# 计算库

# [numpy](http://www.numpy.org/)

NumPy提供了两种基本的对象：ndarray（N-dimensional array object）和 ufunc（universal function object）。ndarray是存储单一数据类型的多维数组，ufunc则是能够对数组进行处理的函数。NumPy 通常与 **SciPy**（Scientific Python）和 **Matplotlib**（绘图库）一起使用。 这种组合广泛用于替代 MatLab，是一个流行的技术计算平台。

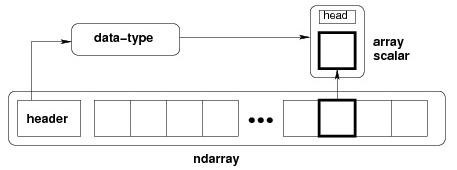
**功能：**

* 数组的算数和逻辑运算。
* 傅立叶变换和用于图形操作的例程。
* 与线性代数有关的操作。 NumPy 拥有线性代数和随机数生成的内置函数。

## Ndarray 对象

NumPy 中定义的最重要的对象是称为 ndarray 的 N 维数组类型。 它描述相同类型的元素集合。 可以使用基于零的索引访问集合中的项目。

ndarray中的每个元素在内存中使用相同大小的块。 ndarray中的每个元素是数据类型对象的对象（称为 dtype）。从ndarray对象提取的任何元素（通过切片）由一个数组标量类型的 Python 对象表示。



**创建ndarray**

**numpy.array(object,dtype=None,copy=True,order=None,subok=False,ndmin=0)**

| 参数 | 属性 |
| --- | --- |
| object | 任何暴露数组接口方法的对象都会返回一个数组或任何（嵌套）序列。 |
| dtype | 可选，数组的所需数据类型 |
| copy | 可选，默认为true，对象是否被复制。 |
| order | C（按行）、F（按列）或A（任意，默认）。 |
| subok | 默认情况下，返回的数组被强制为基类数组。 如果为true，则返回子类。 |
| ndimin | 指定返回数组的最小维数。 |

## 数据类型

### numpy数据类型

| 数据类型 | 描述 |
| --- | --- |
| bool\_ | 存储为一个字节的布尔值（真或假） |
| int\_ | 默认整数，相当于 C 的long，通常为int32或int64 |
| intc | 相当于 C 的int，通常为int32或int64 |
| intp | 用于索引的整数，通常为int32或int64 |
| int8 | 字节（-128 ~ 127） |
| int16 | 16 位整数（-32768 ~ 32767） |
| int32 | 32 位整数（-2147483648 ~ 2147483647） |
| int64 | 64 位整数 |
| uint8 | 8 位无符号整数（0 ~ 255） |
| uint16 | 16 位无符号整数（0 ~ 65535） |
| uint32 | 32 位无符号整数（0 ~ 4294967295） |
| uint64 | 64 位无符号整数（0 ~ 18446744073709551615） |
| float16 | 半精度浮点：符号位，5 位指数，10 位尾数 |
| float32 | 单精度浮点：符号位，8 位指数，23 位尾数 |
| float64、float\_ | 双精度浮点：符号位，11 位指数，52 位尾数 |
| complex64 | 复数，由两个 32 位浮点表示（实部和虚部） |
| complex128、complex\_ | 复数，由两个 64 位浮点表示（实部和虚部） |

### 数据类型对象

数据类型对象描述了对应于数组的固定内存块的解释，取决于以下方面：

* 数据类型（整数、浮点或者 Python 对象）
* 数据大小
* 字节序（小端或大端）
* 在结构化类型的情况下，字段的名称，每个字段的数据类型，和每个字段占用的内存块部分。
* 如果数据类型是子序列，它的形状和数据类型。

字节顺序取决于数据类型的前缀<或>。 <意味着编码是小端（最小有效字节存储在最小地址中）。 >意味着编码是大端（最大有效字节存储在最小地址中）。

**创建dtype**

**numpy.dtype(object, align, copy)**

参数：

* Object：被转换为数据类型的对象。
* Align：如果为true，则向字段添加间隔，使其类似 C 的结构体。
* Copy：生成dtype对象的新副本，如果为flase，结果是内建数据类型对象的引用。

dt=np.dtype([('age',np.int8)])  
a=np.array([(10,),(20),(30,)],dtype=dt)  
print(a['age'])

## 数组

### 广播

**广播**是指 NumPy 在算术运算期间处理不同形状的数组的能力。对数组的算术运算通常在相应的元素上进行。 如果两个阵列具有完全相同的形状，则这些操作被无缝执行。

a = np.array([1,2,3,4])   
b = np.array([10,20,30,40])   
c = a \* b

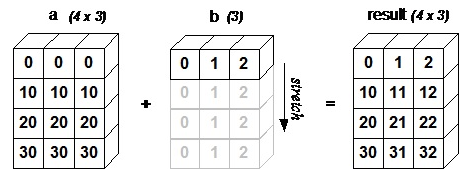
如果两个数组的维数不相同， 较小的数组会**广播**到较大数组的大小，以便使它们的形状可兼容。

满足以下规则，可以进行广播：

* ndim较小的数组会在前面追加一个长度为 1 的维度。
* 输出数组的每个维度的大小是输入数组该维度大小的最大值。
* 如果输入在每个维度中的大小与输出大小匹配，或其值正好为 1，则在计算中可它。
* 如果输入的某个维度大小为 1，则该维度中的第一个数据元素将用于该维度的所有计算。

如果上述规则产生有效结果，并且满足以下条件之一，那么数组被称为**可广播的**。

* 数组拥有相同形状。
* 数组拥有相同的维数，每个维度拥有相同长度，或者长度为 1。
* 数组拥有极少的维度，可以在其前面追加长度为 1 的维度，使上述条件成立。



### 数组操作

1. **数组属性**

**ndarray.shape** #返回一个包含数组维度的元组

调整数组大小

a = np.array([[1,2,3],[4,5,6]])

a.shape = (3,2) #调整数组为三行两列

**ndarray.ndim** #返回数组的维数

**numpy.itemsize**  #返回数组中每个元素的字节单位长度

1. **修改形状**

|  |  |
| --- | --- |
| reshape | 不改变数据的条件下修改形状 |
| flat | 数组上的一维迭代器 |
| flatten | 返回折叠为一维的数组副本 |
| ravel | 返回连续的展开数组 |

**numpy.reshape(arr, newshape, order='C')**

* arr：要修改形状的数组
* newshape：整数或者整数数组，新的形状应当兼容原有形状
* order：'C'为 C 风格顺序，'F'为 F 风格顺序，'A'为保留原顺序。

**numpy.flags**  返回ndarray当前属性值

1. **翻转操作**

|  |  |
| --- | --- |
| 1. | transpose 翻转数组的维度 |
| 2. | ndarray.T 和self.transpose()相同 |
| 3. | rollaxis 向后滚动指定的轴 |
| 4. | swapaxes 互换数组的两个轴 |

**numpy.transpose(arr, axes)**

* arr：要转置的数组
* axes：整数的列表，对应维度，通常所有维度都会翻转。

**numpy.rollaxis(arr, axis, start)**

* arr：输入数组
* axis：要向后滚动的轴，其它轴的相对位置不会改变
* start：默认为零，表示完整的滚动。会滚动到特定位置。

**numpy.swapaxes(arr, axis1, axis2)**

* arr：要交换其轴的输入数组
* axis1：对应第一个轴的整数
* axis2：对应第二个轴的整数

1. **修改维度**

|  |  |
| --- | --- |
| 1. | broadcast 产生模仿广播的对象 |
| 2. | broadcast\_to 将数组广播到新形状 |
| 3. | expand\_dims 扩展数组的形状 |
| 4. | squeeze 从数组的形状中删除单维条目 |

**numpy.expand\_dims(arr, axis)**

* arr：输入数组
* axis：新轴插入的位置

**numpy.squeeze(arr, axis)**

* arr：输入数组
* axis：整数或整数元组，用于选择形状中单一维度条目的子集

1. **数组的连接**

|  |  |
| --- | --- |
| 1. | concatenate 沿着现存的轴连接数据序列 |
| 2. | stack 沿着新轴连接数组序列 |
| 3. | hstack 水平堆叠序列中的数组（列方向） |
| 4. | vstack 竖直堆叠序列中的数组（行方向） |

**numpy.concatenate((a1, a2, ...), axis)**

* a1, a2, ...：相同类型的数组序列
* axis：沿着它连接数组的轴，默认为 0

**numpy.stack(arrays, axis)**

* arrays：相同形状的数组序列
* axis：返回数组中的轴，输入数组沿着它来堆叠

1. **数组分割**

|  |  |
| --- | --- |
| 1. | split 将一个数组分割为多个子数组 |
| 2. | hsplit 将一个数组水平分割为多个子数组（按列） |
| 3. | vsplit 将一个数组竖直分割为多个子数组（按行） |

**numpy.split(ary, indices\_or\_sections, axis)**

* ary：被分割的输入数组
* indices\_or\_sections：可以是整数，表明要从输入数组创建的，等大小的子数组的数量。 如果此参数是一维数组，则其元素表明要创建新子数组的点。
* axis：默认为 0

1. **添加/删除元素**

|  |  |
| --- | --- |
| 1. | resize 返回指定形状的新数组 |
| 2. | append 将值添加到数组末尾 |
| 3. | insert 沿指定轴将值插入到指定下标之前 |
| 4. | delete 返回删掉某个轴的子数组的新数组 |
| 5. | unique 寻找数组内的唯一元素 |

**numpy.resize(arr, shape)**

* arr：要修改大小的输入数组
* shape：返回数组的新形状

**numpy.append(arr, values, axis)**

* arr：输入数组
* values：要向arr添加的值，比如和arr形状相同（除了要添加的轴）
* axis：沿着它完成操作的轴。如果没有提供，两个参数都会被展开。

**numpy.insert(arr, obj, values, axis)**

* arr：输入数组
* obj：在其之前插入值的索引
* values：要插入的值
* axis：沿着它插入的轴，如果未提供，则输入数组会被展开

**Numpy.delete(arr, obj, axis)**

* arr：输入数组
* obj：可以被切片，整数或者整数数组，表明要从输入数组删除的子数组
* axis：沿着它删除给定子数组的轴，如果未提供，则输入数组会被展开

**numpy.unique(arr, return\_index, return\_inverse, return\_counts)**

* arr：输入数组，如果不是一维数组则会展开
* return\_index：如果为true，返回输入数组中的元素下标
* return\_inverse：如果为true，返回去重数组的下标，它可以用于重构输入数组
* return\_counts：如果为true，返回去重数组中的元素在原数组中的出现次数

### 位操作

|  |  |
| --- | --- |
| 1. | bitwise\_and 对输入数组中的整数的二进制表示的相应位执行位与运算 |
| 2. | bitwise\_or 对输入数组中的整数的二进制表示的相应位执行位或运算 |
| 3. | invert 计算位非，对于有符号整数，返回补码。 |
| 4. | left\_shift 将数组元素的二进制表示中的位向左移动到指定位置，右侧补0 |
| 5. | right\_shift将数组元素的二进制表示中的位向右移动到指定位置，左侧补0 |

### 构造函数

1. 创建指定形状和dtype的未初始化数组（数组元素为随机值）

**numpy.empty(shape, dtype = float, order = 'C')**

|  |  |
| --- | --- |
| 1. | Shape 空数组的形状，整数或整数元组 |
| 2. | Dtype 所需的输出数组类型，可选 |
| 3. | Order 'C'为按行的 C 风格数组，'F'为按列的 Fortran 风格数组 |

1. 返回特定大小，以 1 填充的新数组。

**numpy.ones(shape, dtype = None, order = 'C')**

|  |  |
| --- | --- |
| 1. | Shape 空数组的形状，整数或整数元组 |
| 2. | Dtype 所需的输出数组类型，可选 |
| 3. | Order 'C'为按行的 C 风格数组，'F'为按列的 Fortran 风格数组 |

1. Python 序列转换为ndarray

**numpy.asarray(a, dtype = None, order = None)**

|  |  |
| --- | --- |
| 1. | a 任意形式的输入参数，比如列表、列表的元组、元组、元组的元组、元组的列表 |
| 2. | dtype 通常，输入数据的类型会应用到返回的ndarray |
| 3. | order 'C'为按行的 C 风格数组，'F'为按列的 Fortran 风格数组 |

1. 将缓冲区解释为一维数组

**numpy.frombuffer(buffer, dtype = float, count = -1, offset = 0)**

|  |  |
| --- | --- |
| 1. | buffer 任何暴露缓冲区借口的对象 |
| 2. | dtype 返回数组的数据类型，默认为float |
| 3. | count 需要读取的数据数量，默认为-1，读取所有数据 |
| 4. | offset 需要读取的起始位置，默认为0 |

1. 从任何可迭代对象构建一个ndarray对象

**numpy.fromiter(iterable, dtype, count = -1)**

|  |  |
| --- | --- |
| 1. | iterable 任何可迭代对象 |
| 2. | dtype 返回数组的数据类型 |
| 3. | count 需要读取的数据数量，默认为-1，读取所有数据 |

1. 从数值范围创建数组

**numpy.arange(start, stop, step, dtype)**

|  |  |
| --- | --- |
| 1. | start 范围的起始值，默认为0 |
| 2. | stop 范围的终止值（不包含） |
| 3. | step 两个值的间隔，默认为1 |
| 4. | dtype 返回ndarray的数据类型，如果没有提供，则会使用输入数据的类型。 |

**numpy.linspace(start, stop, num, endpoint, retstep, dtype)**

|  |  |
| --- | --- |
| 1. | start 序列的起始值 |
| 2. | stop 序列的终止值，如果endpoint为true，该值包含于序列中 |
| 3. | num 要生成的等间隔样例数量，默认为50 |
| 4. | endpoint 序列中是否包含stop值，默认为ture |
| 5. | retstep 如果为true，返回样例，以及连续数字之间的步长 |
| 6. | dtype 输出ndarray的数据类型 |

返回一个ndarray对象，其中包含在对数刻度上均匀分布的数字。 刻度的开始和结束端点是某个底数的幂，通常为 10。

**numpy.logspace(start, stop, num, endpoint, base, dtype)**

|  |  |
| --- | --- |
| 1. | start 起始值是base \*\* start |
| 2. | stop 终止值是base \*\* stop |
| 3. | num 范围内的数值数量，默认为50 |
| 4. | endpoint 如果为true，终止值包含在输出数组当中 |
| 5. | base 对数空间的底数，默认为10 |
| 6. | dtype 输出数组的数据类型，如果没有提供，则取决于其它参数 |

### 数组上的迭代

NumPy 包包含一个迭代器对象numpy.nditer。 它是一个有效的多维迭代器对象，可以用于在数组上进行迭代。 数组的每个元素可使用 Python 的标准Iterator接口来访问。

a = np.arange(0,60,5)  
*for* x *in* np.nditer(a):  
 print(x) #迭代输出每一个元素

1. **迭代顺序**

如果相同元素使用 F 风格顺序存储，则迭代器选择以更有效的方式对数组进行迭代。

a = np.arange(0,60,5)  
a = a.reshape(3,4)  
b = a.T  
c = b.copy(order='C') #以 C 风格顺序排序  
*for* x *in* np.nditer(c):  
 print(x) #输出0 20 40 5 25 45 10 30 50 15 35 55  
c = b.copy(order='F') #以 F 风格顺序排序  
*for* x *in* np.nditer(c):  
 print(x) # 输出0 5 10 15 20 25 30 35 40 45 50 55

可以通过显式提醒，来强制nditer对象使用某种顺序

*for* x *in* np.nditer(a, order = 'C'):

1. **修改数组的值**

nditer对象有另一个可选参数op\_flags。 其默认值为只读，但可以设置为读写或只写模式。 这将允许使用此迭代器修改数组元素。

*for* x *in* np.nditer(a, op\_flags=['readwrite']):

1. **外部循环**

nditer类的构造器拥有flags参数，它可以接受下列值

|  |  |
| --- | --- |
| 1. | c\_index 可以跟踪 C 顺序的索引 |
| 2. | f\_index 可以跟踪 Fortran 顺序的索引 |
| 3. | multi-index 每次迭代可以跟踪一种索引类型 |
| 4. | external\_loop 给出的值是具有多个值的一维数组，而不是零维数组 |

1. **广播迭代**

如果两个数组是**可广播的**，nditer组合对象能够同时迭代它们。

## 切片和索引

ndarray对象中的元素遵循基于零的索引。 有三种可用的索引方法类型： 字段访问，基本切片和高级索引。

### 基本切片

基本切片是 Python 中基本切片概念到 n 维的扩展。 通过将start，stop和step参数提供给内置的slice函数来构造一个 Python的slice对象。 此slice对象被传递给数组来提取数组的一部分。

通过将由冒号分隔的切片参数（start:stop:step）直接提供给ndarray对象，也可以获得相同的结果

如果只输入一个参数，则将返回与索引对应的单个项目。 如果使用start:，则从该索引向后的所有项目将被提取。 如果使用两个参数（以:分隔），则对两个索引（不包括停止索引）之间的元素以默认步骤进行切片。

a = np.arange(10)  
s = slice(2,7,2)   
b = a[2:7:2]

切片还可以包括省略号（...），来使选择元组的长度与数组的维度相同。 如果在行位置使用省略号，它将返回包含行中元素的ndarray。

a = np.array([[1,2,3],[3,4,5],[4,5,6]])  
print(a[...,1]) # 第二列元素  
print(a[1,...]) # 第二行元素  
print(a[...,1:]) # 第二列及剩余元素

### 高级索引

如果一个ndarray是非元组序列，数据类型为整数或布尔值的ndarray，或者至少一个元素为序列对象的元组，我们就能够用它来索引ndarray。高级索引始终返回数据的副本。 与此相反，切片只提供了一个视图。

有两种类型的高级索引：整数和布尔值。

1. 整数索引

每个整数数组表示该维度的下标值。

x = np.array([[ 0, 1, 2],[ 3, 4, 5],[ 6, 7, 8],[ 9, 10, 11]])   
rows = np.array([[0,0],[3,3]])   
cols = np.array([[0,2],[0,2]]) #(0,0)(0,2)(3,0)(3,2)的元素  
y = x[rows,cols] #输出一个二维数组

x = np.array([[1, 2], [3, 4], [5, 6]])   
y = x[[0,1,2], [0,1,0]] #(0,0)(1,1)(2,0)  
print y

1. 布尔索引

结果对象是布尔运算（例如比较运算符）的结果时，将使用此类型的高级索引。

print (x[x > 5]) #打印出大于 5 的元素

## 字符串函数（numpy.char）

|  |  |
| --- | --- |
| 1. | add() 返回两个str或Unicode数组的逐个字符串连接 |
| 2. | multiply() 返回按元素多重连接后的字符串 |
| 3. | center() 返回给定字符串的副本，其中元素位于特定字符串的中央 |
| 4. | capitalize() 返回给定字符串的副本，其中只有第一个字符串大写 |
| 5. | title() 返回字符串或 Unicode 的按元素标题转换版本 |
| 6. | lower() 返回一个数组，其元素转换为小写 |
| 7. | upper() 返回一个数组，其元素转换为大写 |
| 8. | split() 返回字符串中的单词列表，并使用分隔符来分割 |
| 9. | splitlines() 返回元素中的行列表，以换行符分割 |
| 10. | strip() 返回数组副本，其中元素移除了开头或者结尾处的特定字符 |
| 11. | join() 返回一个字符串，它是序列中字符串的连接 |
| 12. | replace() 返回字符串的副本，其中所有子字符串的出现位置都被新字符串取代 |
| 13. | decode() 按元素调用str.decode |
| 14. | encode() 按元素调用str.encode |

## 算数函数

1. 三角函数

NumPy 拥有标准的三角函数，它为弧度制单位的给定角度返回三角函数比值。

arcsin，arccos，和arctan函数返回给定角度的sin，cos和tan的反三角函数。 这些函数的结果可以通过numpy.degrees()函数通过将弧度制转换为角度制来验证。

1. 舍入函数

**numpy.around(a,decimals)**

|  |  |
| --- | --- |
| 1. | a 输入数组 |
| 2. | decimals 要舍入的小数位数，默认值0。如果为负，整数将四舍五入到小数点左侧 |

**numpy.floor()**

此函数返回不大于输入参数的最大整数。 即标量x 的下限是最大的整数i ，使得i <= x。 注意在Python中，向下取整总是从 0 舍入

**numpy.ceil()**

ceil()函数返回输入值的上限，即，标量x的上限是最小的整数i ，使得i> = x。

## 算数运算

用于执行算术运算（如add()，subtract()，multiply()和divide()）的输入数组必须具有相同的形状或符合数组广播规则。

# 图形库

# Matplotlib

## 图形及参数

plt.plot(x, y, label='First Line') # label：线条名称，以后可以在图例中显示

plt.xlabel('Plot Number') # 相应的轴创建标签

plt.ylabel('Important var')

plt.title('Interesting Graph\nCheck it out') #创建图的标题

plt.legend() #生成默认图例

plt.show()

1. 条形图

plt.bar创建条形图

plt.bar([1,3,5,7,9], [5,3,7,8,3],label='first')  
plt.bar([2,4,6,8,10], [12,3,2,6,4],color='g',label='second')  
plt.title('bar')  
plt.legend()  
plt.ylabel('y')  
plt.xlabel('x')  
plt.show()

1. 直方图

直方图倾向于通过将区段组合在一起来显示分布。

plt.hist，首先需要放入所有的值，然后指定放入哪个桶或容器

a=np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])  
plt.hist(a,bins=[0,20,40,60,80,100],color='g',histtype='bar', rwidth=0.8)  
plt.title('histogrm')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.legend()  
plt.show()

1. 散点图

x = [1,2,3,4,5,6,7,8]  
y = [5,2,4,2,1,4,5,2] #如果在 3 维绘制则是 3 个。  
plt.scatter(x,y, label='skitscat', color='k', s=25, marker="o")  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Interesting Graph\nCheck it out')  
plt.legend()  
plt.show()

1. 堆叠图

堆叠图用于显示『部分对整体』随时间的关系。基本上类似于饼图，只是随时间而变化。

days = [1,2,3,4,5]  
sleeping = [7,8,6,11,7]  
eating = [2,3,4,3,2]  
working = [7,8,7,2,2]  
playing = [8,5,7,8,13]  
# 画一些空行，给予它们符合我们的堆叠图的相同颜色  
plt.plot([],[],color='m', label='Sleeping', linewidth=5)  
plt.plot([],[],color='c', label='Eating', linewidth=5)  
plt.plot([],[],color='r', label='Working', linewidth=5)  
plt.plot([],[],color='k', label='Playing', linewidth=5)  
plt.stackplot(days,sleeping,eating,working,playing, colors=['m','c','r','k'])  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Interesting Graph\nCheck it out')  
plt.legend()  
plt.show()

1. 饼图

Matplotlib 会处理切片大小以及一切事情，我们只需要提供数值

slices = [7,2,2,13]  
activities = ['sleeping','eating','working','playing']  
cols = ['c','m','r','b']  
plt.pie(slices, # 切片  
 labels=activities, #  
 colors=cols, # 颜色列表  
 startangle=90, # 指定图形的起始角度  
 shadow= *True*, # 给绘图添加一个字符大小的阴影  
 explode=(0,0.1,0,0), # 拉出第二个切片  
 autopct='%1.1f%%') # 百分比放置到图表上  
plt.title('Interesting Graph\nCheck it out')  
plt.show()

### 加载数据

1. 从文件加载

从文件中提取数据来图形化，使用选择的分隔符分割文件中的数据。loadtxt函数不要求文件是一个.txt文件，它可以是一个.csv或者 python 列表对象

# 将索引为0的元素存储到x列表，将索引为1的元素存储到y列表中  
x, y = np.loadtxt('E:\Spider\output.txt', delimiter=',', unpack=*True*)  
plt.plot(x,y, label='Loaded from file!')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Interesting Graph\nCheck it out')  
plt.legend()  
plt.show()

### basemap

1. 地理绘图

basemap投影选项URL：<https://matplotlib.org/basemap/users/mapsetup.html>

坐标需要转换，其中西经和南纬坐标是负值，北纬和东经坐标是正值。

*from* mpl\_toolkits.basemap *import* Basemap  
m = Basemap(projection='mill',  
 llcrnrlat = -90, # 左下角的纬度  
 llcrnrlon = -180, # 左下角的经度  
 urcrnrlat = 90, # 右上角的纬度  
 urcrnrlon = 180, # 右上角的经度  
 resolution='l') #分辨率 c:粗糙，l：低，h：高，f：完整  
m.drawcoastlines()  
m.drawcountries(linewidth=2) # 画出国家，线宽为2  
##m.drawstates(color='b') #蓝色线条画出州  
##m.drawcounties(color='darkred') #画出国家，黑色  
m.etopo() #模型版本  
# m.bluemarble() #模型版本  
plt.title('Basemap Tutorial')  
plt.show()

1. 绘制坐标

[Matplotlib 标记文档](http://matplotlib.org/api/markers_api.html)

m = Basemap(projection='mill',  
 llcrnrlat = 10,  
 llcrnrlon = 73,  
 urcrnrlat = 54,  
 urcrnrlon = 135,  
 resolution='l') #中国区域坐标  
m.drawcoastlines()  
m.drawcountries(linewidth=2)  
m.etopo()  
xs = []  
ys = []  
beij1, beij2 = 116.405913, 39.916237  
xpt, ypt = m(beij1, beij2)  
xs.append(xpt)  
ys.append(ypt)  
m.plot(xpt, ypt, 'r\*', markersize=15)

sh1, sh2 = 121.475941, 31.235435  
xpt, ypt = m(sh1, sh2)  
xs.append(xpt)  
ys.append(ypt)  
m.plot(xpt, ypt, 'r\*', markersize=15)  
  
m.plot(xs, ys, color='k', linewidth=3, label='Flight 98') # 直线连接  
m.drawgreatcircle(beij1, beij2, sh1, sh2, color='r', linewidth=3, label='Arc') # 弧线连接  
plt.legend(loc=4)  
plt.title('Basemap Tutorial')  
plt.show()

### 3D绘图

*from* mpl\_toolkits.mplot3d *import* axes3d

fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')