Numpy索引和切片

索引和切片：

1. 获取某行的数据：

```
# 1. 如果是一维数组
a = np.arange(12)
print(a[1]) #获取下标为1的元素(从0开始计数)

# 2.如果是多维数组
a1 = a.reshape((3,4))
print(a1[1]) #获取下标为1的行的数据(从0开始计数)

# 3. 如果是三维数组
a2 = a.reshape(2,3,2)
print(a2[1])  #获取第1块的元素(从0开始计数)
```

2. 连续获取某几行的数据：

```
# 1. 获取连续的几行的数据
a1 = np.arange(0,24).reshape((4,6))
print(a1[0:2]) #获取0行到1行的数据(左闭右开)

# 2. 获取不连续的几行的数据
print(a1[[0,2,3]]) #获取第0、2、3行的数据

# 3. 也可以使用负数进行索引
print(a1[[-1,-2]]) #获取倒数第1行、倒数第2行的数据

# 4.数据翻转
print(a[::-1])  #行翻转
print(a[:,::-1])  #列翻转
print(a[::-1,::-1]) #数组所有元素翻转
```

3. 获取某行某列的数据：

```
a1 = np.arange(0,24).reshape((4,6))
print(a1[1,1]) #获取1行1列的数据

print(a1[0:2,0:2]) #获取0-1行的0-1列的数据
print(a1[[1,2],[2,3]]) #获取(1,2)和(2,3)的两个数据
```

> *注：在上面的例子中，我们利用了整数数组进行索引，这种索引方式称为"花式索引"，这里的整数数组可以是`Numpy`的`ndarray`，也可以是`Python`的`list`、`tuple`等可迭代类型，花式索引也可以使用正向或负向索引*

4. 获取某列的数据：

```
a1 = np.arange(0,24).reshape((4,6))
print(a1[:,1]) #获取第1列的数据
```

案例：通过数组切片处理图像

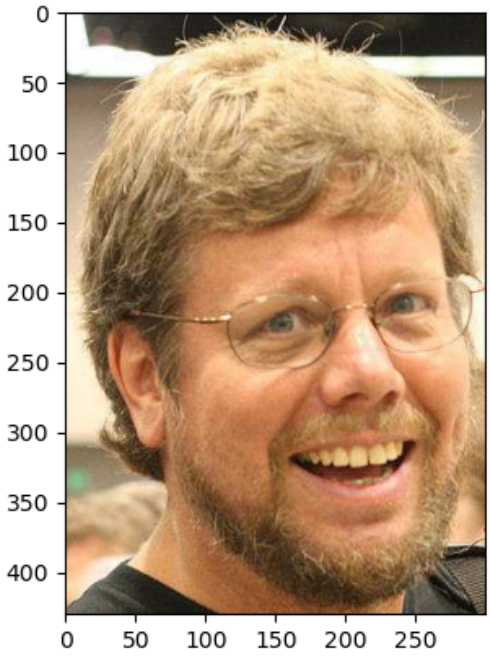
学习基础知识总是比较枯燥且没有成就感的，所以我们还是来个案例为大家演示下上面学习的数组索引和切片操作到底有什么用。前面我们说到过，可以用三维数组来表示图像，那么通过图像对应的三维数组进行操作，就可以实现对图像的处理，如下所示。

读入图片创建三维数组对象

```
import matplotlib.pyplot as plt  #需要导入matplotlib库
img_path = "res/guido.jpg"  #图片所在路径
guido_img = plt.imread(img_path)   #通过imread方法获取图片，生成一个图片三维数组
plt.imshow(guido_img)  #显示图像
plt.show()  #显示在窗口
```

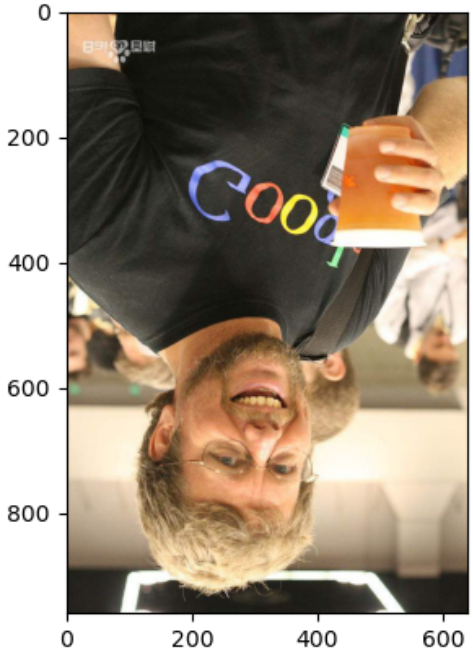
把Guido的头切出来

```
plt.imshow(guido_img[20:450,100:400])
plt.show()
```



将图像进行垂直翻转，实际上就是对数组的"行"进行反向切片

```
plt.imshow(guido_img[::-1])
plt.show()
```



将图像进行水平翻转，实际上就是对数组的"列"进行反向切片

```
plt.imshow(guido_img[:,::-1])
plt.show()
```

将图像进行反色处理，实际上就是将原图像RGB通道改为BGR通道

```
plt.imshow(guido_img[:,:,::-1])
plt.show()
```

> *小结*
>
> 1. *如果数组是一维的，那么索引和切片与`python`的列表一致*
> 2. *如果是多维的，那么在中括号中，两个值通过逗号分隔，逗号前面是行，逗号后面是列，如果只有一个值，那么代表的是行*

布尔索引：

布尔索引就是通过布尔类型的数组对数组元素进行索引，布尔类型的数组可以手动构造，也可以通过关系运算来产生布尔类型的数组。布尔运算也是矢量的，比如以下代码：

```
a1 = np.arange(0,24).reshape((4,6))
print(a1<10) #会返回一个新的数组，这个数组中的值全部都是bool类型
> [[ True  True  True  True  True  True]
 [ True  True  True  True False False]
 [False False False False False False]
 [False False False False False False]]
```

这样看上去没有什么用，假如我现在要实现一个需求，要将`a1`数组中所有小于10的数据全部都提取出来。那么可以使用以下方式实现：

```
a1 = np.arange(0,24).reshape((4,6))
a2 = a1 < 10
print(a1[a2]) #这样就会在a1中把a2中为True的元素对应的位置的值提取出来
```

其中布尔运算可以有`!=`、`==`、`>`、`<`、`>=`、`<=`以及`&(与)`和`|(或)`。示例代码如下：

```
a1 = np.arange(0,24).reshape((4,6))
a2 = a1[(a1 < 5) | (a1 > 10)]
print(a2)
```

> *提示：切片操作虽然创建了新的数组对象，但是新数组和原数组共享了数组中的数据，简单的说，如果通过新数组对象或原数组对象修改数组中的数据，其实修改的是同一块数据。花式索引和布尔索引也会创建新的数组对象，而且新数组复制了原数组的元素，新数组和原数组并不是共享数据的关系*

值的替换：

利用索引，也可以做一些值的替换。把满足条件的位置的值替换成其他的值。比如以下代码：

```
a1 = np.arange(0,24).reshape((4,6))
a1[3] = 0 #将第三行的所有值都替换成0
print(a1)
```

也可以使用条件索引来实现：

```
a1 = np.arange(0,24).reshape((4,6))
a1[a1 < 5] = 0 #将小于5的所有值全部都替换成0
print(a1)
```

还可以使用函数来实现：

```
# where函数：
a1 = np.arange(0,24).reshape((4,6))
a2 = np.where(a1 < 10,1,0) #把a1中所有小于10的数全部变成1，其余的变成0
print(a2)
```