1.Python入門

目录

1 概述 基本概念、语言优势、典型应用

2 Python基础语法 数据结构、面向对象、JSON、异常处理

3 机器学习四剑客 Numpy、Pandas、PIL、Matplotlib

4 课程实践 实践：豆瓣高分电影爬取

目录

1 概述 基本概念、语言优势、典型应用

2 Python基础语法 数据结构、面向对象、JSON、异常处理

3 机器学习四剑客 Numpy、Pandas、PIL、Matplotlib

4 课程实践 实践：豆瓣高分电影爬取

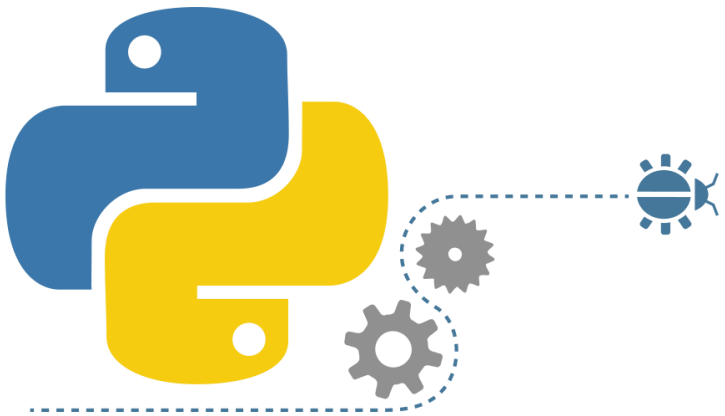
概述

基本概念

语言优势

典型应用

Python及其发展历程



Python is a programming language that lets you work quickly and integrate systems more effectively.

- Python是一门解释型、面向对象的高级编程语言.
- Python是开源免费的、支持交互式、可跨平台移植的脚本语言.

诞生和发展

- 1991年，第一个Python编译器(同时也是解释器)诞生。它是用C语言实现的，并能够调用C库(.so文件)。从一出生，Python已经具有了：类，函数，异常处理，包含表和词典在内的核心数据类型，以及模块为基础的拓展系统。
- 2000年，Python 2.0 由 BeOpen PythonLabs 团队发布，加入内存回收机制，奠定了Python语言框架的基础
- 2008年，Python 3 在一个意想不到的情况下发布了，对语言进行了彻底的修改，没有向后兼容

概述

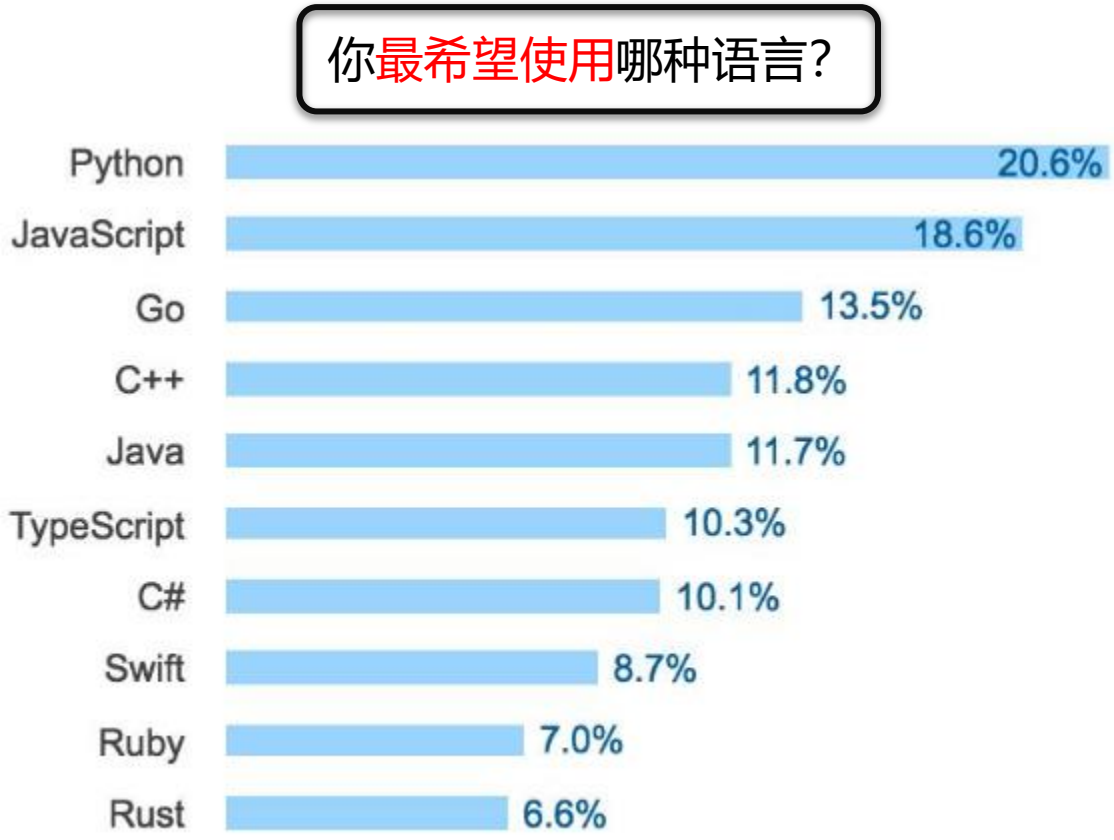
基本概念

语言优势

典型应用

为什么选择Python

Stack Overflow 2017年开发者调查报告显示，Python作为主流编程语言之一，发展势头已无可阻挡。



Python的优势

Python的设计混合了传统语言的软件工程的特点和脚本语言的易用性，具有如下特性：

- 开源、易于维护
- 可移植
- 易于使用、简单优雅
- 广泛的标准库、功能强大
- 可扩展、可嵌入
- …

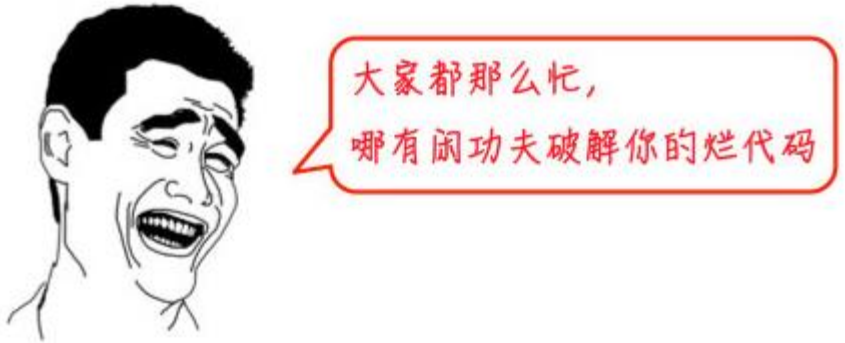
Python也存在缺点

◆运行速度慢

- Python是解释型语言，运行时翻译为机器码非常耗时，而C语言是运行前直接编译成CPU能执行的机器码。但是大量的应用程序不需要这么快的运行速度，因为用户根本感觉不出来。

◆代码不能加密

- 解释型语言发布程序就是发布源代码，而C语言只需要把编译后的机器码发布出去，从机器码反推出C代码是不可能的



概述

基本概念

语言优势

典型应用

典型应用

Django,TurboGears,web2py等框架

NumPy，SciPy，Matplotlib可以让Python程序员编写科学计算程序

Python对于各种网络协议的支持很完善，因此经常被用于编写服务器软件、网络爬虫

大多数Linux发行版以及NetBSD、OpenBSD和MacOSX都集成了Python，可以在终端下直接运行Python

PyQt、PySide、wxPython、PyGTK是Python快速开发桌面应用程序的利器

很多游戏使用C++编写图形显示等高性能模块，而使用Python或者Lua编写游戏的逻辑、服务器

典型应用

Python应用方向

数据分析

对数据进行清洗、去重、规格化和针对性的分析是大数据行业的基石。Python是数据分析的主流语言之一

科学计算

随着NumPy，SciPy，Matplotlib等众多程序库的开发，Python越来越适合于做科学计算、绘制高质量的2D和3D图像

常规软件开发

支持函数式编程和OOP面向对象编程，适用于常规的软件开发、脚本编写、网络编程

人工智能

Python在人工智能大范畴领域内的**机器学习、神经网络、深度学习**等方面都是主流的编程语言，得到广泛的支持和应用

网络爬虫

大数据行业获取数据的核心工具。Python是编写网络爬虫的主流编程语言，Scrapy爬虫框架应用非常广泛

WEB开发

基于Python的Web开发框架很多，如Django，Flask

目录

1 概述 基本概念、语言优势、典型应用

2 Python基础语法 数据结构、面向对象、JSON、异常处理

3 机器学习四剑客 Numpy、Pandas、PIL、Matplotlib

4 课程实践 实践：豆瓣高分电影爬取

Python基础语法

数据结构

面向对象

JSON

异常处理

Python安装

- Python是跨平台的，可以运行在Windows、Mac和各种Unix/Linux系统上。
- Python有两个版本，一个是2.x版，一个是3.x版，这两个版本是**不兼容的**，本教程以Python3.5版本为基础（Windows上安装时注意添加环境变量）。
- Python代码是以 .py 为扩展名的文本文件，要运行代码，需要安装Python解释器：
- CPython

  官方默认编译器，安装Python后直接获得该解释器，以>>>作为提示符
- Ipython

  基于Cpython的一个交互式解释器，用In [序号]: 作为提示符

第一个Python程序

- 交互模式：在命令行敲击命令 python，即可进入Python交互模式，提示符是>>>。
- 命令模式：在Python交互模式下输入 exit()，就退出了Python交互模式，回到命令行模式

```
C:\> python
Python 3.7.3 …
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
>>> print('hello world')
hello world
```

```
C:\> python hello.py
```

Python数据结构

数字（Number）

- Python Number 数据类型用于存储数值，包括整型、长整型、浮点型、复数。
- Python 中数学运算常用的函数基本都在 math 模块

```
import math

print(math.ceil(4.1))    #返回数字的上入整数
print(math.floor(4.9))   #返回数字的下舍整数
print(math.fabs(-10))    #返回数字的绝对值
print(math.sqrt(9))      #返回数字的平方根
print(math.exp(1))       #返回e的x次幂
```

```
5
4
10.0
3.0
2.718281828459045
```

Python数据结构

数字（Number）

Python 中随机数

- 随机生成一个 [0,1)范围内的实数

```
import random

ran = random.random()
print(ran)
```

```
0.39523871821810574
```

- 随机生成一个 [1,20)范围内的整数

```
ran = random.randint(1,20)
print(ran)
```

```
14
```

- 当使用 random.seed(x) 设定好种子之后，random() 生成的随机数将会是同一个。

```
print ("-------- 设置种子 seed --------")
random.seed(10)
print ("Random number with seed 10 : ", random.random())

# 生成同一个随机数
random.seed(10)
print ("Random number with seed 10 : ", random.random())
```

```
------- 设置种子 seed -------
Random number with seed 10 :  0.5714025946899135
Random number with seed 10 :  0.5714025946899135
```

Python数据结构

字符串（String）

◆ 单引号、双引号、三引号

➢ Python中的字符串可以使用单引号 、双引号和三引号（三个单引号或三个双引号）括起来，使用反斜杠 \ 转义特殊字符

```
print('Hello World!')
print("Hello World!")
```

```
Hello World!
Hello World!
```

```
print("The \t is a tab")
print('I\'m going to the movies')
```

```
The 	 is a tab
I'm going to the movies
```

```
print('''I'm going to the movies''')

html = '''
<HTML><HEAD><TITLE>
Friends CGI Demo</TITLE></HEAD>
<BODY><H3>ERROR</H3>
<B>%s</B><P>
<FORM><INPUT TYPE=button VALUE=Back
ONCLICK="window.history.back()"></FORM>
</BODY></HTML>
'''
print(html)
```

```
I'm going to the movies

<HTML><HEAD><TITLE>
Friends CGI Demo</TITLE></HEAD>
<BODY><H3>ERROR</H3>
<B>%s</B><P>
<FORM><INPUT TYPE=button VALUE=Back
ONCLICK="window.history.back()"></FORM>
</BODY></HTML>
```

Python数据结构

字符串（String）

- 字符串连接
  - 使用+运算符

```
str1 = "Hello "
str2 = "World!"
print(str1+str2)
```

```
Hello World!
```

  - 使用join运算符

```
new_str = '-'.join('Hello')
print(new_str)
```

```
H-e-l-l-o
```

Python数据结构

列表（List）

- 声明一个列表，并使用下标访问元素

```
#声明一个列表
names = ['jack','tom','tonney','superman','jay']

#通过下标或索引获取元素
print(names[0])
print(names[1])
```

```
jack
tom
```

- 访问最后一个元素

```
#获取最后一个元素
print(names[-1])
print(names[len(names)-1])
```

- 访问第一个元素

```
#获取第一个元素
print(names[-5])
```

Python数据结构

列表（List）

◆ 列表查询

- 查询names列表中有没有值为’superman’的元素

```
#查询names里面有没有superman
for name in names:
    if name == 'superman':
        print('有超人')
        break
else:
    print('有超人')
```

```
#更简单的方法,来查询names里有没有superman
if 'superman' in names:
    print('有超人')
else:
    print('有超人')
```

Python数据结构

列表（List）

◆ 列表添加

➢ append()：在列表末尾追加元素

```
#声明一个空列表
girls = []

#append(),末尾追加
girls.append('杨超越')
print(girls)
```

```
['杨超越']
```

➢ extend()：合并列表

```
models = ['刘雯','奚梦瑶']
girls.extend(models)
#girls = girls + models
print(girls)
```

```
['杨超越', '刘雯', '奚梦瑶']
```

➢ insert()：在指定位置添加

```
girls.insert(1,'虞书欣')
print(girls)
```

```
['杨超越', '虞书欣', '刘雯', '奚梦瑶']
```

Python数据结构

列表（List）

◆ 列表修改

➢ 修改指定元素

```
fruits = ['apple','pear','香蕉','pineapple','草莓']
print(fruits)
fruits[-1] = 'strawberry'
print(fruits)
```

```
['apple', 'pear', '香蕉', 'pineapple', '草莓']
['apple', 'pear', '香蕉', 'pineapple', 'strawberry']
```

➢ 将fruits列表中的‘香蕉’替换为‘banana’

```
for fruit in fruits:
    if '香蕉' in fruit:
        fruit = 'banana'
print(fruits)
```

```
for i in range(len(fruits)):
    if '香蕉' in fruits[i]:
        fruits[i] = 'banana'
        break
print(fruits)
```

Python数据结构

列表（List）

◆ 列表删除

- del()

```
words = ['cat','hello','pen','pencil','ruler']
del words[1]
print(words)
```

- remove()

```
words = ['cat','hello','pen','pencil','ruler']
words.remove('cat')
print(words)
```

- pop()

```
words = ['cat','hello','pen','pencil','ruler']
words.pop(1)
print(words)
```

Python数据结构

列表（List）

◆ 列表切片

➢ []中设置要使用的第一个元素和最后一个元素的索引。牢记：左闭右开。

元素	cat	dog	tiger	snake	mouse	bird
正向	0	1	2	3	4	5
反向	-6	-5	-4	-3	-2	-1

```
animals = ['cat','dog','tiger','snake','mouse','bird']

print(animals[2:5])

print(animals[-1:])

print(animals[-3:-1])

print(animals[::2])

print(animals[-5:-1:2])
```

```
['tiger', 'snake', 'mouse']
['bird']
['snake', 'mouse']
['cat', 'tiger', 'mouse']
['dog', 'snake']
```

Python数据结构

列表（List）

◆ 列表排序

➢ 生成10个不同的随机数，存至列表中，并进行排序

```
import  random

random_list = []
for i in range(10):
    ran = random.randint(1,20)
    if ran not in  random_list:
        random_list.append(ran)
print(random_list)
```

```
import  random

random_list = []
i = 0
while i < 10:
    ran = random.randint(1,20)
    if ran not in  random_list:
        random_list.append(ran)
        i+=1
print(random_list)

#默认升序
new_list = sorted(random_list)
print(new_list)

#降序
new_list = sorted(random_list,reverse =True)
print(new_list)
```

Python数据结构

元组（Tuple）

- 与列表类似，区别是元组中的内容不可修改
  - 定义一个元组，**注意：元组中只有一个元素时，需要在后面加逗号**

```
tuple1 = ()
print(type(tuple1))
```

```
<class 'tuple'>
```

```
tuple2 = ('hello')
print(type(tuple2))
```

```
<class 'str'>
```

```
tuple3 = ('hello',)
print(type(tuple3))
```

```
<class 'tuple'>
```

Python数据结构

元组（Tuple）

◆ 列表转元组

➢ 元组不能修改，所以不存在往元组里加入元素。作为那作为容器的元组，如何存放元素？

```
import random
random_list = []
for i in range(10):
    ran = random.randint(1,20)
    random_list.append(ran)
print(random_list)

random_tuple = tuple(random_list)
print(random_tuple)
```

```
[16, 19, 1, 7, 15, 16, 9, 6, 2, 17]
(16, 19, 1, 7, 15, 16, 9, 6, 2, 17)
```

Python数据结构

元组（Tuple）

- 元组截取

```
print(random_tuple)                (16, 19, 1, 7, 15, 16, 9, 6, 2, 17)
print(random_tuple[0])             16
print(random_tuple[-1])            17
print(random_tuple[1:-3])          (19, 1, 7, 15, 16, 9)
print(random_tuple[::-1])          (17, 2, 6, 9, 16, 15, 7, 1, 19, 16)
```

- 元组的一些函数

```
print(max(random_tuple))           19
print(min(random_tuple))           1
print(sum(random_tuple))           108
print(len(random_tuple))           10
```

Python数据结构

元组（Tuple）

- 元组的拆包与装包

元组元素个数与变量个数相等：

```
#定义一个元组
t3 = (1,2,3)

#将元组赋值给变量a,b,c
a,b,c = t3

#打印a,b,c
print(a,b,c)
```

```
1 2 3
```

元组元素个数与变量个数不相等：

```
#定义一个元组，包含5个元素
t4 = (1,2,3,4,5)

#将t4[0],t4[1]分别赋值给a,b;其余的元素装包后赋值给c
a,b,*c = t4

print(a,b,c)
print(c)
```

```
1 2 [3, 4, 5]
[3, 4, 5]
```

Python数据结构

字典（Dict）

- 定义一个空字典

```
dict1 = {}
```

- 定义一个字典的同时，进行初始化

```
dict2 = {'name':'杨超越','weight':45,'age':25}
print(dict2['name'])
```

- 定义一个字典，之后添加元素

```
dict4 = {}
dict4['name'] = '虞书欣'
dict4['weight'] = 43
print(dict4)
```

- 修改字典元素

```
dict4['weight'] = 44
print(dict4)
```

Python数据结构

字典（Dict）

◆ 字典相关函数

➢ items()

```
dict5 = {'杨超越':165,'虞书欣':166,'上官喜爱':164}
print(dict5.items())
for key,value in dict5.items():
    if value > 165:
        print(key)
```

```
dict_items([('杨超越', 165), ('虞书欣', 166), ('上官喜爱', 164)])
虞书欣
```

➢ values()

```
#求小姐姐的平均身高
heights = dict5.values()
print(heights)
total = sum(heights)
avg = total/len(heights)
print(avg)
```

```
dict_values([165, 166, 164])
165.0
```

➢ keys()

```
names = dict5.keys()
print(names)
```

```
dict_keys(['杨超越', '虞书欣', '上官喜爱'])
```

Python数据结构

字典（Dict）

◆ 字典删除

- del

```
dict6 = {'杨超越':165,'虞书欣':166,'上官喜爱':164}
del dict6['杨超越']
print(dict6)
```

```
{'虞书欣': 166, '上官喜爱': 164}
```

- pop()

```
result = dict6.pop('虞书欣')
print(result)
print(dict6)
```

```
166
{'上官喜爱': 164}
```

Python基础语法

数据结构

面向对象

JSON

异常处理

Python面向对象

类

```
class Animal:

    def __init__(self,name):
        self.name = name
        print('动物名称实例化')
    def eat(self):
        print(self.name +'要吃东西啦！ ')
    def drink(self):
        print(self.name +'要喝水啦！ ')

cat =  Animal('miaomiao')
print(cat.name)
cat.eat()
cat.drink()
```

定义一个类Animals:

(1)**init**()定义构造函数，与其他面向对象语言不同的是，Python语言中，会明确地把代表自身实例的self作为第一个参数传入

(2)创建一个实例化对象 cat，**init**()方法接收参数

(3)使用点号 . 来访问对象的属性。

```
动物名称实例化
miaomiao
miaomiao要吃东西啦！
miaomiao要喝水啦！
```

Python面向对象

类

```
class Person:
    def __init__(self,name):
        self.name = name
        print ('调用父类构造函数')

    def eat(self):
        print('调用父类方法')

class Student(Person):  # 定义子类
    def __init__(self):
        print ('调用子类构造方法')

    def study(self):
        print('调用子类方法')

s = Student()           # 实例化子类
s.study()               # 调用子类的方法
s.eat()                 # 调用父类方法
```

✓ 通过继承创建的新类称为子类或派生类，被继承的类称为基类、父类或超类。

```
调用子类构造方法
调用子类方法
调用父类方法
```

Python基础语法

数据结构

面向对象

JSON

异常处理

Python JSON

JSON序列化与反序列化

- JSON序列化：json.dumps 用于将 Python 对象编码成 JSON 字符串。

```
import json
data = [ { 'b' : 2, 'd' : 4, 'a' : 1, 'c' : 3, 'e' : 5 } ]
# json = json.dumps(data)
# print(json)
json_format = json.dumps(data, sort_keys=True, indent=4, separators=(',', ': '))
print(json_format)
```

```
[
    {
        "a": 1,
        "b": 2,
        "c": 3,
        "d": 4,
        "e": 5
    }
]
```

- JSON反序列化：json.loads 用于解码 JSON 数据。该函数返回 Python 字段的数据类型。

```
import json
jsonData = '{"a":1,"b":2,"c":3,"d":4,"e":5}'
text = json.loads(jsonData)
print(text)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

Python基础语法

数据结构

面向对象

JSON

异常处理

Python异常处理

Python异常处理

- try/except语句用来检测try语句块中的错误，从而让except语句捕获异常信息并处理。

```
try:
    fh = open("/home/aistudio/data/testfile01.txt", "w")
    fh.write("这是一个测试文件，用于测试异常!!")
except IOError:
    print('Error: 没有找到文件或读取文件失败')
else:
    print ('内容写入文件成功')
    fh.close()
```

➢finally中的内容，退出try时总会执行。

```
try:
    f = open("/home/aistudio/data/testfile02.txt", "w")
    f.write("这是一个测试文件，用于测试异常!!")
finally:
    print('关闭文件')
    f.close()
```

目录

1 概述 基本概念、语言优势、典型应用

2 Python基础语法 数据结构、面向对象、JSON、异常处理

3 机器学习四剑客 Numpy、Pandas、PIL、Matplotlib

4 课程实践 实践：豆瓣高分电影爬取

机器学习四剑客

Numpy库

Pandas库

PIL库

Matplotlib库

Numpy库

Numpy库介绍

◆Numpy（Numerical Python的简称）是高性能科学计算和数据分析的基础包。其部分功能如下：

- ndarray，一个具有矢量算术运算和复杂广播能力的快速且节省空间的多维数组。
- 用于对整组数据进行快速运算的标准数学函数（无需编写循环）。
- 用于读写磁盘数据的工具以及用于操作内存映射文件的工具。
- 线性代数、随机数生成以及傅里叶变换功能。
- 用于集成由C、C++、Fortran等语言编写的代码的工具。

Numpy库

ndarray

- 创建ndarray：创建数组最简单的办法就是使用array函数。它接受一切序列型的对象（包括其他数组），然后产生一个新的含有传入数据的numpy数组。其中，嵌套序列（比如由一组等长列表组成的列表）将会被转换为一个多维数组
- 除了np.array之外，还有一些函数也可以新建数组：
  - zeros和ones分别可以创建指定长度或者形状的全0或全1数组
  - Empty可以创建一个没有任何具体值的数组

```
>>> import numpy as np
>>> a = [1, 2, 3, 4]   # 创建简单的列表
>>> b = np.array(a)    # 将列表转换为数组
>>> b
array([1, 2, 3, 4])
```

```
# 创建10行10列的数值为浮点0的矩阵
array_zero = np.zeros([10,10])
```

```
# 创建10行10列的数值为浮点1的矩阵
array_one = np.ones([10,10])
```

Numpy库

ndarray

- 创建随机数组
  - 均匀分布
    - ✓ np.random.rand(10, 10)　创建指定形状(示例为10行10列)的数组(范围在0至1之间)
    - ✓ np.random.uniform(0, 100)　创建指定范围内的一个数
    - ✓ np.random.randint(0, 100)　创建指定范围内的一个整数
  - 正态分布
    - ✓ np.random.normal(1.75, 0.1, (2, 3))　给定均值/标准差/维度的正态分布

Numpy库

ndarray

◆查看数组属性的用法

用法	说明
b.size	数组元素个数
b.Shape	数组形状
b.ndim	数组维度
b.dtype	数组元素类型

Numpy库

数组和标量之间的运算

- 数组很重要，因为它可以使我们不用编写循环即可对数据执行批量运算。这通常叫做矢量化（vectorization）。大小相等的数组之间的任何算术运算都会将运算应用到元素级。同样，数组与标量的算术运算也会将那个标量值传播到各个元素

```
>>> arr = np.array([[1., 2., 3.], [4., 5., 6.]])
>>> arr
array([[1., 2., 3.],
       [4., 5., 6.]])
```

```
>>> arr - arr
array([[0., 0., 0.],
       [0., 0., 0.]])
```

```
>>> arr * arr
array([[ 1.,  4.,  9.],
       [16., 25., 36.]])
```

```
>>> 1 / arr
array([[1.        , 0.5     , 0.33333333],
       [0.25      , 0.2     , 0.16666667]])
```

```
>>> arr ** 0.5
array([[1. , 1.41421356, 1.73205081],
       [2. , 2.23606798, 2.44948974]])
```

Numpy库

基本的索引和切片

- ✓ Numpy数组的索引是一个内容丰富的主题，因为选取数据子集或的单个元素的方式有很多
- ✓ 一维数组很简单。从表面上看，它们跟Python列表的功能差不多
- ✓ 跟列表最重要的区别在于，数组切片是原始数组的视图。这意味着数据不会被复制，视图上的任何修改都会直接反映到源数组上
- ✓ 将一个标量值赋值给一个切片时，该值会自动传播到整个选区（如下图所示）

```
>>> arr = np.arange(10)
>>> arr
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> arr[5]
5
>>> arr[5:8]
array([5, 6, 7])
>>> arr[5:8] = 12
```

```
>>> arr
array([ 0,  1,  2,  3,  4, 12, 12, 12,  8,  9])
>>> arr_slice = arr[5:8]
>>> arr_slice[1] = 12345
>>> arr
array([ 0,  1,  2,  3,  4, 12, 12345, 12,  8,  9])
>>> arr_slice[:] = 64
>>> arr
array([ 0,  1,  2,  3,  4, 64, 64, 64,  8,  9])
```

Numpy库

基本的索引和切片

- ✓ 在二维数组中，各索引位置上的元素不再是标量而是一维数组
- ✓ 可以对各个元素进行递归访问，但是这样有点麻烦
- ✓ 还有一种方式是传入一个以逗号隔开的索引列表来选取单个元素
- ✓ 在多维数组中，如果省略了后面的索引，则返回对象会是一个维度低一点的ndarray

```
>>> arr3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
>>> arr3d
array([[[ 1,  2,  3],
        [ 4,  5,  6]],
       [[ 7,  8,  9],
        [10, 11, 12]]])
```

```
>>> arr3d[0]
array([[1, 2, 3],
       [4, 5, 6]])
```

```
>>> arr3d[0][1]
array([4, 5, 6])
```

```
>>> arr3d[0, 1]
array([4, 5, 6])
```

Numpy库

数学和统计方法

- ✓ 可以通过数组上的一组数学函数对整个数组或某个轴向的数据进行统计计算
- ✓ sum、mean以及标准差std等聚合计算既可以当做数组的实例方法调用，也可以当做顶级Numpy函数使用

```
>>> arr = np.random.randn(5, 4)  # 正态分布的数据
>>> arr.mean()
-0.022341797127577216
>>> np.mean(arr)
-0.022341797127577216
>>> arr.sum()
-0.4468359425515443 5
```

Numpy库

数学和统计方法

- mean 和 sum 这类的函数可以接受一个 axis 参数（用于计算该轴向上的统计值），最终结果是一个少一维的数组

```
>>> arr.mean(axis=1)
array([-0.11320162, -0.032351  , -0.24522299,  0.13275031,  0.14631631])
>>> arr.sum(0)
array([-1.71093252,  3.4431099 , -1.78081725, -0.39819607])
```

Numpy库

数学和统计方法

◆其他如 cumsum 和 cumprod 之类的方法则不聚合，而是产生一个由中间结果组成的数组

```
>>> arr = np.array([[0, 1 ,2], [3, 4, 5], [6, 7, 8]])
>>> arr.cumsum(0)
array([[ 0,  1,  2],
       [ 3,  5,  7],
       [ 9, 12, 15]], dtype=int32)
>>> arr.cumprod(1)
array([[  0,   0,   0],
       [  3,  12,  60],
       [  6,  42, 336]], dtype=int32)
```

Numpy库

数学和统计方法

◆ 基本数组统计方法如下

方法	说明
sum	对数组中全部或某轴向的元素求和。零长度的数组的sum为0
mean	算术平均数。零长度的数组的mean为NaN
std, var	分别为标准差和方差，自由度可调（默认为n）
min, max	最大值和最小值
argmin, argmax	分别为最大和最小元素的索引
cumsum	所有元素的累加
cumprod	所有元素的累积

Numpy库

线性代数

- 线性代数（如矩阵乘法、矩阵分解、行列式以及其他方阵数学等）是任何数组库的重要组成部分
- numpy提供了一个用于矩阵乘法的dot函数（既是一个数组方法，也是numpy命名空间中的一个函数）

```
>>> x = np.array([[1., 2., 3.], [4., 5., 6.]])
>>> y = np.array([[6., 23.], [-1, 7], [8, 9]])
>>> x
array([[1., 2., 3.],
       [4., 5., 6.]])
>>> y
array([[ 6., 23.],
       [-1.,  7.],
       [ 8.,  9.]])
>>> x.dot(y)  # 相当于np.dot(x, y)
array([[ 28.,  64.],
       [ 67., 181.]])
```

Numpy库

线性代数

- numpy.linalg 中有一组标准的矩阵分解运算以及诸如求逆和行列式之类的东西
- 它们跟 MATLAB 和 R 等语言所使用的是相同的行业标准级Fortran库
- 右边为常用的 numpy.linalg 函数：

函数	说明
diag	以一维数组的形式返回方阵的对角线（或非对角线）元素，或将一维数组转换为方阵（非对角线元素为0）
dot	矩阵乘法
trace	计算对角线元素的和
det	计算矩阵行列式
eig	计算方阵的特征值和特征向量
inv	计算方阵的逆
pinv	计算矩阵的Mooer-Penrose伪逆
qr	计算OR分解
svd	计算奇异值分解（SVD）
solve	解线性方程组Ax=b，其中A为一个方阵
lstsq	计算Ax=b的最小二乘解

机器学习四剑客

Numpy库

Pandas库

PIL库

Matplotlib库

Pandas库

Pandas库介绍

- Pandas是Python第三方库，提供高性能易用数据类型和分析工具
- Pandas基于Numpy实现，常与Numpy和Matplotlib一同使用
- Pandas中有两大核心数据结构：Series（一维数据） 和 DataFrame（多特征数据,既有行索引,又有列索引）

Series

index	value
0	12
1	4
2	7
3	9

Series的数据结构为"键值对"的形式

键—>可以重复

DataFrame

index	writer	title	price
0	mark	cookbook	23.56
1	barket	HTML5	50.70
2	tom	Python	12.30
3	job	Numpy	28.00

DataFrame可以进行行索引列索引

是Pandas中重要的数据结构

Pandas库

Series

- Series是一种类似于一维数组的对象，它由一维数组（各种numpy数据类型）以及一组与之相关的数据标签（即索引）组成
- Series的创建：
  - ✓ 使用Python数组创建
  - ✓ 使用numpy数组创建
  - ✓ 使用python字典创建

注意：与字典不同的是：Series允许索引重复

```
>>> import pandas as pd
>>> import numpy as np
>>> pd.Series([11, 12], index=["北京", "上海"])
北京    11
上海    12
dtype: int64
```

```
>>> pd.Series(np.arange(3,6))
0    3
1    4
2    5
dtype: int32
```

```
>>> pd.Series({"北京": 11, "上海": 12, "深圳": 14})
北京    11
上海    12
深圳    14
dtype: int64
```

Pandas库

Series

- Series的字符串表现形式为：索引在左边，值在右边
- 如果没有为数据指定索引，则自动创建一个0到N-1（N为数据的长度）的整数型索引
- 可以通过Series的values和index属性获取其数组表示形式和索引对象
- 与普通numpy数组相比，可以通过索引的方式选取Series中的单个或一组值

```
>>> obj = pd.Series([4, 7, -5, 3])
>>> obj.values
array([ 4,  7, -5,  3], dtype=int64)
>>> obj.index
RangeIndex(start=0, stop=4, step=1)
```

```
>>> obj[2]
-5
>>> obj[1] = 8
>>> obj[[0, 1, 3]]
0    4
1    8
3    3
dtype: int64
```

Pandas库

Series

◆ Series中最重要的一个功能是：它会在算术运算中自动对齐不同索引的数据

```
>>> obj2 = pd.Series({"Ohio": 35000, "Oregon": 16000, "Texas": 71000, "Utah": 5000})
>>> obj3 = pd.Series({"California": np.nan, "Ohio": 35000, "Oregon": 16000, "Texas": 71000})
```

```
>>> obj2 + obj3
California        NaN
Ohio          70000.0
Oregon        32000.0
Texas        142000.0
Utah              NaN
dtype: float64
```

Pandas库

Series

- Series对象本身及其索引都有一个name属性，该属性跟pandas其他的关键功能关系非常密切
- Series的索引可以通过赋值的方式就地修改

```
>>> obj3.name= 'population'
>>> obj3.index.name = 'state'
>>> obj3
state
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
Name: population, dtype: float64
```

```
>>> obj = pd.Series([4, 7, -5, 3])
>>> obj.index = ['Bob', 'Steve', 'Jeff', 'Ryan']
>>> obj
Bob      4
Steve    7
Jeff    -5
Ryan     3
dtype: int64
```

Pandas库

DataFrame

- DataFrame是一个表格型的数据结构，它含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔值等）
- DataFrame既有行索引也有列索引，它可以被看做由Series组成的字典（共用同一个索引）
- 跟其他类似的数据结构相比（如R语言的data.frame），DataFrame中面向行和面向列的操作基本上是平衡的
- DataFrame中的数据是以一个或多个二维块存放的（而不是列表、字典或别的一维数据结构）

Pandas库

DataFrame

- 构成DataFrame的方法很多，最常用的一种是直接传入一个由等长列表或Numpy数组组成的字典
- DataFrame结果会自动加上索引（跟Series一样），且全部会被有序排列

```
>>> data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'], 'year': [2000
, 2001, 2002, 2001, 2002], 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
>>> frame = pd.DataFrame(data)
>>> frame
   state  year  pop
0   Ohio  2000  1.5
1   Ohio  2001  1.7
2   Ohio  2002  3.6
3 Nevada  2001  2.4
4 Nevada  2002  2.9
```

Pandas库

DataFrame

- 如果指定了列顺序，则DataFrame的列就会按照指定顺序进行排列
- 跟原Series一样，如果传入的列在数据中找不到，就会产生NAN值

```
>>> pd.DataFrame(data,
columns=['year', 'state', 'pop'])
   year   state  pop
0  2000    Ohio  1.5
1  2001    Ohio  1.7
2  2002    Ohio  3.6
3  2001  Nevada  2.4
4  2002  Nevada  2.9
```

```
>>> frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
index=['one', 'two', 'three', 'four', 'five'])
>>> frame2
       year   state  pop debt
one    2000    Ohio  1.5  NaN
two    2001    Ohio  1.7  NaN
three  2002    Ohio  3.6  NaN
four   2001  Nevada  2.4  NaN
five   2002  Nevada  2.9  NaN
>>> frame2.columns
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

Pandas库

DataFrame

- 通过类似字典标记的方式或属性的方式，可以将DataFrame的列获取为一个Series
- 返回的Series拥有原DataFrame相同的索引，且其name属性也已经被相应地设置好了

```
>>> frame2['state']
one        Ohio
two        Ohio
three      Ohio
four     Nevada
five     Nevada
Name: state, dtype: object
```

```
>>> frame2['year']
one       2000
two       2001
three     2002
four      2001
five      2002
Name: year, dtype: int64
```

Pandas库

DataFrame

- 列可以通过赋值的方式进行修改
- 例如，给那个空的“delt”列赋上一个标量值或一组值

```
>>> frame2['debt'] = 16.5
>>> frame2
      year   state  pop  debt
one    2000    Ohio  1.5  16.5
two    2001    Ohio  1.7  16.5
three  2002    Ohio  3.6  16.5
four   2001  Nevada  2.4  16.5
five   2002  Nevada  2.9  16.5
>>> frame2['debt'] = np.arange(5.)
>>> frame2
      year   state  pop  debt
one    2000    Ohio  1.5   0.0
two    2001    Ohio  1.7   1.0
three  2002    Ohio  3.6   2.0
four   2001  Nevada  2.4   3.0
five   2002  Nevada  2.9   4.0
```

Pandas库

DataFrame

- 将列表或数组赋值给某个列时，其长度必须跟DataFrame的长度相匹配
- 如果赋值的是一个Series，就会精确匹配DataFrame的索引，所有空位都将被填上缺失值

```
>>> val = pd.Series([-1.2, -1.5, -1.7], index=['two', 'four', 'five'])
>>> frame2['debt'] = val
>>> frame2
       year   state  pop  debt
one    2000    Ohio  1.5   NaN
two    2001    Ohio  1.7  -1.2
three  2002    Ohio  3.6   NaN
four   2001  Nevada  2.4  -1.5
five   2002  Nevada  2.9  -1.7
```

Pandas库

DataFrame

- 为不存在的列赋值会创建出一个新列
- 关键字del用于删除列

```
>>> frame2['eastern'] = frame2.state == 'Ohio'
>>> frame2
       year   state  pop  debt  eastern
one    2000    Ohio  1.5   NaN     True
two    2001    Ohio  1.7  -1.2     True
three  2002    Ohio  3.6   NaN     True
four   2001  Nevada  2.4  -1.5    False
five   2002  Nevada  2.9  -1.7    False
>>> del frame2['eastern']
>>> frame2.columns
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

Pandas库

DataFrame

- 将嵌套字典（也就是字典的字典）传给DataFrame，它就会被解释为：外层字典的键作为列，内层键则作为行索引
- 也可以对上述结果进行转置

```
>>> pop = {'Nevada': {2001: 2.4, 2002: 2.9}, 'Ohio':
{2000: 1.5, 2001: 1.7, 2002
: 3.6}}
>>> frame3 = pd.DataFrame(pop)
>>> frame3
      Nevada  Ohio
2000     NaN   1.5
2001     2.4   1.7
2002     2.9   3.6
```

```
>>> frame3.T
        2000  2001  2002
Nevada   NaN   2.4   2.9
Ohio     1.5   1.7   3.6
```

Pandas库

DataFrame

◆如果设置了DataFrame的index和columns的name属性，则这些信息也会被显示出来

```
>>> frame3.index.name = 'year'
>>> frame3.columns.name = 'state'
>>> frame3
state  Nevada  Ohio
year
2000      NaN   1.5
2001      2.4   1.7
2002      2.9   3.6
```

Pandas库

DataFrame

- 跟Series一样，values属性也会以二维ndarray的形式返回DataFrame中的数据
- 如果DataFrame各列的数据类型不同，则数组的数据类型就会选用能兼容所有列的数据类型

```
>>> frame3.values
array([[nan, 1.5],
       [2.4, 1.7],
       [2.9, 3.6]])
```

```
>>> frame2.values
array([[2000, 'Ohio', 1.5, nan],
       [2001, 'Ohio', 1.7, -1.2],
       [2002, 'Ohio', 3.6, nan],
       [2001, 'Nevada', 2.4, -1.5],
       [2002, 'Nevada', 2.9, -1.7]], dtype=object)
```

Pandas库

索引对象

- Pandas的索引对象负责管理轴标签和其他元数据（比如轴名称等）
- 构建DataFrame时，所用到的任何数组或其他序列的标签都会被转换成一个Index
- Index对象是不可修改的，因此用户不能对其进行修改

```
>>> obj = pd.Series(range(3), index=['a', 'b', 'c'])
>>> index = obj.index
>>> index
Index(['a', 'b', 'c'], dtype='object')
>>> index[1:]
Index(['b', 'c'], dtype='object')
```

Pandas库

索引对象

- Pandas的每个索引都有一些方法和属性，它们可用于设置逻辑并回答有关该索引包含的数据的常见问题。右表中列出了Index的方法和属性：

方法	说明
append	连接另一个Index对象，产生一个新的Index
diff	计算差集，并得到一个Index
intersection	计算交集
union	计算并集
isin	计算一个指示各值是否都包含在参数集合中的布尔型数组
delete	删除索引处的元素，并得到新的Index
drop	删除传入的值，并得到新的Index
insert	将元素插入到索引处，并得到新的Index
is_monotonic	当各元素均大于等于前一个元素时，返回True
is_unique	当Index没有重复值时，返回True
unique	计算Index中唯一值的数组

机器学习四剑客

Numpy库

Pandas库

PIL库

Matplotlib库

PIL库

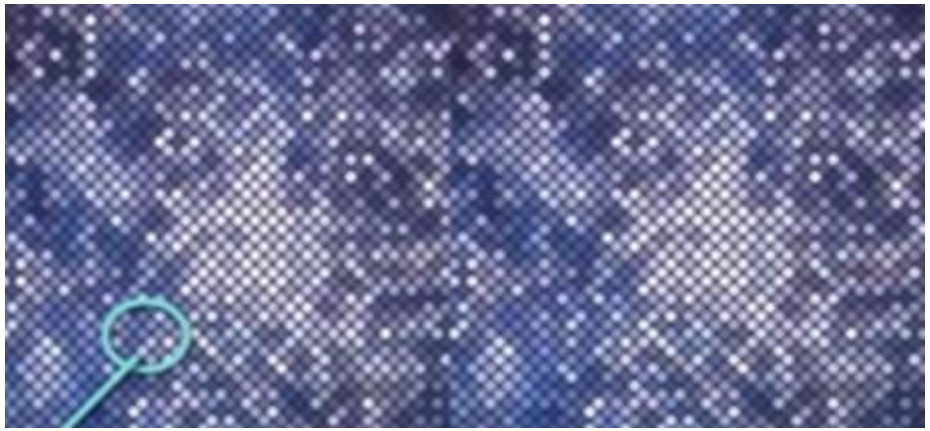
PIL库介绍

- PIL库是一个具有强大图像处理能力的第三方库
- 在命令行下的安装方法：pip install pillow
- 在使用过程中的引入方法：from PIL import Image
- Image 是 PIL 库中代表一个图像的类（对象）

PIL库

图像的数组表示

- 图像是一个由像素组成的二维矩阵，每个元素是一个RGB值



PIL库

PIL库

◆Image 模块中的一个简单的例子：读取图片，并进行45°旋转，然后进行可视化

```
>>> from PIL import Image
>>> im = Image.open('test.png')  # 读取图片
>>> im.rotate(45).show()  # 将图片旋转，并用系统自带的图片工具显示图片
```

PIL库

PIL库

- 创建缩略图
- 缩略图不能直接双击打开，要使用PIL.Image的open读取，然后使用show()方法进行显示。

```
>>> from PIL import Image
>>> import glob, os
>>> size = 128, 128
>>> for infile in glob.glob('*.jpg') # glob的作用是文件搜索，返回一个列表
...     file, ext = os.path.splitext(infile)  # 将文件名和扩展名分开，用于之后
的重命名保存
...     im = Image.open(infile)
...     im.thumbnail(size, Image.ANTIALIAS)  # 等比例缩放
...     im.save(file + '.thumbnail', 'JPEG')
```

PIL库

PIL库

- 常用的图片的融合或者合成函数如下
  - PIL.image.alpha_composite(im1,im2)
  - PIL.image.blend(im1,im2,alpha)
  - PIL.Image.composite(im1,im2,mask)
- 上述方法要求im1和im2的mode和size要一致；alpha代表图片占比的意思；mask是mode可以为”1”；”L”或者”RGBA”的size和im1、im2一致的

PIL库

Image模块

- Image.convert(mode=None,matrix=None,dither=None,palette=0,color=256)
- 使用下面的方式也可以实现图像的灰度化处理
- Image.copy()　# 将读取的图片复制一份

```
>>> from PIL import Image
>>> im = Image.open('test.png')
>>> im = im.convert("L")
>>> im.show()
```

PIL库

Image模块

- 获取图片的基本信息

```
>>> from PIL import Image
>>> im = Image.open('test.png')  # 读取图片
>>> bands = im.getbands()   # 显示该图像的所有通道，返回一个
tuple
>>> bands
('R', 'G', 'B', 'A')
>>> bboxs = im.getbbox()   # 返回一个像素坐标
>>> bboxs
(0, 0, 238, 295)
```

PIL库

Image模块

- 图像粘贴操作
  - Image.paste(im, box=None, maske=None)
  - 使用im粘贴到原图片中
  - 两个图片的mode和size要求一致，不一致可以使用convert()和resize()进行调整

机器学习四剑客

Numpy库

Pandas库

PIL库

Matplotlib库

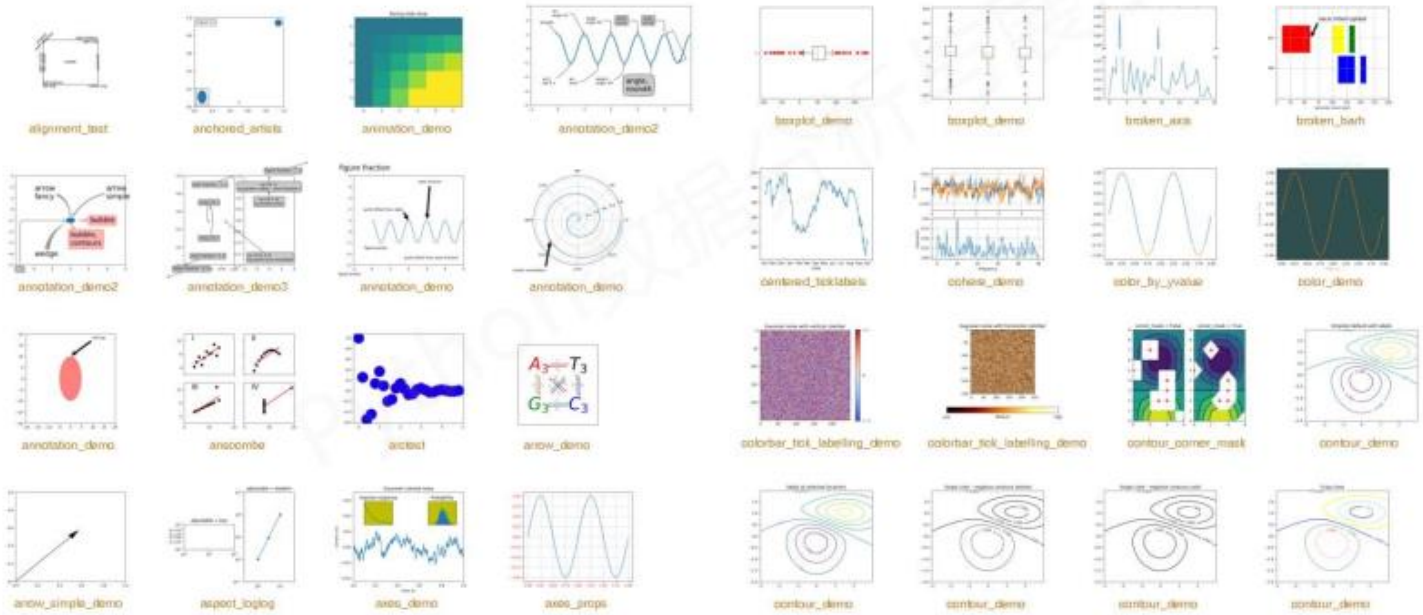
Matplotlib库

Matplotlib库介绍

- Matplotlib库由各种可视化类构成，内部结构复杂
- 受Matlab启发，matplotlib.pylot是绘制各类可视化图形的命令字库，相当于快捷方式

Matplotlib库的效果

http://matplotlib.org/gallery.html

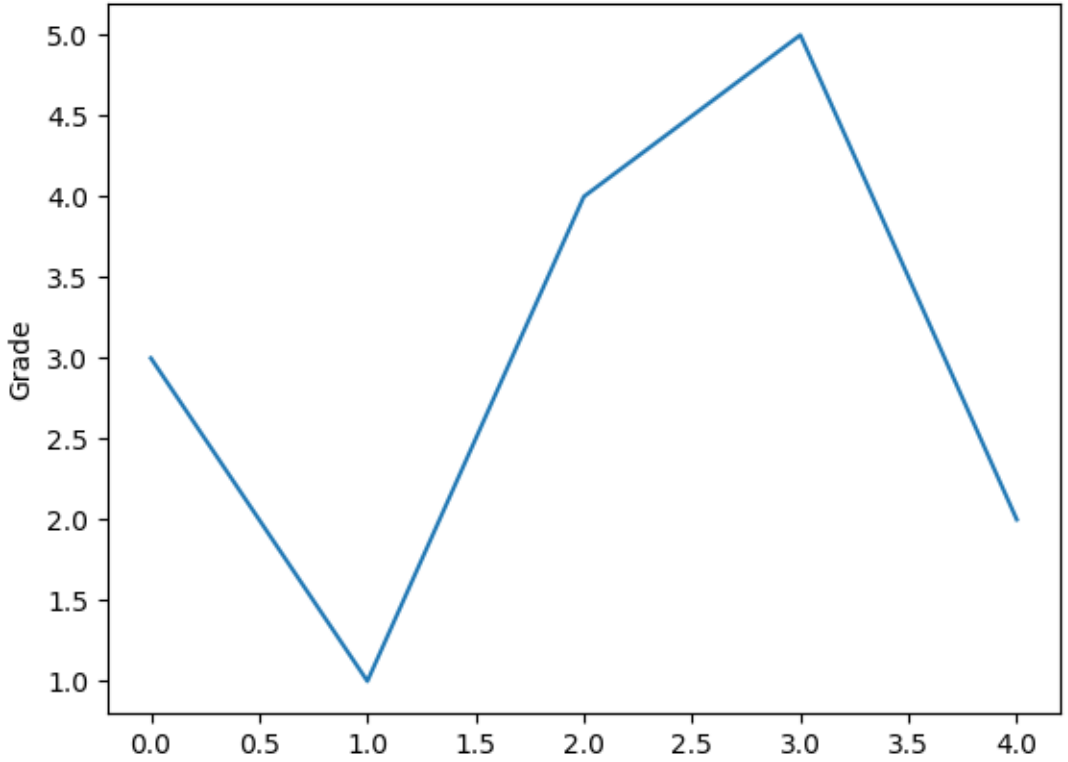


Matplotlib库

Matplotlib库使用

- plt.plot() 只有一个输入列表或数组时，参数被当做Y轴，X轴以索引自动生成
- plt.savefig() 将输出图形存储为文件，默认PNG格式，可以通过dpi修改输出质量

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([3, 1, 4, 5, 2])
[<matplotlib.lines.Line2D object at
0x000000000B2A0978>]
>>> plt.ylabel("Grade")
Text(0, 0.5, 'Grade')
>>> plt.savefig("test", dpi=600)  # PNG文件
>>> plt.show()
```

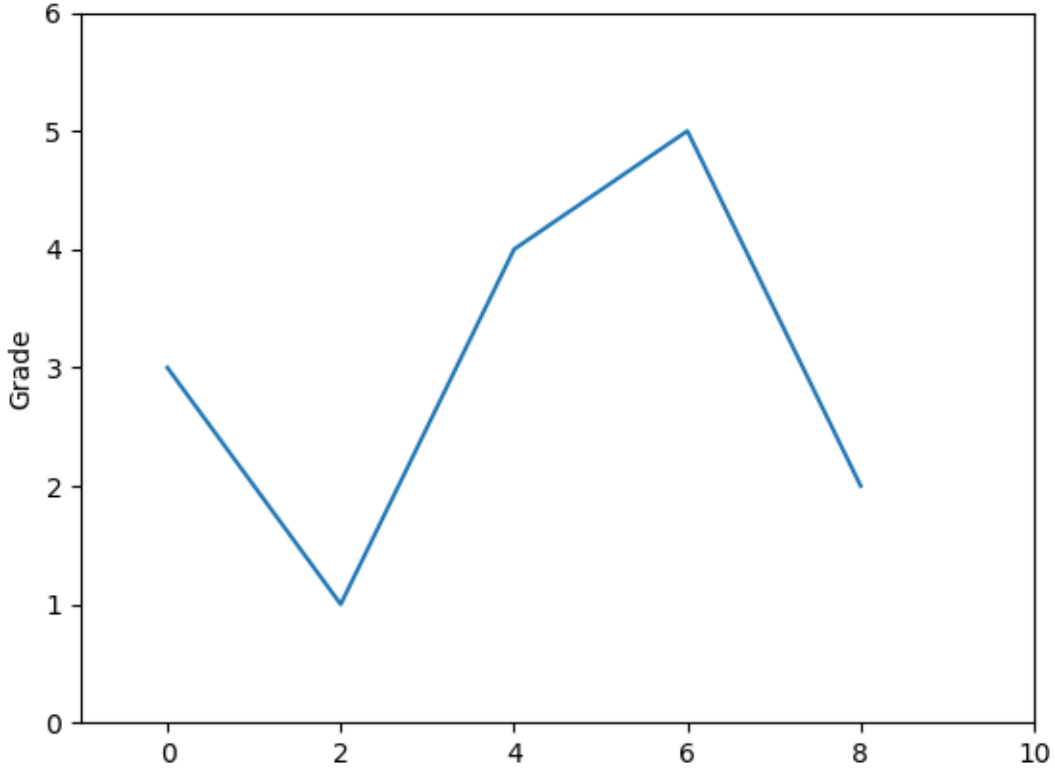


Matplotlib库

Matplotlib库使用

- plt.plot(x, y) 当有两个以上参数时，按照x轴和y轴顺序绘制数据点

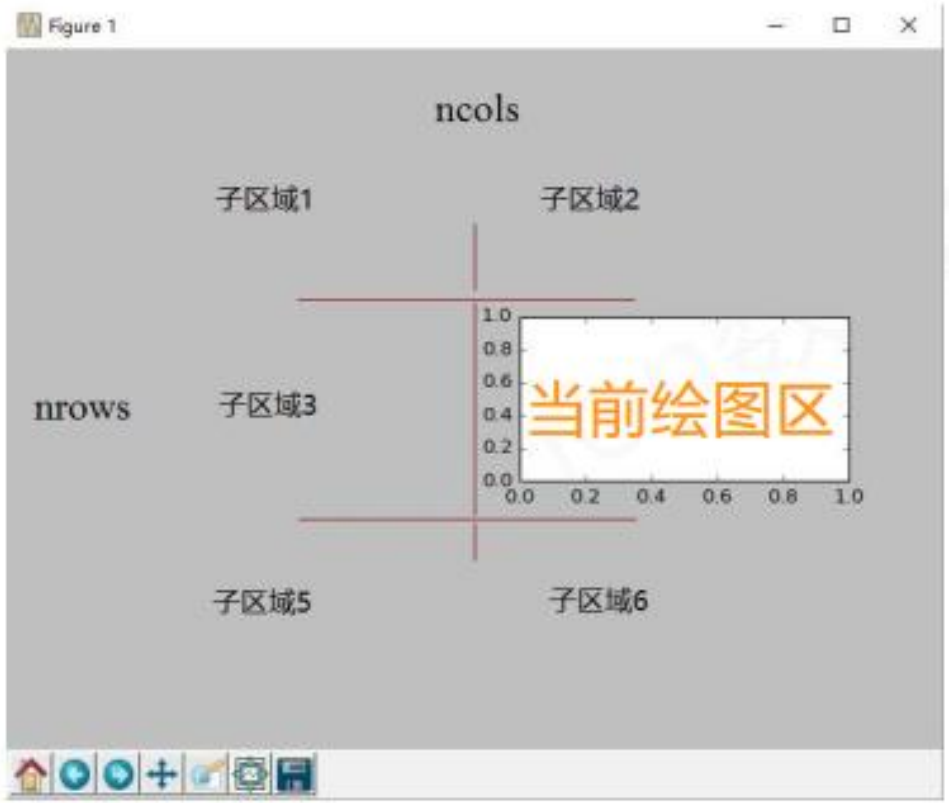
```
>>> import matplotlib.pyplot as plt
>>> plt.plot([0, 2, 4, 6, 8], [3, 1, 4, 5, 2])
[<matplotlib.lines.Line2D object at
0x000000000DA38470>]
>>> plt.ylabel("Grade")
Text(0, 0.5, 'Grade')
>>> plt.axis([-1, 10, 0, 6])
[-1, 10, 0, 6]
>>> plt.show()
```



Matplotlib库

Matplotlib库使用

- pyplot的绘图区域
  - plt.subplot(nrows, ncols, plot_number)
  - 在全局绘图区域中创建一个分区体系，并定位到一个子绘图区域



Matplotlib库

Matplotlib库使用

◆ pyplot的基础图标函数如下

函数	说明
plt.plot(x,y,fmt,…)	绘制一个坐标图
plt.boxplot(data, notch, position)	绘制一个箱形图
plt.bar(left, height, width, bottom)	绘制一个条形图
plt.barh(width, bottom, left, height)	绘制一个横向条形图
plt.polar(theta, r)	绘制极坐标图
plt.pie(data, explode)	绘制饼图

Matplotlib库

Matplotlib库使用

◆ pyplot的基础图标函数如下

函数	说明
plt.psd(x, NFFT=256, pad_to, Fs)	绘制功率谱密度图
plt.specgram(x, NFFT=256, pad_to, F)	绘制谱图
plt.cohere(x, y, NFFT=256, Fs)	绘制X-Y的相关性函数
plt.scatter(x, y)	绘制散点图，其中，x和y长度相同
plt.step(x, y, where)	绘制步阶图
plt.hist(x, bins, normed)	绘制直方图

Matplotlib库

Matplotlib库使用

◆ pyplot的基础图标函数如下

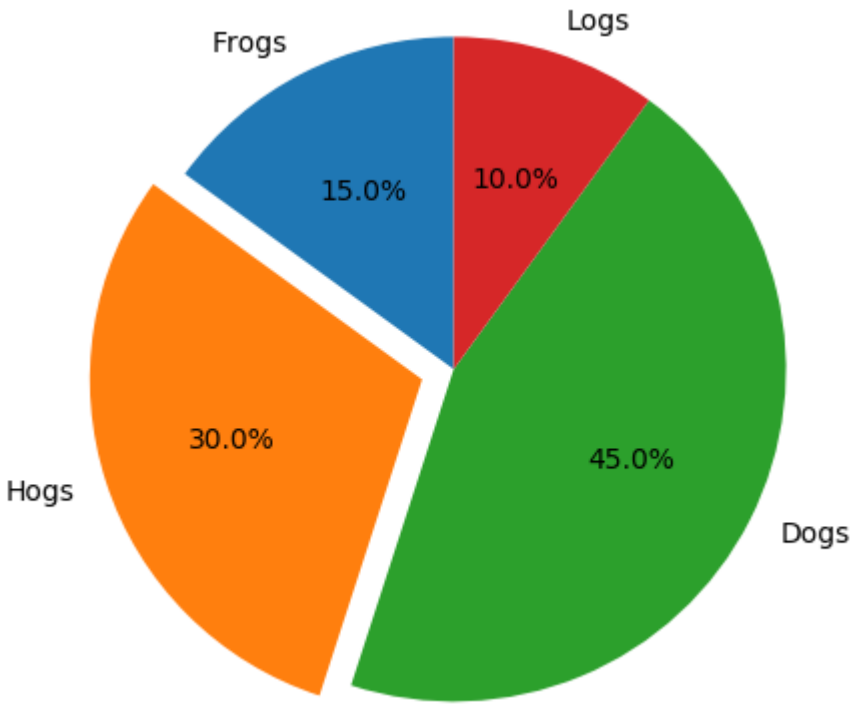
函数	说明
plt.contour(X, Y, Z, N)	绘制等值图
plt.vlines()	绘制垂直图
plt.stem(x, y, linefmt, markerfmt)	绘制柴火图
plt.plot_date()	绘制数据日期

Matplotlib库

Matplotlib库使用

◆ 饼图的绘制

```
>>> import matplotlib.pyplot as plt
>>> labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
>>> sizes = [15, 30, 45, 10]
>>> explode = (0, 0.1, 0, 0)
>>> plt.pie(sizes, explode=explode, labels=labels,
autopct='%1.1f%%', shadow=False, startangle=90)
>>> plt.axis('equal')
>>> plt.show()
```

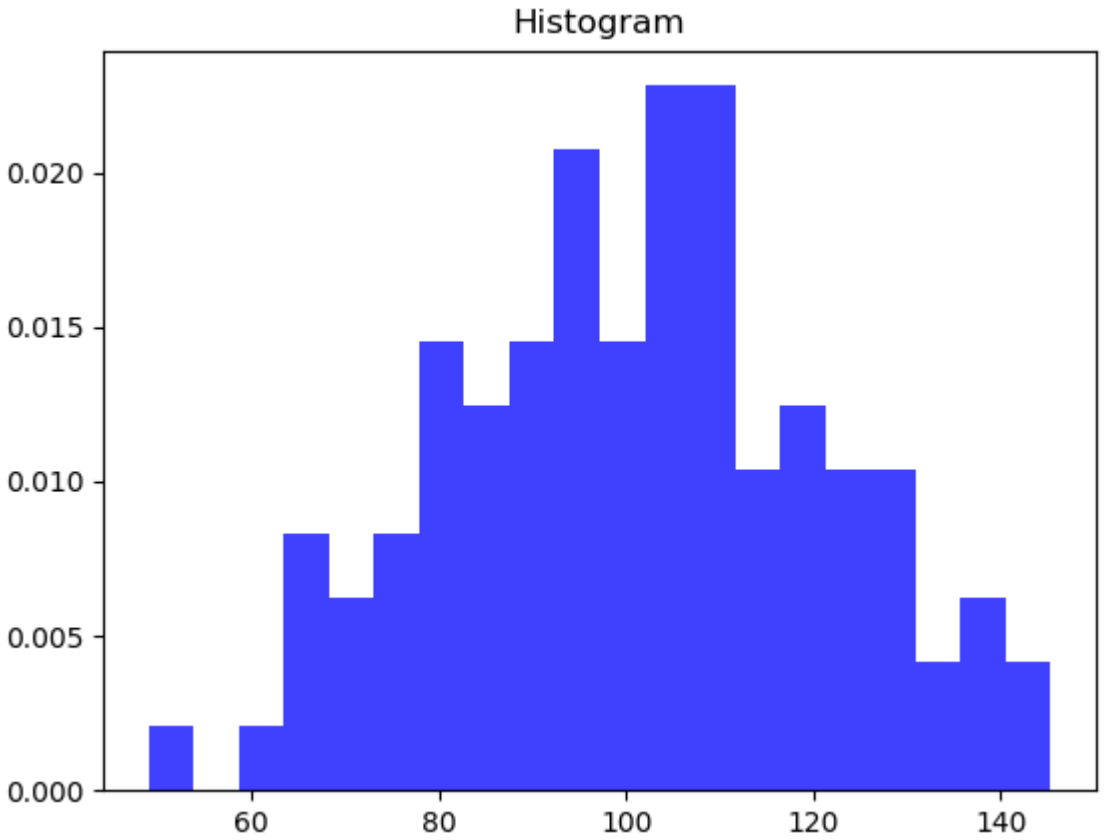


Matplotlib库

Matplotlib库使用

◆ **直方图的绘制**

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> np.random.seed(0)
>>> mu, sigma = 100, 20  # 均值和标准值
>>> a = np.random.normal(mu, sigma,
size=100)                          → 直方图的个数
>>> plt.hist(a, 20, normed=1,
histtype='stepfilled', facecolor='b', alpha=0.75)
>>> plt.title('Histogram')
>>> plt.show()
```

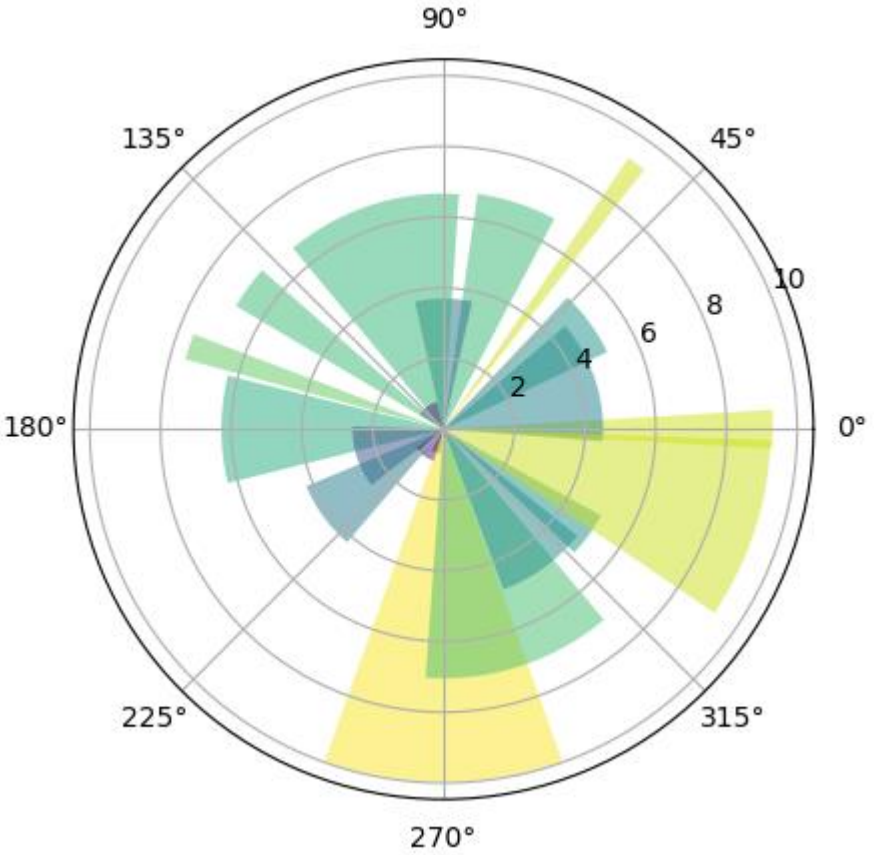


Matplotlib库

Matplotlib库使用

◆ 极坐标图的绘制

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> N = 20
>>> theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
>>> radii = 10 * np.random.rand(N)
>>> width = np.pi / 4 * np.random.rand(N)
>>> ax = plt.subplot(111, projection='polar')
>>> bars = ax.bar(theta, radii, width=width, bottom=0.0)
>>> for r, bar in zip(radii, bars):
...     bar.set_facecolor(plt.cm.viridis(r / 10.))
...     bar.set_alpha(0.5)
...
>>> plt.show()
```

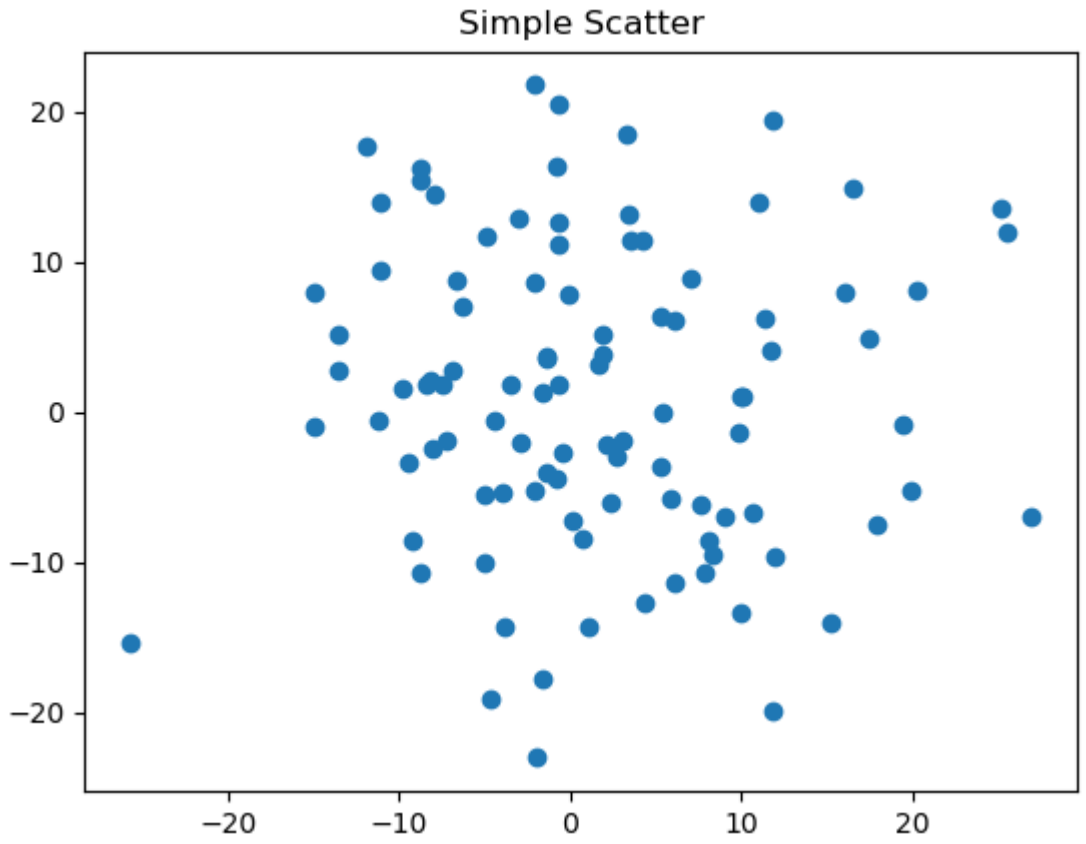


Matplotlib库

Matplotlib库使用

◆ 散点图的绘制

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots()
>>> ax.plot(10 * np.random.randn(100), 10 *
np.random.randn(100), 'o')
>>> ax.set_title('Simple Scatter')
Text(0.5, 1.0, 'Simple Scatter')
>>> plt.show()
```



目录

1 概述 基本概念、语言优势、典型应用

2 Python基础语法 数据结构、面向对象、JSON、异常处理

3 机器学习四剑客 Numpy、Pandas、PIL、 Matplotlib

4 课程实践 实践：豆瓣高分电影爬取

课程实践

实践：豆瓣高分电影爬取