In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from pandas import Series,DataFrame
```

# 一 numpy的方法

In [2]:
```
#1 np.ones #
arr = np.ones((1,10))# 1行10列
arr
```

Out[2]:
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

In [3]:
```
# 2 np.zeros
np.zeros((10,1))# 10行1列 数值为0
```

Out[3]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

In [4]:
```
# np.full((3,3),fill_value=9)  # 三行三列个9
np.full((3,3),fill_value=9)
```

Out[4]:
```
array([[9, 9, 9],
       [9, 9, 9],
       [9, 9, 9]])
```

In [5]:
```
# np.linspace()
np.linspace(1,100,20) # 1-100 分为20个数 自动计算
```

Out[5]:
```
array([  1.        ,   6.21052632,  11.42105263,  16.63157895,
        21.84210526,  27.05263158,  32.26315789,  37.47368421,
        42.68421053,  47.89473684,  53.10526316,  58.31578947,
        63.52631579,  68.73684211,  73.94736842,  79.15789474,
        84.36842105,  89.57894737,  94.78947368, 100.        ])
```

In [6]:
```
# np.arange()
np.arange(0,100,2) # 从0-100,步距为 2 #左闭右开
```

Out[6]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
       34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66,
       68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98])
```

In [7]:
```
# np.random.randint()
np.random.seed(100) # 设置时间种子 让随机不在随机
np.random.randint(0,100,size=(3,3)) #3行3列 数值都是随机数
```

Out[7]:
```
array([[ 8, 24, 67],
       [87, 79, 48],
       [10, 94, 52]])
```

In [8]:
```
# np.random.randn 标准正太分布
np.random.randn(2,5)
```

Out[8]:
```
array([[ 0.35467445, -0.78606433, -0.2318722 ,  0.20797568,  0.93580797],
       [ 0.17957831, -0.5771615 , -0.53337271, -0.22540212, -0.31491934]])
```

In [9]:
```
# np.random.random(size=None)
# 生成0到1的随机数
np.random.random(size=(5,5))
```

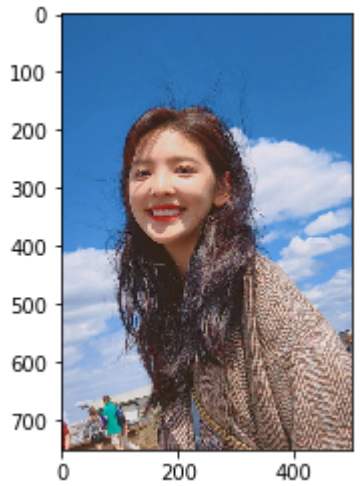
Out[9]:
```
array([[0.97862378, 0.81168315, 0.17194101, 0.81622475, 0.27407375],
       [0.43170418, 0.94002982, 0.81764938, 0.33611195, 0.17541045],
       [0.37283205, 0.00568851, 0.25242635, 0.79566251, 0.01525497],
       [0.59884338, 0.60380454, 0.10514769, 0.38194344, 0.03647606],
       [0.89041156, 0.98092086, 0.05994199, 0.89054594, 0.5769015 ]])
```

# 二. ndarray(多维数组的属性)

4个比记参数 ndim维度 shape:形状(各维度的长度) size总长度 dtype:元素类型

In [10]:
```
img_arr = plt.imread('./gsx.jpg') # imread 读取图片方法
plt.imshow(img_arr)  # 展示图片方法
```

Out[10]:
```
<matplotlib.image.AxesImage at 0x26e226369e8>
```



In [11]:
```
img_arr.ndim  # ** 维度
```

Out[11]: 3

In [12]:
```
img_arr.size # 大小
```

Out[12]: 1125000

In [13]:
```
img_arr.shape # 像素吧
```

Out[13]: (750, 500, 3)

In [14]:
```
img_arr.dtype # 元素类型
```

Out[14]: dtype('uint8')

# 三 ndarray(多维数组)的基本操作

1 索引
2 切片
3 变形
    - 图片倒置
4 级联
    - 图片九宫格
5 切分
6 副本
    - ndarray.copy()

## 索引

- 一维与列表完全一致 多维时同理

In [15]:
```
arr = np.random.randint(1,100,size=(5,4))
arr
```

Out[15]:
```
array([[14, 31, 18, 54],
       [69, 51, 92, 92],
       [84, 54, 79,  1],
       [14, 58, 77,  4],
       [71,  4, 85, 80]])
```

In [16]:
```
arr[1,2]  #一行两列 的值
```

Out[16]:
```
92
```

In [17]:
```
arr[1][2] # 先取出第一行 然后再取索引为二的值
```

Out[17]:
```
92
```

In [18]:
```
arr[1] = 1000 # 索引为一 的行  数值全部改为1000
arr
```

Out[18]:
```
array([[  14,   31,   18,   54],
       [1000, 1000, 1000, 1000],
       [  84,   54,   79,    1],
       [  14,   58,   77,    4],
       [  71,    4,   85,   80]])
```

## 2.切片

- 一维与列表完全一致,多维同理

In [19]:
```
arr[0:2]  # 获取前两行的数据
```

Out[19]:
```
array([[  14,   31,   18,   54],
       [1000, 1000, 1000, 1000]])
```

In [20]:
```
arr[:,0:2] # 获取前两列的数据
```

Out[20]:
```
array([[  14,   31],
       [1000, 1000],
       [  84,   54],
       [  14,   58],
       [  71,    4]])
```

In [21]:
```
arr[0:2,0:2]  #获取二维数组前两行和前两列数据
```

Out[21]:
```
array([[  14,   31],
       [1000, 1000]])
```

将数据反转，例如[1,2,3]---->[3,2,1] :: 进行切片

In [22]:
```
arr = np.array([1,2,3])
arr
```

Out[22]:
```
array([1, 2, 3])
```

In [23]:
```
arr = arr[::-1] # ::-1进行倒序操作
arr
```

Out[23]:
```
array([3, 2, 1])
```

多维数组

- 行倒序 arr[::-1]
- 列倒序 arr[:,::-1]
- 全部倒序 arr[::-1,::-1]

## 3. 变形

使用arr.reshape() 函数,注意参数是一个tuple()!

- 基本使用

In [24]:
```
#1.将一维数组变形成多维数组
arr_new = arr.reshape((3,1))
arr_new
```

Out[24]:
```
array([[3],
       [2],
       [1]])
```

In [25]:
```
#2.将多维数组变形成一维数组
arr_new = arr_new.reshape((3))
arr_new
```

Out[25]:
```
array([3, 2, 1])
```

In [26]:
```
# -1 表示自动计算行数
arr_new.reshape(-1,1)
```

Out[26]:
```
array([[3],
       [2],
       [1]])
```

# 图片倒置

In [27]:
```
img_arr.shape
```

Out[27]:
```
(750, 500, 3)
```

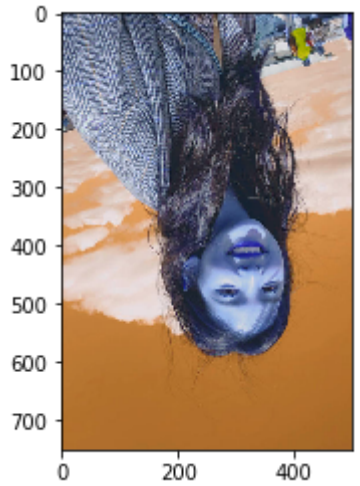
In [28]:
```
# 将数据变为 一维
img_arr_new = img_arr.reshape((750*500*3))
```

In [29]:
```
# 倒置
img_arr_new = img_arr_new[::-1]
```

In [30]:
```
# 变为原来的 形状 shape
img_arr_new = img_arr_new.reshape((750,500,3))
```

In [31]:
```
plt.imshow(img_arr_new)
```

Out[31]:
```
<matplotlib.image.AxesImage at 0x26e226f7978>
```




## 4.级联

- np.concatenate()

In [32]:
```
arr1 = np.random.randint(1,100,size=(3,3))
arr1
```

Out[32]:
```
array([[11, 88, 61],
       [ 4, 49, 53],
       [44, 37,  6]])
```

In [33]:
```
arr2 = np.random.randint(1,100,size=(3,3))
arr2
```

Out[33]:
```
array([[72, 39, 87],
       [95, 99, 43],
       [85, 96, 77]])
```

In [34]:
```
np.concatenate((arr1,arr2),axis=1) #横着 加 级联
```

Out[34]:
```
array([[11, 88, 61, 72, 39, 87],
       [ 4, 49, 53, 95, 99, 43],
       [44, 37,  6, 85, 96, 77]])
```

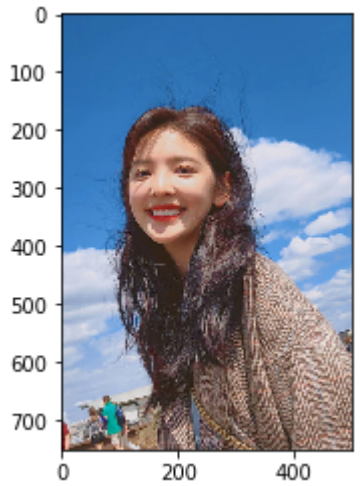
In [35]:
```
np.concatenate((arr1,arr2),axis=0) #竖着加 级联
```

Out[35]:
```
array([[11, 88, 61],
       [ 4, 49, 53],
       [44, 37,  6],
       [72, 39, 87],
       [95, 99, 43],
       [85, 96, 77]])
```

- 图片九宫格

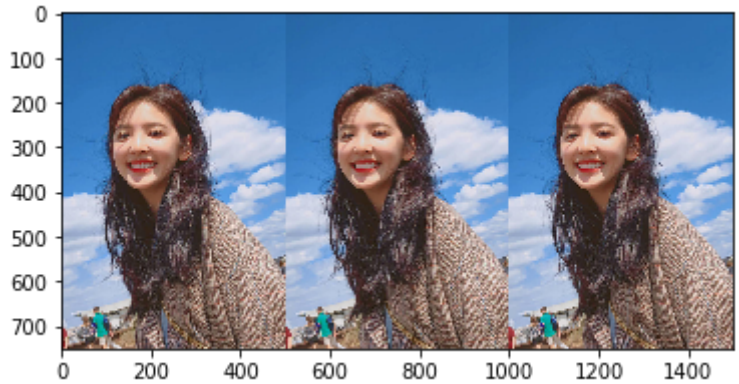
In [36]:
```
plt.imshow(img_arr)
```

Out[36]:
```
<matplotlib.image.AxesImage at 0x26e2275e390>
```




In [37]:
```
heng_three = np.concatenate((img_arr,img_arr,img_arr),axis=1) #横着加三张
plt.imshow(heng_three)
```

Out[37]:
```
<matplotlib.image.AxesImage at 0x26e24f781d0>
```



In [38]:
```
nine_img = np.concatenate((heng_three,heng_three,heng_three),axis=0) # 竖着加三次
plt.imshow(nine_img)
```

Out[38]: <matplotlib.image.AxesImage at 0x26e2532d3c8>


np.hstack 与 np.vstack (横,竖)

In [39]:
```
plt.imshow(np.hstack((nine_img,nine_img)))  # hstack 横着加 h横
```

Out[39]:
```
<matplotlib.image.AxesImage at 0x26e27a5d2e8>
```


级联需要注意的点:

- 级联的参数是列表：一定要加中括号或小括号
- 维度必须相同
- 形状相符:在维度保持一致的前提下，如果进行横向（axis=1）级联，必须保证进行级联的数组行数保持一致。如果进行纵向（axis=0）级联，必须保证进行级联的数组列数保持一致。
- 可通过axis参数改变级联的方向

## 5.切分

与级联类似,三个函数完成切分工作

- np.split(arr,行 / 列号，轴):参数2是一个列表类型
- np.vsplit
- np.hsplit

In [40]:
```
arr3 = np.random.randint(1,100,size=(6,5))
arr3
```

Out[40]:
```
array([[34, 59, 43, 23,  1],
       [56, 99, 20, 54, 69],
       [63, 51, 69, 36, 24],
       [10, 49, 22, 26, 55],
       [ 7, 38, 59, 40, 95],
       [52, 31, 67, 25, 56]])
```

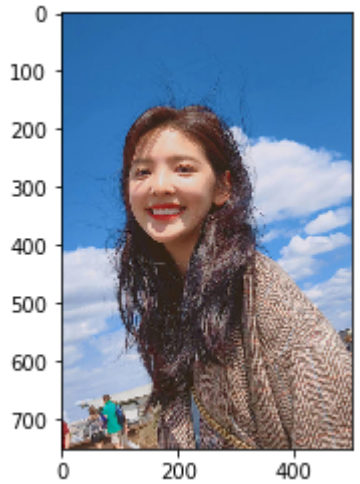
In [41]:
```
np.split(arr3,(2,4),axis=1)  # 在2列 4列 前面切
```

Out[41]:
```
[array([[34, 59],
        [56, 99],
        [63, 51],
        [10, 49],
        [ 7, 38],
        [52, 31]]), array([[43, 23],
        [20, 54],
        [69, 36],
        [22, 26],
        [59, 40],
        [67, 25]]), array([[ 1],
        [69],
        [24],
        [55],
        [95],
        [56]])]
```

- 切分照片

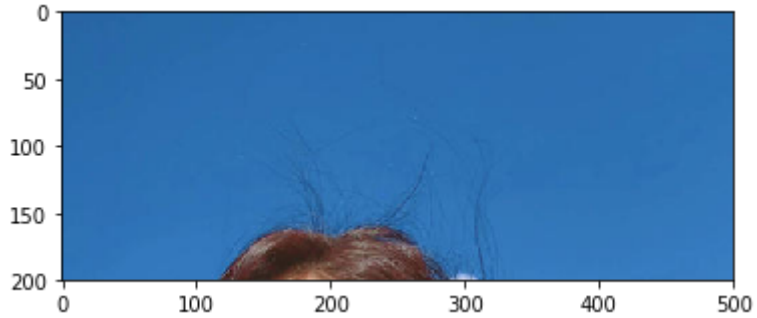
In [42]:
```
# 读取照片 imread('路径')
plt.imshow(img_arr)
```

Out[42]: <matplotlib.image.AxesImage at 0x26e26724898>




In [43]:
```
img_s = np.split(img_arr,(200,),axis=0)[0] # 200行
plt.imshow(img_s)
```

Out[43]: <matplotlib.image.AxesImage at 0x26e2689d7b8>




## 6.副本

- 所有赋值运算不会为ndarray的任何元素创建副本。对赋值后的对象的操作也对原来的对象生效。

- 可使用copy()函数创建副本

In [44]:
```
arr_ = arr.copy()
arr_
```

Out[44]:
```
array([3, 2, 1])
```

In [45]:
```
arr_[2] = 11111
arr_
```

Out[45]:
```
array([    3,     2, 11111])
```

# 四 ndarray的聚合操作

## 1.求和 np.sum

In [46]:
```
arr_sum = np.random.randint(1,100,size=(3,3))
arr_sum
```

Out[46]:
```
array([[18, 46, 90],
       [46, 36, 17],
       [13, 87, 83]])
```

In [47]:
```
arr_sum.sum(axis= 1) #每一行的和
```

Out[47]:
```
array([154,  99, 183])
```

In [48]:
```
arr_sum.sum(axis = 0) #每一列的和
```

Out[48]:
```
array([ 77, 169, 190])
```

## 2.最大最小值: np.max/np.min

In [49]:
```
arr_sum.max(axis=0)  # 每一列的 最大值
```

Out[49]:
```
array([46, 87, 90])
```

In [50]:
```
arr_sum.min(axis=1) # 每一行的最小值
```

Out[50]:
```
array([18, 17, 13])
```

### 3.平均值:np.mean()

In [51]:
```
arr_sum
```

Out[51]:
```
array([[18, 46, 90],
       [46, 36, 17],
       [13, 87, 83]])
```

In [52]:
```
arr_sum.mean(axis=0) # 每一列的平均值
```

Out[52]:
```
array([25.66666667, 56.33333333, 63.33333333])
```

In [53]:
```
arr_sum.std(axis=0) # 每一列的方差
```

Out[53]:
```
array([14.52201394, 22.06555888, 32.88701196])
```

### 其他聚合操作

```
Function Name    NaN-safe Version    Description
np.sum    np.nansum    Compute sum of elements
np.prod    np.nanprod    Compute product of elements
np.mean    np.nanmean    Compute mean of elements
np.std    np.nanstd    Compute standard deviation
np.var    np.nanvar    Compute variance
np.min    np.nanmin    Find minimum value
np.max    np.nanmax    Find maximum value
np.argmin    np.nanargmin    Find index of minimum value
np.argmax    np.nanargmax    Find index of maximum value
np.median    np.nanmedian    Compute median of elements
np.percentile    np.nanpercentile    Compute rank-based statistics of elements
np.any    N/A    Evaluate whether any elements are true
np.all    N/A    Evaluate whether all elements are true
np.power 幂运算
```

# 五 广播机制

【重要】ndarray广播机制的三条规则:缺失维度的数组将维度补充为进行运算的数组的维度。缺失的数组元素使用已有元素进行补充。

- 规则一：为缺失的维度补1(进行运算的两个数组之间的维度只能相差一个维度)
- 规则二：缺失元素用已有值填充
- 规则三：缺失维度的数组只能有一行或者一列

例1： m = np.ones((2, 3)) a = np.arange(3) 求m+a

In [54]:
```
m = np.ones((2,3))
a = np.arange(3)
m+a
```

Out[54]:
```
array([[1., 2., 3.],
       [1., 2., 3.]])
```

例2： a = np.arange(3).reshape((3, 1)) b = np.arange(3) 求a+b

In [55]:
```
a = np.arange(3).reshape((3,1))
b = np.arange(3)
a+b
```

Out[55]:
```
array([[0, 1, 2],
       [1, 2, 3],
       [2, 3, 4]])
```

In [56]:
```
#习题3:
a = np.ones((4, 1))
b = np.arange(4)
a+b
```

Out[56]:
```
array([[1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.]])
```

# 六 ndarray的排序

## 1. 快速排序

np.sort()与ndarray.sort()都可以，但有区别：

- np.sort()不改变输入
- ndarray.sort()本地处理，不占用空间，但改变输入

In [57]:
```
np.random.seed(10)
arr = np.random.randint(1,100,size=(5,5))
arr
```

Out[57]:
```
array([[10, 16, 65, 29, 90],
       [94, 30,  9, 74,  1],
       [41, 37, 17, 12, 55],
       [89, 63, 34, 73, 79],
       [50, 52, 55, 78, 70]])
```

In [58]:
```
np.sort(arr,axis=0) # 0是竖着排 1是横着排
```

Out[58]:
```
array([[10, 16,  9, 12,  1],
       [41, 30, 17, 29, 55],
       [50, 37, 34, 73, 70],
       [89, 52, 55, 74, 79],
       [94, 63, 65, 78, 90]])
```

In [59]:
```
arr # np.sort() 不改变输入
```

Out[59]:
```
array([[10, 16, 65, 29, 90],
       [94, 30,  9, 74,  1],
       [41, 37, 17, 12, 55],
       [89, 63, 34, 73, 79],
       [50, 52, 55, 78, 70]])
```

In [60]:
```
arr.sort(axis=1) # 0是竖着排 1是横着排
```

In [61]:
```
arr # arr.sort() 改变输入
```

Out[61]:
```
array([[10, 16, 29, 65, 90],
       [ 1,  9, 30, 74, 94],
       [12, 17, 37, 41, 55],
       [34, 63, 73, 79, 89],
       [50, 52, 55, 70, 78]])
```

In [ ]:
```

```