In [1]:

```
import numpy as np
n=100000
np.sum(4.0/np.r_[1:n:4,-3:-n:-4])
```

Out[1]:

```
3.1415726535897939
```

In [2]:

```
from scipy.integrate import quad#使用scipy中提供的数值积分函数quad（）计算pi
quad(lambda x:(1-x**2)**0.5,-1,1)[0]*2
```

Out[2]:

```
3.1415926535897967
```

In [11]:

```
import matplotlib as pl
x,y =np.mgrid[-2:2:500j,-2:2:500j]
z=(x**2+y**2-1)**3-x**2*y**3
```

In [54]:

```
z
```

Out[54]:

```
array([[ 375.        ,  369.93426051,  364.93295201, ...,  302.4597258
3,
         306.70071939,  311.        ],
       [ 370.06149098,  365.04162837,  360.0858048 , ...,  298.1123624
3,
         302.31395354,  306.57348892],
       [ 365.18536909,  360.21100368,  355.30028775, ...,  293.8246220
1,
         297.98716355,  302.20730876],
       ...,
       [ 365.18536909,  360.21100368,  355.30028775, ...,  293.8246220
1,
         297.98716355,  302.20730876],
       [ 370.06149098,  365.04162837,  360.0858048 , ...,  298.1123624
3,
         302.31395354,  306.57348892],
       [ 375.        ,  369.93426051,  364.93295201, ...,  302.4597258
3,
         306.70071939,  311.        ]])
```

In [19]:

```
import numpy as np
a=np.array([1,2,3,4])#array（）函数创造数组
b=np.array([5,6,7,8])
c=np.array([[1,2,3,4],[4,5,6,7],[7,8,9,10]])
print(c)
```

```
[[ 1  2  3  4]
 [ 4  5  6  7]
 [ 7  8  9 10]]
```

In [20]:

```
print(b)
```

```
[5 6 7 8]
```

In [18]:

```
c
```

Out[18]:

```
array([[ 1,  2,  3,  4],
       [ 4,  5,  6,  7],
       [ 7,  8,  9, 10]])
```

In [21]:

```
a.shape#数组的形状可以通过shape属性获得
```

Out[21]:

```
(4,)
```

In [22]:

```
c.shape
```

Out[22]:

```
(3, 4)
```

In [27]:

```
import numpy as np
c=np.array([[1,2,3,4],[4,5,6,7],[7,8,8,9]])
c.shape=4,3#不是转置，而是改变数组的每个长度
c
```

Out[27]:

```
array([[1, 2, 3],
       [4, 4, 5],
       [6, 7, 7],
       [8, 8, 9]])
```

In [28]:

```
c.shape=2,-1
c
```

Out[28]:

```
array([[1, 2, 3, 4, 4, 5],
       [6, 7, 7, 8, 8, 9]])
```

In [37]:

```
a=np.array([1,2,3,4])
d=a.reshape((2,2))#创建指定的新数组，原数组的形状保持不变，且可以用a.reshape(2,2)
d
```

Out[37]:

```
array([[1, 2],
       [3, 4]])
```

In [33]:

```
import numpy as np
a=np.array([1,2,3,4])
a[1]=100#第二个元素改成100
a
```

Out[33]:

```
array([  1, 100,   3,   4])
```

In [34]:

```
d=a.reshape(2,2)
```

In [35]:

```
d
```

Out[35]:

```
array([[  1, 100],
       [  3,   4]])
```

In [38]:

```
c.dtype#元素类型通过dtype获得
```

Out[38]:

```
dtype('int64')
```

In [40]:

```
ai32=np.array([1,2,3,4],dtype=np.int32)#用dtype参数创建数组时指定元素类型
af=np.array([1,2,3,4],dtype=float)
ac=np.array([1,2,3,4],dtype=complex)
```

In [42]:

```
ai32.dtype
```

Out[42]:

```
dtype('int32')
```

In [43]:

```
af.dtype
```

Out[43]:

```
dtype('float64')
```

In [44]:

```
ac.dtype
```

Out[44]:

```
dtype('complex128')
```

In [45]:

```
[key for key,value in np.typeDict.items()if value is np.float64]#获得float64类型对应
```

Out[45]:

```
['double', 'd', 12, 'float64', 'Float64', 'f8', 'float_', 'float']
```

In [47]:

```
set(np.typeDict.values())
```

Out[47]:

```
{numpy.bool_,
 numpy.bytes_,
 numpy.complex128,
 numpy.complex256,
 numpy.complex64,
 numpy.datetime64,
 numpy.float128,
 numpy.float16,
 numpy.float32,
 numpy.float64,
 numpy.int16,
 numpy.int32,
 numpy.int64,
 numpy.int64,
 numpy.int8,
 numpy.object_,
 numpy.str_,
 numpy.timedelta64,
 numpy.uint16,
 numpy.uint32,
 numpy.uint64,
 numpy.uint64,
 numpy.uint8,
 numpy.void}
```

In [48]:

```
c.dtype.type
```

Out[48]:

```
numpy.int64
```

In [49]:

```
a=np.int16(200)
a*a
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Runtim
eWarning: overflow encountered in short_scalars
```

Out[49]:

```
-25536
```

In [50]:

```
v1=3.14
v2=np.float64(v1)
%timeit v1*v1
% timeit v2*v2
```

```
The slowest run took 32.58 times longer than the fastest. This could m
ean that an intermediate result is being cached.
10000000 loops, best of 3: 44.3 ns per loop
The slowest run took 35.80 times longer than the fastest. This could m
ean that an intermediate result is being cached.
10000000 loops, best of 3: 95.6 ns per loop
```

In [51]:

```
t1=np.array([1,2,3,4],dtype=np.float)#astype可以对数组的元素类型进行转化
t2=t1.astype(np.int32)
```

In [52]:

```
t2
```

Out[52]:

```
array([1, 2, 3, 4], dtype=int32)
```

In [53]:

```
t1
```

Out[53]:

```
array([ 1.,  2.,  3.,  4.])
```