库

- Numpy是python语言中进行数值计算的基础函数库
- 处理对象是N维的数字阵列
- 是其它复杂软件库（Scipy, OpenCV）的基本组成部分
- 开源

使用示例-创建矩阵

```
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 6]])
print A
# [[1 2 3]
#  [4 5 6]]

Af = np.array([1, 2, 3], float)
```

使用示例-创建矩阵

```
np.arange(0, 1, 0.2)
# array([ 0. ,  0.2,  0.4,  0.6,  0.8])

np.linspace(0, 2*np.pi, 4)
# array([ 0.0,  2.09,  4.18, 6.28])

A = np.zeros((2,3))
# array([[ 0.,  0.,  0.],
#        [ 0.,  0.,  0.]])
# np.ones, np.diag
A.shape
# (2, 3)
```

使用示例-创建随机矩阵

```
np.random.random((2,3))
# array([[ 0.78084261,  0.64328818,  0.55380341],
#        [ 0.24611092,  0.37011213,  0.83313416]])

a = np.random.normal(loc=1.0, scale=2.0, size=(2,2))
# array([[ 2.87799514,  0.6284259 ],
#        [ 3.10683164,  2.05324587]])

np.savetxt("a_out.txt", a)
# save to file
b = np.loadtxt("a_out.txt")
# read from file
```

使用示例-矩阵对象的属性

```
a = np.arange(10).reshape((2,5))

a.ndim     # 2 dimension
a.shape    # (2, 5) shape of array
a.size     # 10 # of elements
a.T        # transpose
a.dtype    # data type
```

第一维度
维度
元素个数
转置
数据类型

使用示例-访问矩阵的部分元素

```
a = np.random.random((4,5))
```
创建一个4行5列的矩阵

```
a[2, :]
# third row, all columns
```
第3行的所有元素

```
a[1:3]
# 2nd, 3rd row, all columns
```
第2，3行的所有元素

```
a[:, 2:4]
# all rows, columns 3 and 4
```
第3，4列的所有元素

使用示例-矩阵基本运算

```
a = np.arange(4)
# array([0, 1, 2, 3])

b = np.array([2, 3, 2, 4])

a * b  # array([ 0,  3,  4, 12])
b - a  # array([2, 2, 0, 1])

c = [2, 3, 4, 5]
a * c  # array([ 0,  3,  8, 15])
```

使用示例-矩阵乘法

```
np.dot(A, B)
# array([[ 2.,  2.,  2.],
#        [ 2.,  2.,  2.],
#        [ 2.,  2.,  2.]])

np.dot(B, A)
# array([[ 3.,  3.],
#        [ 3.,  3.]])

np.dot(B.T, A.T)
# array([[ 2.,  2.,  2.],
#        [ 2.,  2.,  2.],
#        [ 2.,  2.,  2.]])

np.dot(A, B.T)
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# ValueError: shapes (3,2) and (3,2) not aligned:
# ... 2 (dim 1) != 3 (dim 0)
```

```
A = np.ones((3, 2))
# array([[ 1.,  1.],
#        [ 1.,  1.],
#        [ 1.,  1.]])
A.T
# array([[ 1.,  1.,  1.],
#        [ 1.,  1.,  1.]])

B = np.ones((2, 3))
# array([[ 1.,  1.,  1.],
#        [ 1.,  1.,  1.]])
```

Numpy中的线性代数子模块

`import numpy.linalg` → 线性代数子模块

`qr`	Computes the QR decomposition
`cholesky`	Computes the Cholesky decomposition
`inv(A)`	Inverse 求矩阵的逆矩阵
`solve(A,b)`	Solves $Ax = b$ for A full rank 求解线性方程组
`lstsq(A,b)`	Solves $\arg\min_x \|Ax - b\|_2$ 最小二乘法
`eig(A)`	Eigenvalue decomposition 矩阵的特征值分解
`eig(A)`	Eigenvalue decomposition for symmetric or hermitian
`eigvals(A)`	Computes eigenvalues.
`svd(A, full)`	Singular value decomposition 矩阵的奇异值分解
`pinv(A)`	Computes pseudo-inverse of A 求矩阵的广义逆矩阵

Numpy中的随机采样子模块

```
import numpy.random
```
→ 随机采样子模块

`rand(d0,d1,...,dn)`	Random values in a given shape 指定维度的随机矩阵
`randn(d0, d1, ...,dn)`	Random standard normal 服从高斯分布的的随机矩阵
`randint(lo, hi, size)`	Random integers [lo, hi) 指定了取值范围的随机数
`choice(a, size, repl, p)`	Sample from a 从序列a中随机采样部分元素
`shuffle(a)`	Permutation (in-place) 随机打乱序列a中的元素排列
`permutation(a)`	Permutation (new array)

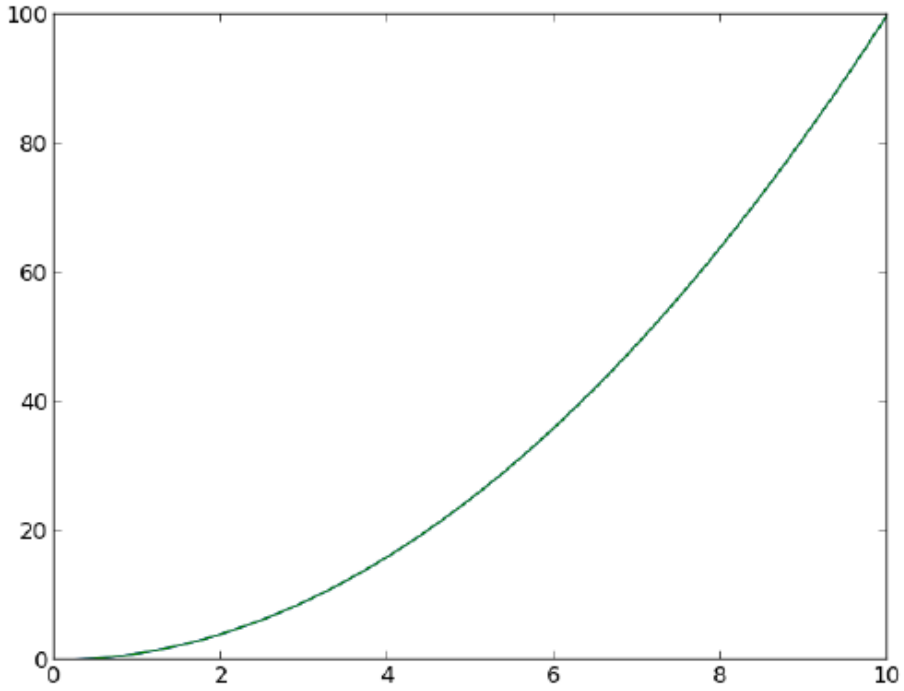
Matplotlib绘图库

- Matplotlib是python语言中进行绘图的函数库
- 与Numpy库配合使用
- 语法和Matlab比较接近

Scatter plot（散点图绘制）

- 散点图用于表示两个数值变量之间的联系或依存趋势

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.power(x, 2)
plt.plot(x, y)
plt.show()
```

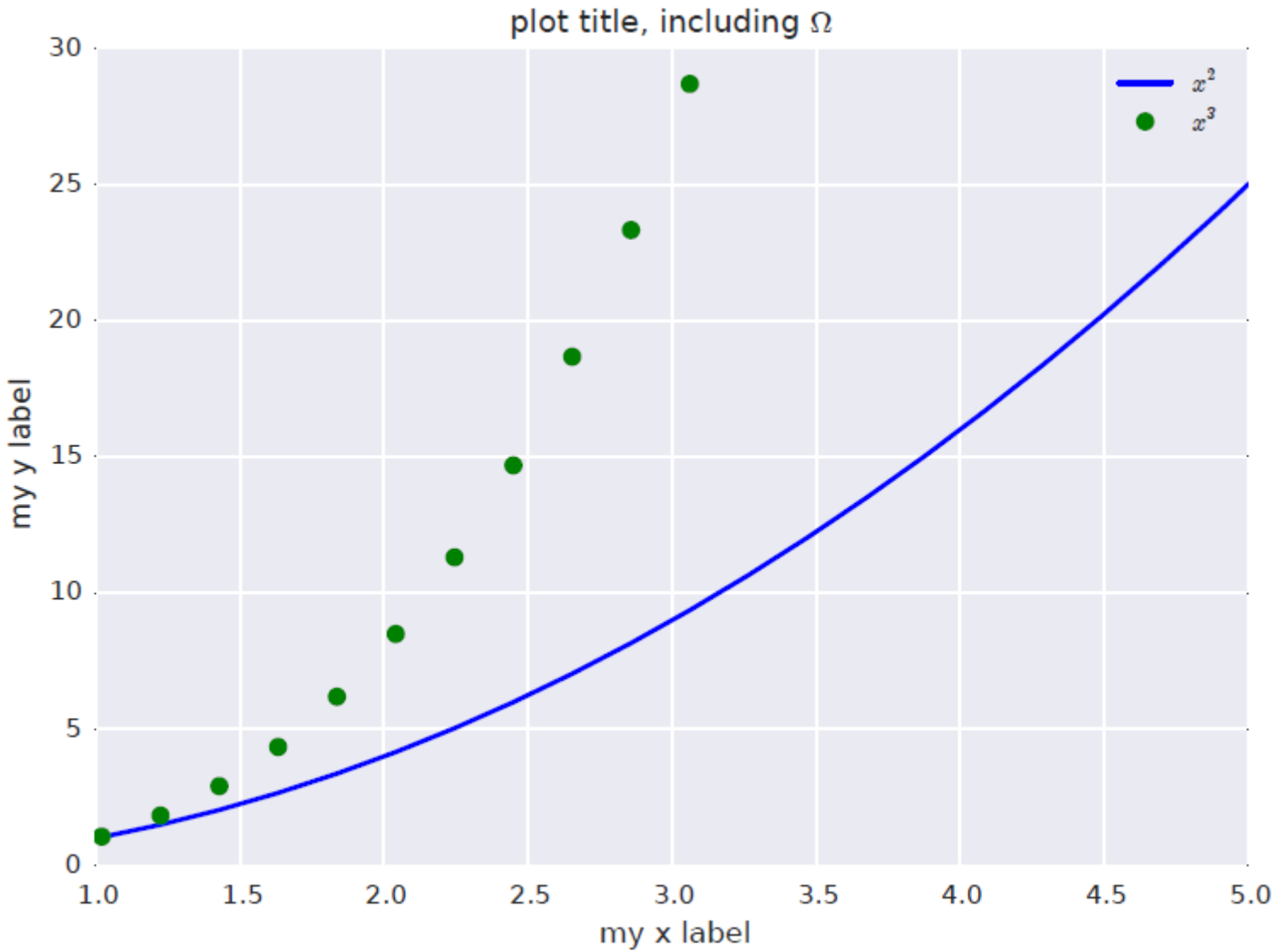


散点图

- 为散点图添加多条曲线与说明信息

```
x = np.linspace(0, 10, 50)
y1 = np.power(x, 2)
y2 = np.power(x, 3)

plt.plot(x, y1, 'b-', label='$x^2$')
plt.plot(x, y2, 'go', label='$x^3$')
plt.xlim((1, 5))
plt.ylim((0, 30))
plt.xlabel('my x label')
plt.ylabel('my y label')
plt.title('plot title, including $\Omega$')
plt.legend()

plt.savefig('line_plot_plus2.pdf')
```

散点图

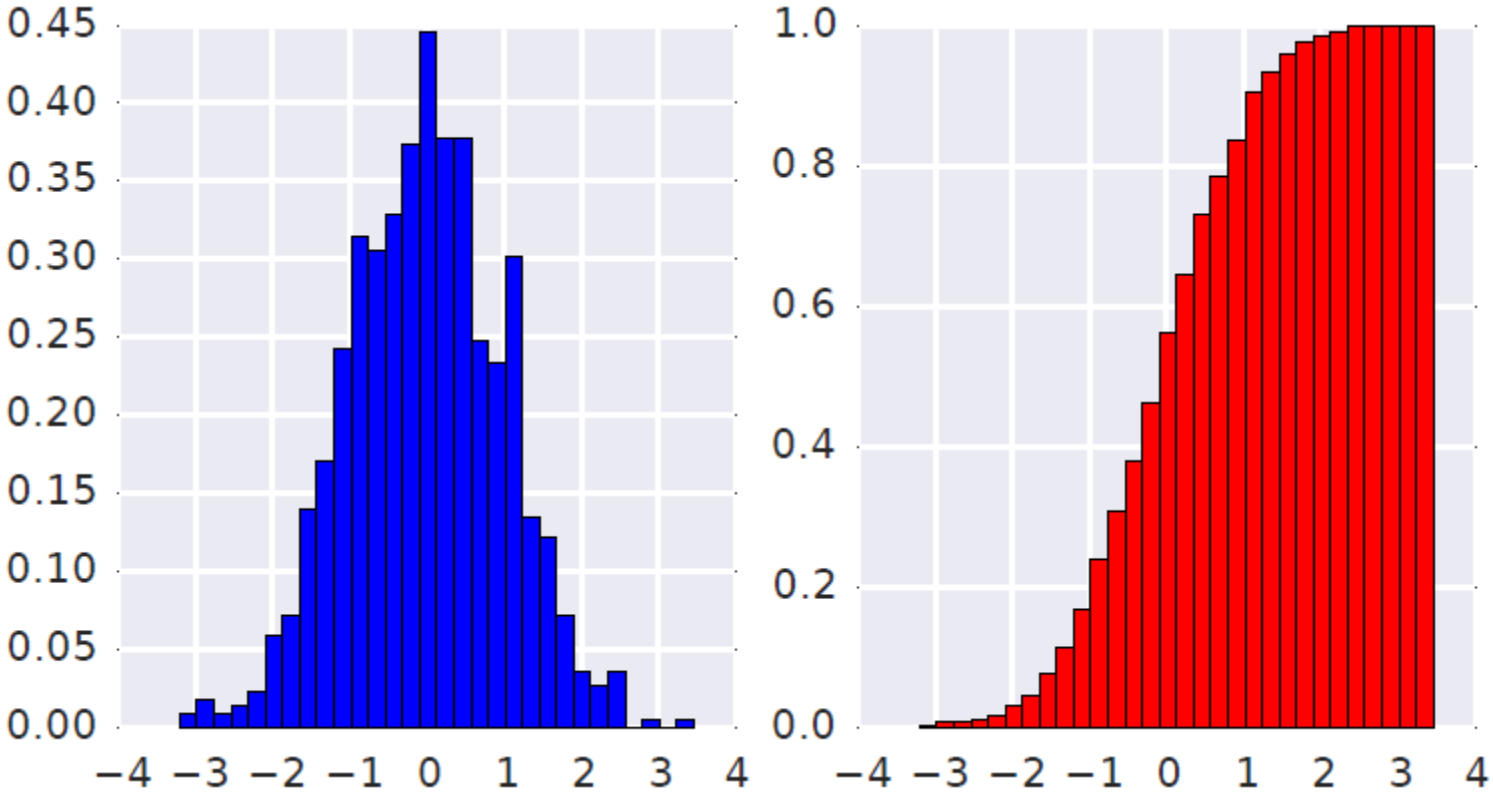

plot title, including Ω
my y label
my x label
x^2
x^3
0
5
10
15
20
25
30
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

直方图(histogram)

- 直方图是反映数据概率分布的总结性图示

```
data = np.random.randn(1000)

f, (ax1, ax2) = plt.subplots(1, 2, figsize=(6,3))

# histogram (pdf)
ax1.hist(data, bins=30, normed=True, color='b')

# empirical cdf
ax2.hist(data, bins=30, normed=True, color='r',
         cumulative=True)

plt.savefig('histogram.pdf')
```

直方图

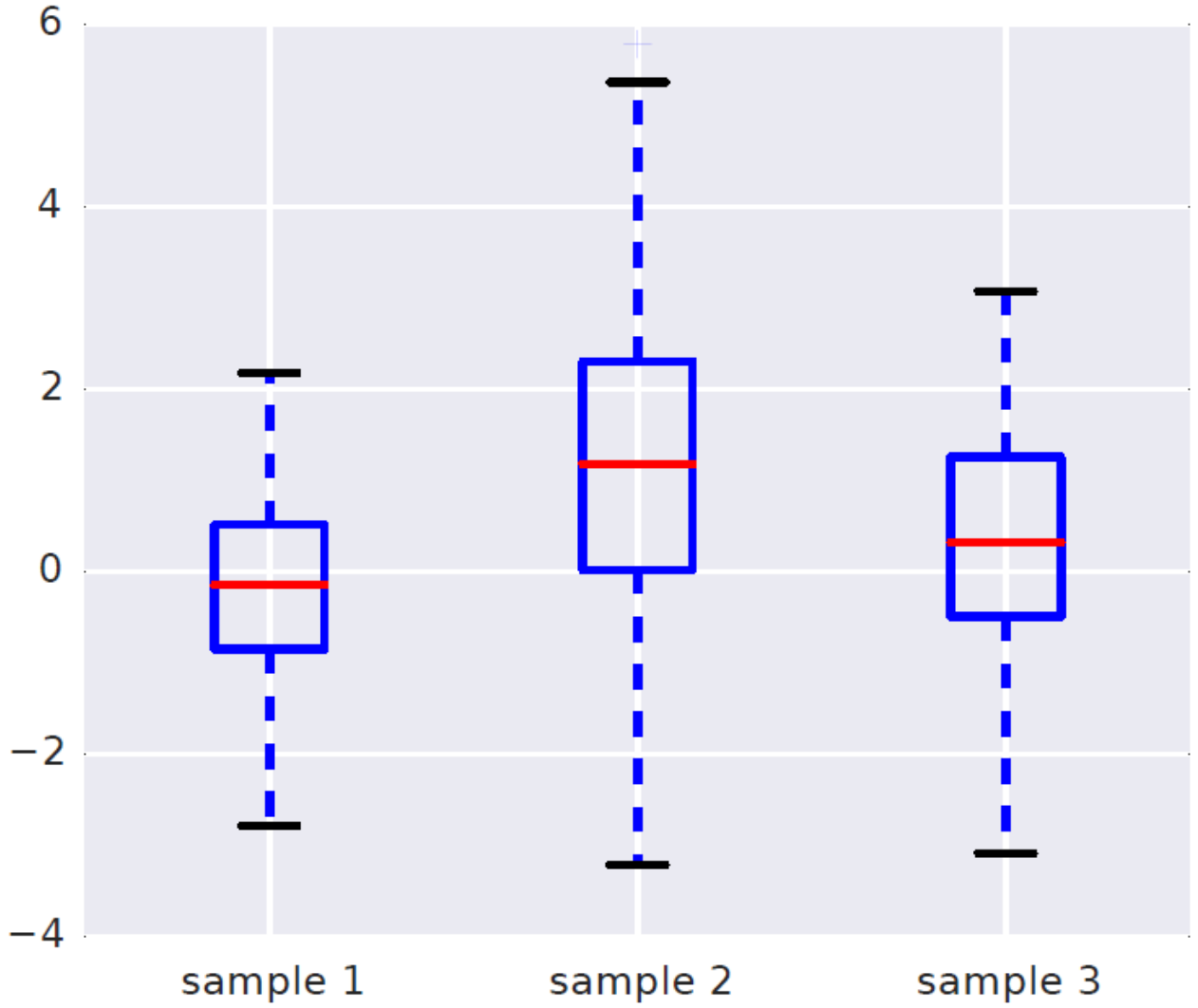

0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4 −3 −2 −1 0 1 2 3 4
1.0
0.8
0.6
0.4
0.2
0.0
−4 −3 −2 −1 0 1 2 3 4

箱型图绘制（box plot）

- **箱型图**是显示一组数据分散情况的统计图

```
samp1 = np.random.normal(loc=0., scale=1., size=100)
samp2 = np.random.normal(loc=1., scale=2., size=100)
samp3 = np.random.normal(loc=0.3, scale=1.2, size=100)

f, ax = plt.subplots(1, 1, figsize=(5,4))

ax.boxplot((samp1, samp2, samp3))
ax.set_xticklabels(['sample 1', 'sample 2', 'sample 3'])
plt.savefig('boxplot.pdf')
```

箱型图

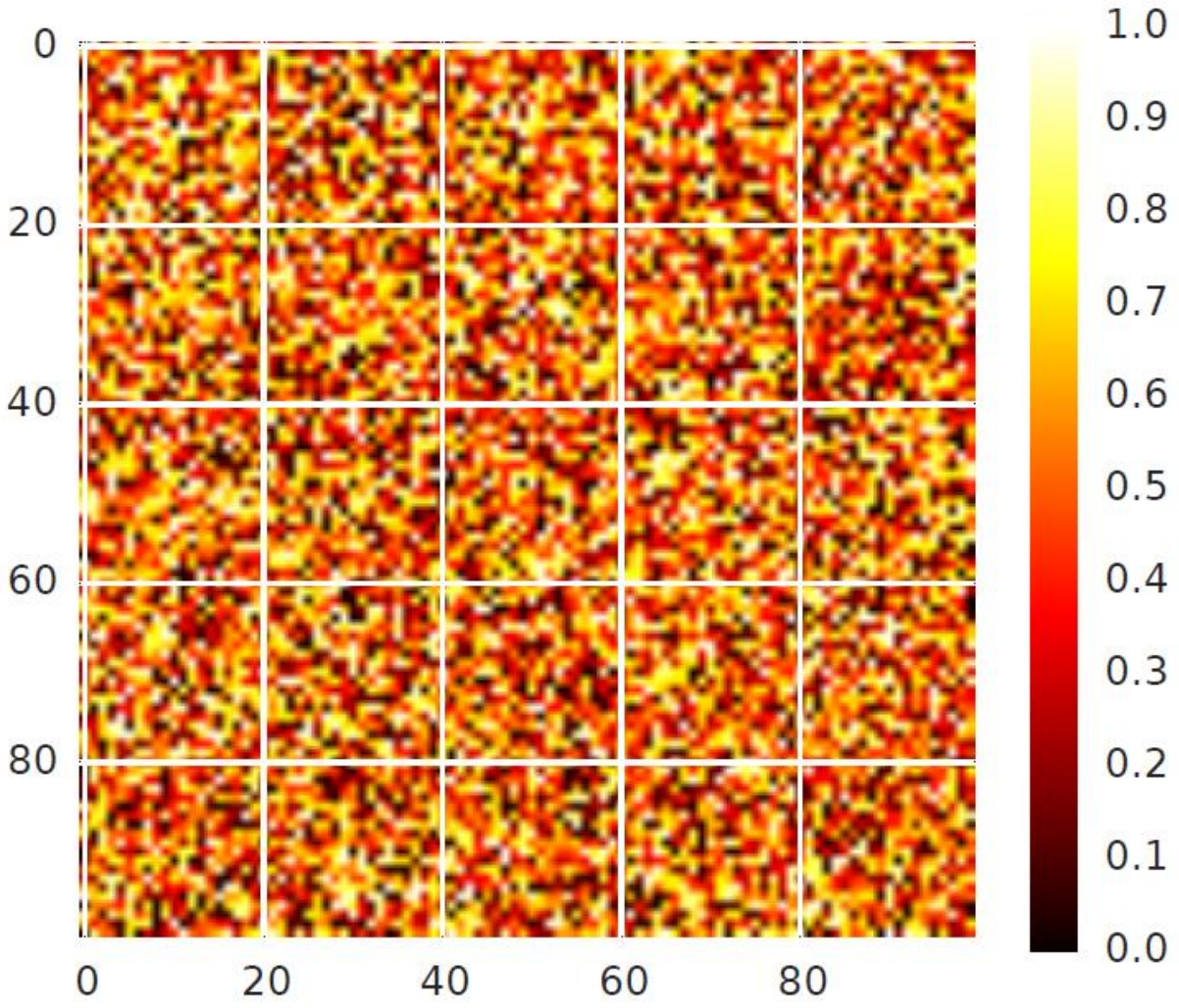


图像绘制（image plot）

- **Image plot**用于绘制一个二维矩阵数据

```
A = np.random.random((100, 100))

plt.imshow(A)
plt.hot()
plt.colorbar()

plt.savefig('imageplot.pdf')
```

图像绘制



Anaconda

Anaconda是一个针对python/R语言的软件发布包

- 开源
- 跨平台支持： Windows, Linux, Mac OS
- 包含了Numpy, Scipy, Matplotlib等众多流行的python软件库
- 包含了python集成开发环境-Spyder
- 包含了软件管理工具-conda

Anaconda的安装-超级简单！

- 从Anaconda官网下载安装文件
- 执行安装文件并按指示操作

OpenCV计算机视觉库

- OpenCV: Open Source Computer Vision Library
  - 创建于1999年，开源的计算机视觉软件库
- 跨平台：Windows，Linux，Mac OS
- 支持移动端：Android, iPhone
- 支持多种语言：C/C++, Python

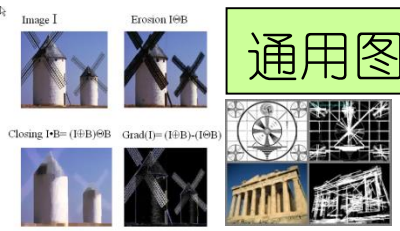
OpenCV 概览:

> 500 functions

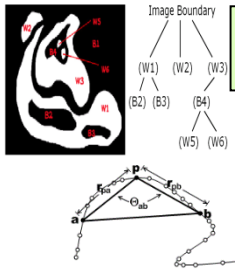
Robot support

opencv.willowgarage.com

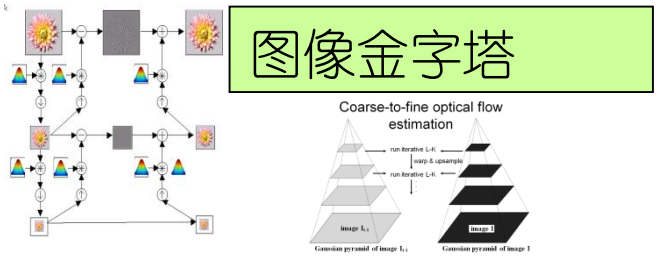
通用图像处理函数



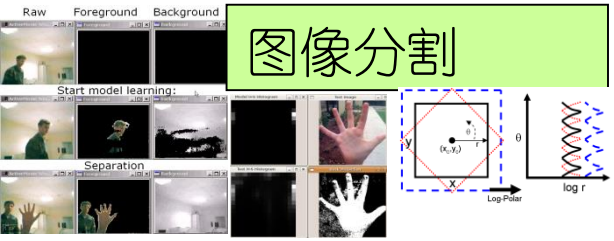
几何描述



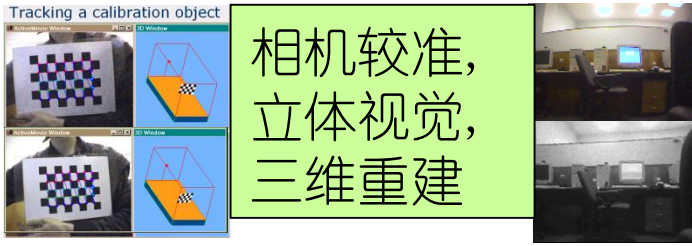
图像金字塔



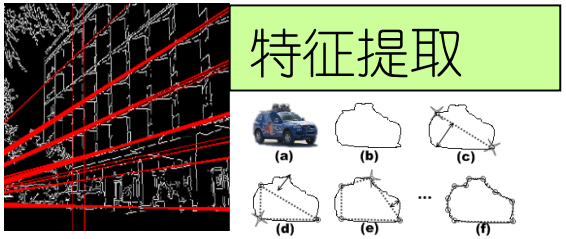
图像分割



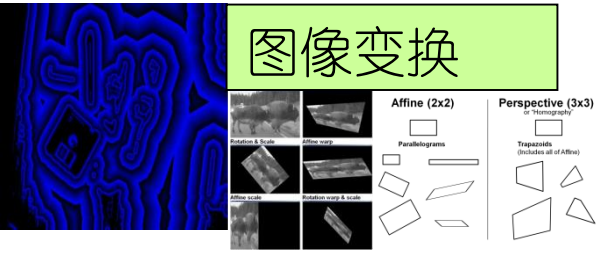
相机较准, 立体视觉, 三维重建



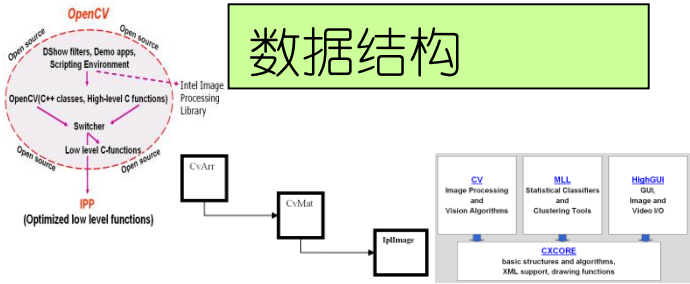
特征提取



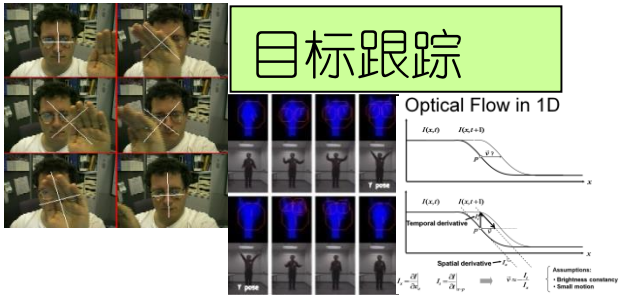
图像变换



数据结构

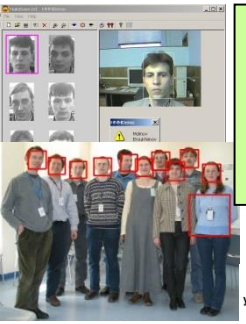


目标跟踪

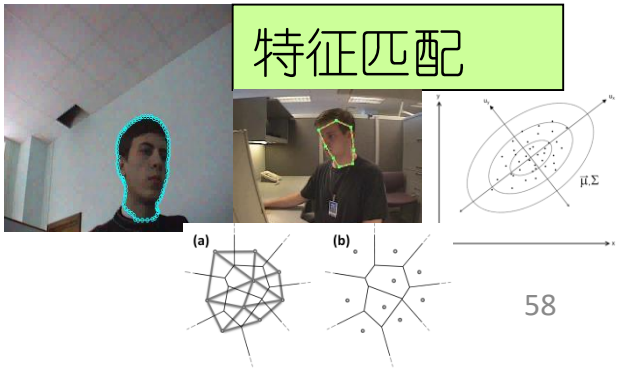


机器学习:

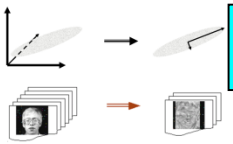
- 物体检测,
- 人脸识别

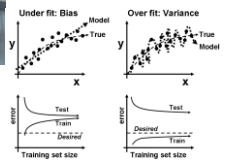


特征匹配



矩阵运算





OpenCV的主要子模块

- **core** Basic data structures 基本数据结构
- **imgproc** Image processing, filter, transformation 图像处理
- **highgui** GUI, codecs, image/video capturing 图像视频读取显示
- **calib3d** Camera calibration, 3D reconstruction 相机较准，三维重建
- **feature2d** 2D feature (detector, descriptor, matching) 特征提取
- **video** Motion tracking, foreground extraction 视频处理
- **objdetect** Object detection (face, people) 物体检测
- **ml** Machine learning library 机器学习
- **gpu** GPU acceleration gpu加速

OpenCV安装-Windows

- 使用预编译的安装文件
  - 便捷但不灵活
- 从源文件编译安装
  1. 下载代码
  2. 安装集成开发环境
  3. 安装CMake，并配置和生成Makefile
  4. 使用IDE去编译
- 添加DLL文件的系统路径

OpenCV安装-Linux

1. 安装gcc，CMake，ffmpeg等依赖库
2. 下载源代码
3. 使用CMake配置和生成Makefile
4. 编译OpenCV
   - make
   - make install

- 建议参考网上分享的安装指南

OpenCV的Python接口

- OpenCV的python接口函数名与C++类似
- 使用前导入cv2模块：import cv2

确保python的环境变量包含OpenCV的cv2模块的安装路径

以Linux的基于anaconda的python3.7环境举例：

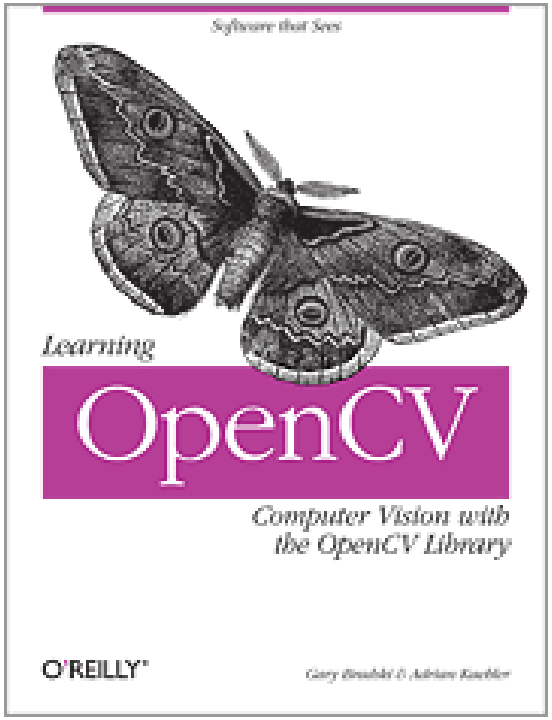
1. 找到cv2模块的安装路径：opencv_path/lib/python3.7/site-packages/cv2/python3.7/cv2.cpython-36m-x86_64-linux-gpu.so
2. 在python环境的软件包路径（anaconda_path/lib/python3.7/site-packages/）下建立符合连接：ln -s cv2模块路径 cv2.so

以Windows的基于anaconda的python3.7环境举例：

1. 找到cv2模块的安装路径：opencv_path/build/python/cv2/python3.7/cv2.cp37-win_amd64.pyd
2. 将上一步找到的文件复制到python环境的软件包路径（anaconda_path/Lib/site-packages/）下，更名为cv2.pyd
3. 将opencv的DLL文件夹(opencv_path/build/x64/vc15)添加到系统环境变量PATH里

OpenCV参考资料

- 在线资料
  - https://docs.opencv.org/
  - https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
- 参考书



OpenCV使用示例-特征匹配

1. 基于ORB算法的特征检测
2. 基于brute-force的特征匹配
3. 特征匹配结果的可视化

```python
import numpy as np
import cv2

img1 = cv2.imread('church.jpg',0) # queryImage
img2 = cv2.imread('church_part.jpg',0) # trainImage

# Initiate SIFT detector
orb = cv2.ORB()

# find the keypoints and descriptors with SIFT
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Match descriptors.
matches = bf.match(des1,des2)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]
img3 = np.zeros((max(h1, h2), w1 + w2, 3), np.uint8)
img3[:h1, :w1, 0] = img1
img3[:h2, w1:, 0] = img2
img3[:, :, 1] = img3[:, :, 0]
img3[:, :, 2] = img3[:, :, 0]

for m in matches[:10]:
    # draw the keypoints
    # print m.queryIdx, m.trainIdx, m.distance
    color = tuple([np.random.randint(0, 255) for _ in xrange(3)])
    cv2.line(img3, (int(kp1[m.queryIdx].pt[0]), int(kp1[m.queryIdx].pt[1])) , (int(
kp2[m.trainIdx].pt[0] + w1), int(kp2[m.trainIdx].pt[1])), color)

cv2.imshow("result", img3)
cv2.waitKey()
```

Read images

读取图像

Feature detection

特征检测

```
import numpy as np
import cv2

img1 = cv2.imread('church.jpg',0) # queryImage
img2 = cv2.imread('church_part.jpg',0) # trainImage

# Initiate SIFT detector
orb = cv2.ORB()

# find the keypoints and descriptors with SIFT
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Match descriptors.
matches = bf.match(des1,des2)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]
img3 = np.zeros((max(h1, h2), w1 + w2, 3), np.uint8)
img3[:h1, :w1, 0] = img1
img3[:h2, w1:, 0] = img2
img3[:, :, 1] = img3[:, :, 0]
img3[:, :, 2] = img3[:, :, 0]

for m in matches[:10]:
    # draw the keypoints
    # print m.queryIdx, m.trainIdx, m.distance
    color = tuple([np.random.randint(0, 255) for _ in xrange(3)])
    cv2.line(img3, (int(kp1[m.queryIdx].pt[0]), int(kp1[m.queryIdx].pt[1])) , (int(
kp2[m.trainIdx].pt[0] + w1), int(kp2[m.trainIdx].pt[1])), color)

cv2.imshow("result", img3)
cv2.waitKey()
```

Brute-force feature matching

特征匹配

```
import numpy as np
import cv2

img1 = cv2.imread('church.jpg',0) # queryImage
img2 = cv2.imread('church_part.jpg',0) # trainImage

# Initiate SIFT detector
orb = cv2.ORB()

# find the keypoints and descriptors with SIFT
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Match descriptors.
matches = bf.match(des1,des2)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]
img3 = np.zeros((max(h1, h2), w1 + w2, 3), np.uint8)
img3[:h1, :w1, 0] = img1
img3[:h2, w1:, 0] = img2
img3[:, :, 1] = img3[:, :, 0]
img3[:, :, 2] = img3[:, :, 0]

for m in matches[:10]:
    # draw the keypoints
    # print m.queryIdx, m.trainIdx, m.distance
    color = tuple([np.random.randint(0, 255) for _ in xrange(3)])
    cv2.line(img3, (int(kp1[m.queryIdx].pt[0]), int(kp1[m.queryIdx].pt[1])) , (int(
kp2[m.trainIdx].pt[0] + w1), int(kp2[m.trainIdx].pt[1])), color)

cv2.imshow("result", img3)
cv2.waitKey()
```

Display results

结果可视化

```
import numpy as np
import cv2

img1 = cv2.imread('church.jpg',0) # queryImage
img2 = cv2.imread('church_part.jpg',0) # trainImage

# Initiate SIFT detector
orb = cv2.ORB()

# find the keypoints and descriptors with SIFT
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Match descriptors.
matches = bf.match(des1,des2)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]
img3 = np.zeros((max(h1, h2), w1 + w2, 3), np.uint8)
img3[:h1, :w1, 0] = img1
img3[:h2, w1:, 0] = img2
img3[:, :, 1] = img3[:, :, 0]
img3[:, :, 2] = img3[:, :, 0]

for m in matches[:10]:
    # draw the keypoints
    # print m.queryIdx, m.trainIdx, m.distance
    color = tuple([np.random.randint(0, 255) for _ in xrange(3)])
    cv2.line(img3, (int(kp1[m.queryIdx].pt[0]), int(kp1[m.queryIdx].pt[1])) , (int(
kp2[m.trainIdx].pt[0] + w1), int(kp2[m.trainIdx].pt[1])), color)

cv2.imshow("result", img3)
cv2.waitKey()
```

Any questions?