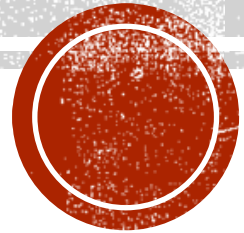
遥感数字图像处理实验课

图像基础入门

李荣昊

2021年10月



自我介绍

姓名：李荣昊

实验室：遥感楼505

邮箱：ronghao.li@stu.pku.edu.cn

（作业请提交邮箱，课代表收齐以后统一发送）

主要内容

- 课程介绍
- 开发环境介绍与配置教程
- Python 图像基本操作

课程介绍

- 课程目的
  - 了解数字图像的常用格式
  - 学习并实现基本图像处理方法
  - 图像-深度学习基础入门
- 课程要求
  - 掌握基本的图像处理算法，具备一定的实现能力
  - 按时完成实验报告及编程作业

课程介绍

- 先修内容
  - Python基本操作
  - 数据结构与算法
  - Jupyter lab/Jupyter notebook的使用
- 后续内容
  - Pytorch基本操作
  - Kaggle/Colab的使用

课程介绍

- 作业形式

  - 个人作业（小作业）：

    编程实现简单的图像处理算法；
    提交时间：下一次实验课上课前一天晚上12点之前。
    命名方式：学号_姓名_第X次实验。
    邮箱：ronghao.li@stu.pku.edu.cn（以收到回复为准）

  - 小组作业（大作业）：

    编程实现某个复杂图像处理需求
    编程实现某个简单的深度学习算法
    提交时间：待定，一般为结课时

开发环境介绍与使用

- 开发环境-课程demo&作业提交
  - Anaconda
  - Python3
  - Jupyter-notebook/Jupyter-lab
  - Pillow/Opencv
  - Numpy
  - Pytorch以及相关环境（CUDA）

开发环境介绍与使用

- Conda的基本操作
  - **获取版本号**
    conda --version 或 **conda –V**
  - **获取帮助**
    conda --help 或 **conda -h**

    查看某一命令的帮助，如update命令及remove命令
    conda update --help 或 **conda remove –help**
  - **环境管理**
    查看环境管理的全部命令帮助
    conda env –h
    创建环境
    conda create --name your_env_name

开发环境介绍与使用

- Conda基本操作
  - **环境管理**

    创建制定python版本的环境

    conda create --name your_env_name python=2.7

    conda create --name your_env_name python=3

    conda create --name your_env_name python=3.5

    创建包含某些包的环境

    conda create --name your_env_name numpy scipy

    创建指定python版本下包含某些包的环境

    conda create --name your_env_name python=3.5 numpy scipy

开发环境介绍与使用

- Conda基本操作
  - **环境管理**

    列举当前所有环境

    conda info --envs

    conda env list

    进入某个环境

    activate your_env_name

    退出当前环境

    deactivate

    复制某个环境

    conda create --name new_env_name --clone old_env_name

开发环境介绍与使用

- Conda基本操作
  - **环境管理**

    删除某个环境

    conda remove --name your_env_name --all

  - **分享环境**

    首先通过**activate target_env**要分享的环境**target_env**，然后输入下面的命令会在相应路径上生成一个**environment.yml**文件，

    conda env export > G:\\environment.yml

    小伙伴拿到**environment.yml**文件后，将该文件放在工作目录下，可以通过以下命令从该文件创建环境

    conda env create -f environment.yml

开发环境介绍与使用

- Conda基本操作
  - 分享环境

```yaml
name: my_pytorch
channels:
- pytorch
- defaults
dependencies:
- matplotlib=3.2.2=0
- matplotlib-base=3.2.2=py37h64f37c6_0
- mkl=2020.0=166
- mkl-service=2.3.0=py37hb782905_0
- mkl_fft=1.0.15=py37h14836fe_0
- mkl_random=1.1.0=py37h675688f_0
- ninja=1.9.0=py37h74a9793_0
- numpy=1.18.1=py37h93ca92e_0
- numpy-base=1.18.1=py37hc3f5095_1
- olefile=0.46=py_0
- openssl=1.1.1f=he774522_0
- pandas=1.1.1=py37ha925a31_0
- pillow=7.0.0=py37hcc1f983_0
- pip=20.0.2=py37_1
- pycparser=2.20=py_0
prefix: E:\software\Miniconda3\envs\my_pytorch
```

environment.yml示例

开发环境介绍与使用

- Conda基本操作
  - **包管理**

    列举当前活跃环境下的所有包

    conda list

    列举一个非当前活跃环境下的所有包

    conda list -n your_env_name

    为指定环境安装某个包

    conda install -n env_name package_name

开发环境介绍与使用

- Jupyter lab

- JupyterLab是Jupyter主打的最新数据科学生产工具，某种意义上，它的出现是为了取代Jupyter Notebook。不过不用担心Jupyter Notebook会消失，JupyterLab包含了Jupyter Notebook所有功能。

  JupyterLab作为一种基于web的集成开发环境，你可以使用它编写notebook、操作终端、编辑markdown文本、打开交互模式、查看csv文件及图片等功能。

- 我们课堂的演示一般以jupyter lab为主，但是这个用jupyter notebook也可以。

开发环境介绍与使用

- Jupyter notebook/lab基本操作
  - **管理**

    激活服务

    jupyter notebook/lab

    选择打开位置：

    jupyter lab D://xxxxxxxxxx

    中断服务

    双击（猛按） **Control-C**

开发环境介绍与使用

- Pytorch简介
  - http://pytorch123.com/FirstSection/PyTorchIntro/

基本图像格式

- 常见图像格式

计算机以位图和矢量图的格式显示图像

➢ 矢量图

矢量图（Vector）：矢量图，也称为面向对象的图像或绘图图像，在数学上定义为一系列由线连接的点。矢量文件中的图形元素称为对象。每个对象都是一个自成一体的实体，它具有颜色、形状、轮廓、大小和屏幕位置等属性。

优点： 文件数据量小、图像质量与分辨率无关

缺点： 不易制作色彩丰富的图像

基本图像格式

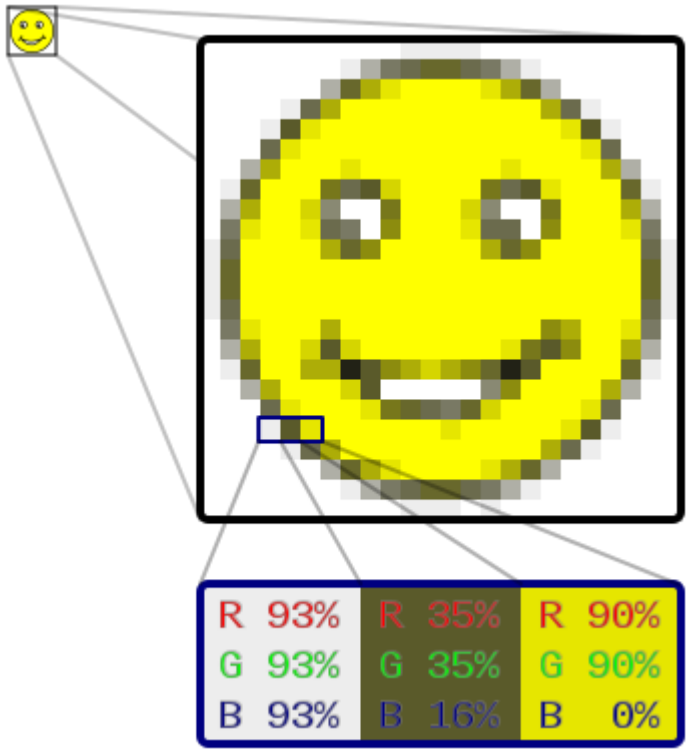
- 常见图像格式

➢ 位图文件

位图（Bitmap）：图像又称点阵图或光栅图，它使用我们称为像素(象素，Pixel)的一格一格的小点来描述图像。

常见的位图文件格式有BMP、JPEG、GIF和PNG。

编码方式：灰度图（黑白），RGB，CMYK……



通常使用8位无符号整形存储

基本图像格式

- 常见图像格式

  - 位图文件打开方式

1.直接从二进制文件打开,读取元数据、数据数组
需要预先知道结构信息
有兴趣的同学可以尝试一下

2.利用已有的库(如pillow,opencv),直接打开到一个结构体

```python
#python code
from PIL import Image
im_PIL = Image.open("../img/horse.jpeg")#打开图像，注意打开路径
im_PIL.show()#显示图像
```

基本图像格式

- **TIF文件**

➢标记图像文件格式TIF (Tag Image File Format)， 它是现存图像文件格式中最复杂的一种，它提供存储各种信息的完备的手段， 可以存储专门的信息而不违反格式宗旨，是目前流行的图像文件交换标准之一。

➢TIF格式文件的设计考虑了扩展性、 方便性和可修改性，因此非常复杂，要求用更多的代码来控制它，结果导致文件读写速度慢， TIF代码也很长。

➢TIF文件由文件头、参数指针表与参数域、参数数据表和图像数据4部分组成。

基本图像格式

- JPEG文件

- JPEG(Joint Photographer's Experts Group)格式即联合图像专家组, 是由ISO和CCITT为静态图像所建立的第一个国际数字图像压缩标准，主要是为了解决专业摄影师所遇到的图像信息过于庞大的问题。
- 由于JPEG的高压缩比和良好的图像质量， 使得它广泛应用于多媒体和网络程序中。JPEG 格式支持 24 位颜色，并保留照片和其他连续色调图像中存在的亮度和色相的显著和细微的变化。
- JPEG 通过有选择地减少数据来压缩文件大小，因为它会弃用数据，故 JPEG 压缩为有损压缩。较高品质设置导致弃用的数据较少，但是 JPEG 压缩方法会降低图像中细节的清晰度， 尤其是包含文字或矢量图形的图像

基本图像格式

- BMP文件

➢BMP（Bitmap）文件是windows系统中的标准图像文件格式，Windows的图形用户界面（graphical user interfaces）也在它的内建图像子系统GDI中对BMP格式提供了支持。BMP位图文件默认的文件扩展名是.BMP，有时也会以.DIB或.RLE作扩展名。

➢BMP与其它位图格式相比，信息丰富，几乎不压缩，由此导致占用空间较大

➢windows 3.0以后的BMP文件与显示设备无关，称为设备无关位图DIB (Device-Independent Bitmap)

基本图像格式

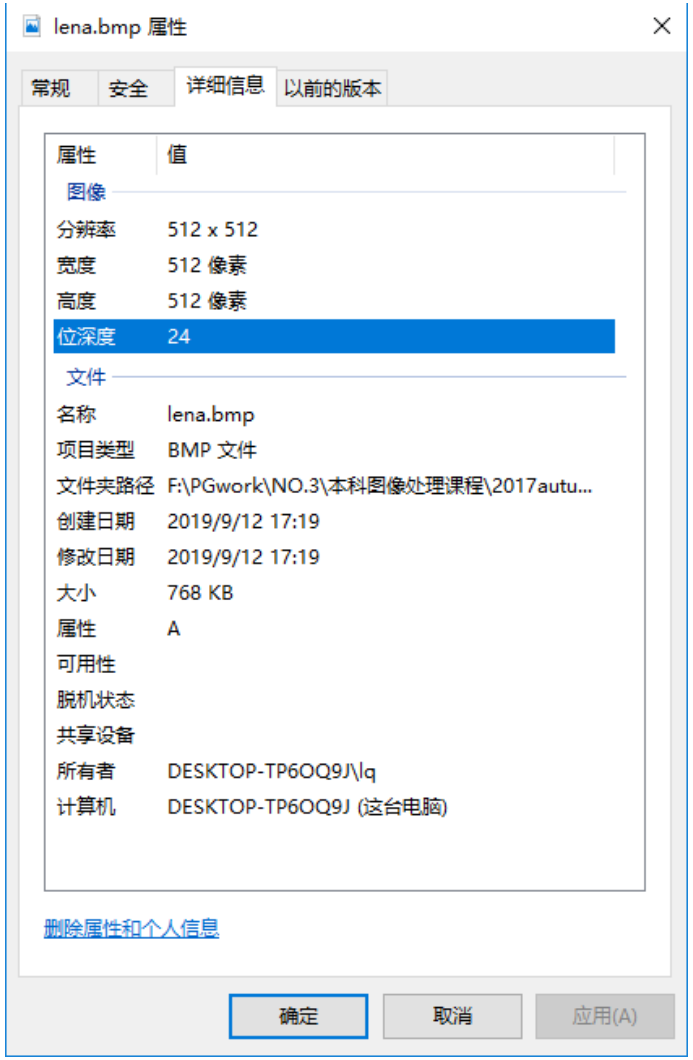
- BMP文件的基本格式
  - 位图头文件(bitmap-file header)
  - 位图信息头(bitmap-information header)
  - 颜色表(color table)
  - 颜色点阵数据(bits data)

☆ 24位真彩色位图没有颜色表，所以其只有另外三部分

基本图像格式

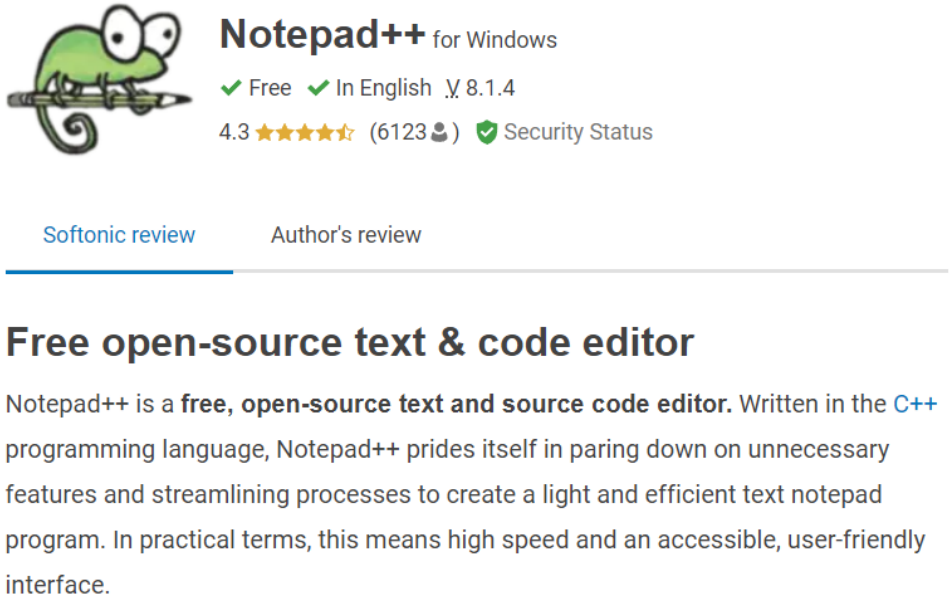
- BMP文件的基本格式

Lena.bmp



基本图像格式

- 轻量级编辑器--Notepad++



Notepad++:
https://notepad-plus.en.softonic.com/

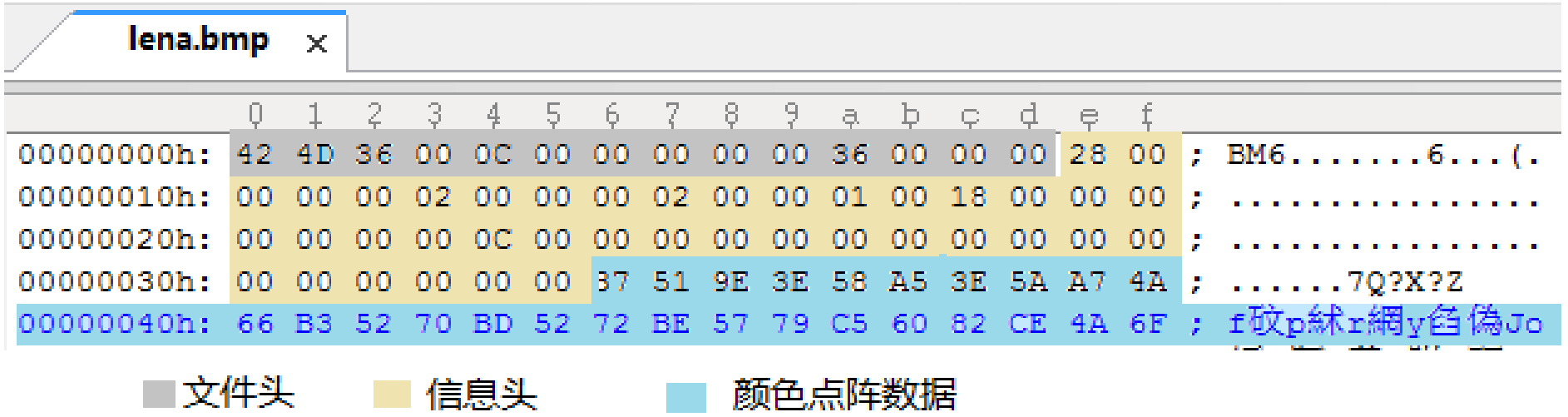
HexEditor:
https://github.com/chcg/NPP_HexEdit/releases

打开notepad++:
设置——导入——导入插件
重启notepad++——插件

基本图像格式

- BMP文件的基本格式

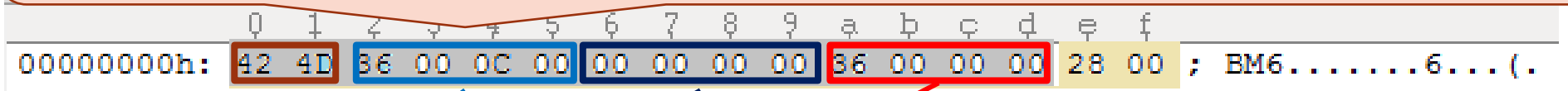
用UltraEdit或者notepad++打开lena.bmp，可以看到这个文件的全部数据如下图所示：



基本图像格式

- BMP文件的基本格式
  - 位图头文件

注意，Windows的数据是倒着念的，这是PC电脑的特色。如果一段数据为50 1A 25 3C，倒着念就是3C 25 1A 50，即0x3C251A50。因此，如果bfSize的数据为36 00 0C 00，实际上就成了0x000C0036，也就是0xC0036。（$12 * 16^4 + 3 * 16 + 6 = 786486$）



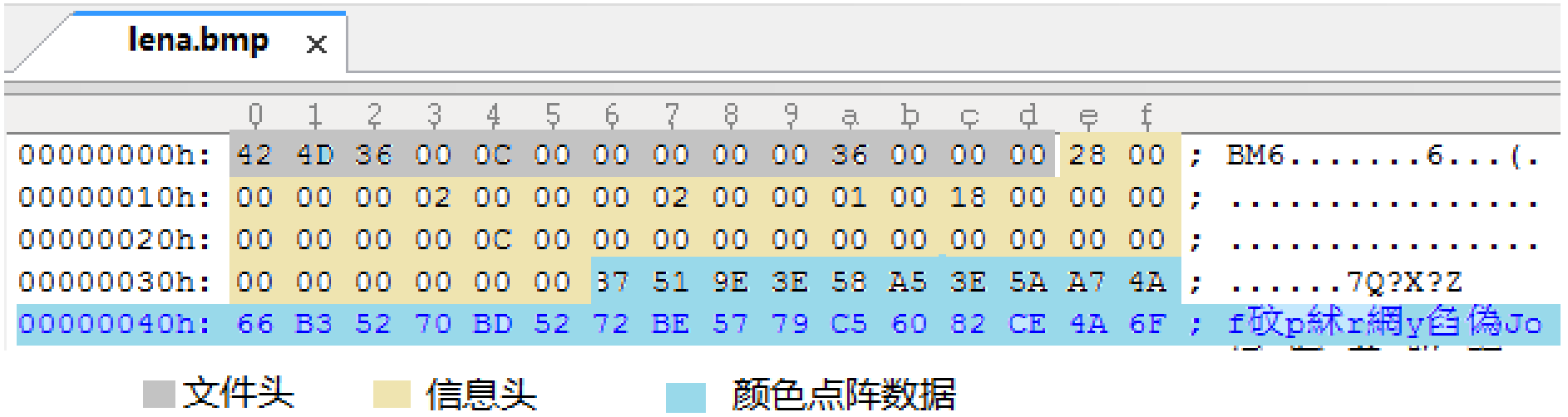
名称	占用空间	内容	实际数据
bfType	2字节	标识，就是“BM”二字	BM
bfSize	4字节	整个BMP文件的大小	0x000C0036(786486)【与右键查看图片属性里面的大小值一样】
bfReserved1/2	4字节	保留字，没用	0
bfOffBits	4字节	偏移数，即 位图文件头+位图信息头+调色板的大小	0x36(54)

基本图像格式

- BMP文件的基本格式

  - 位图信息头

共40个字节，包括信息头的大小、图像的宽、高、每个像素的位数、位图使用颜色数等信息。



基本图像格式

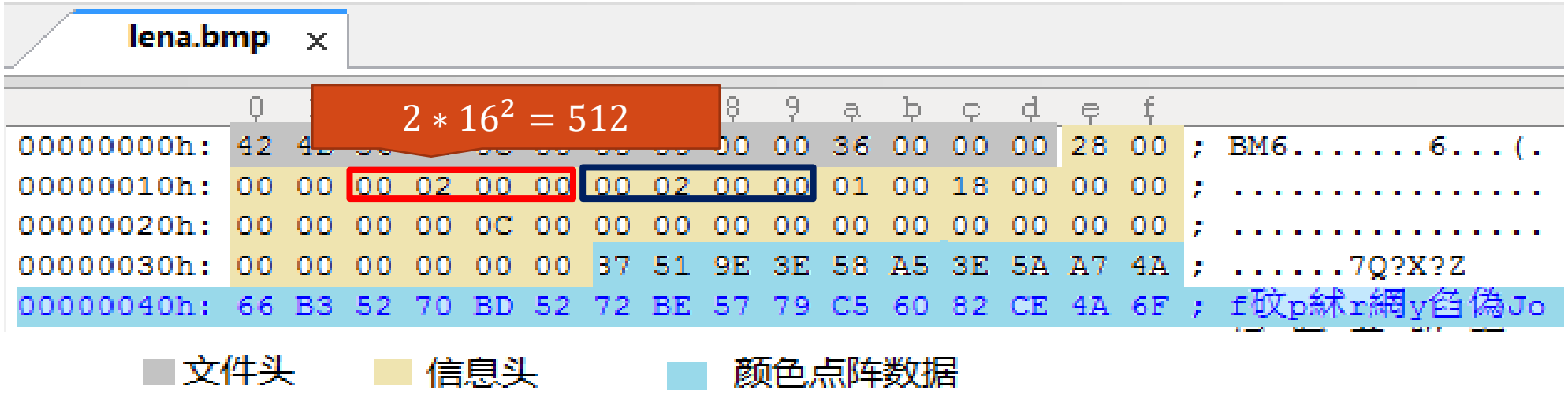
- BMP文件的基本格式

名称	占用空间	内容	实际数据
biSize	4字节	位图信息头的大小，为40	0x28(40)
biWidth	4字节	位图的宽度，单位是像素	0x200(512)
biHeight	4字节	位图的高度，单位是像素	0x200(512)
biPlanes	2字节	固定值1	1
biBitCount	2字节	每个像素的位数 1-黑白图，4-16色，8-256色，24-真彩色	0x18(24)
biCompression	4字节	压缩方式，BI_RGB(0)为不压缩	0
biSizeImage	4字节	位图全部像素占用的字节数，BI_RGB时可设为0	0x0C
biXPelsPerMeter	4字节	水平分辨率(像素/米)	0
biYPelsPerMeter	4字节	垂直分辨率(像素/米)	0
biClrUsed	4字节	位图使用的颜色数 如果为0，则颜色数为2的biBitCount次方	0
biClrImportant	4字节	重要的颜色数，0代表所有颜色都重要	0

基本图像格式

- BMP文件的基本格式
  - 位图信息头

共40个字节，11个数值



作为真彩色位图，主要关心的是biWidth和biHeight这两个数值，两个数值表示图像的尺寸，biSize，biPlanes，biBitCount这几个数值是固定的（以像素为单位）。

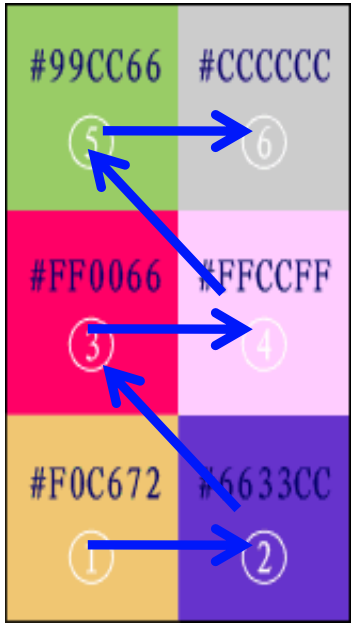
基本图像格式

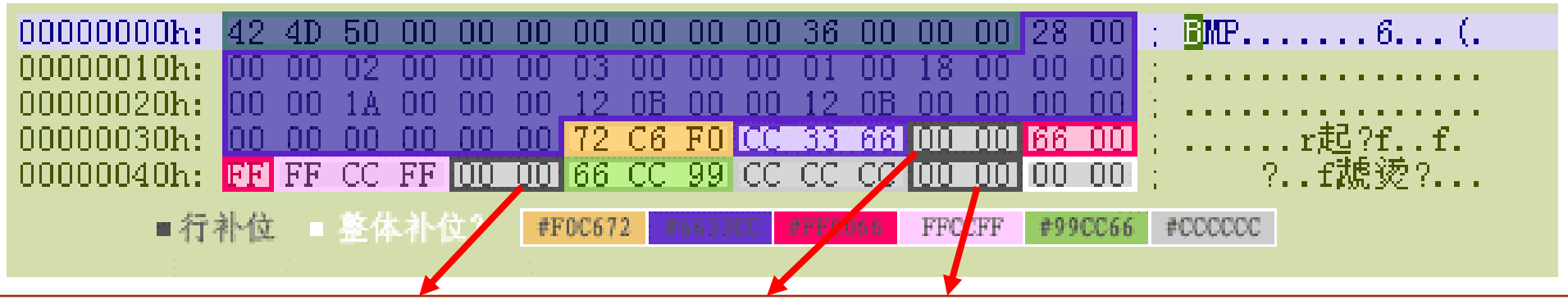
- BMP文件的基本格式
  - 颜色点阵数据

解读关键：倒着看&倒着念

位图全部的像素，是按照自下向上，自左向右的顺序排列的。

RGB数据也是倒着念的，原始数据是按B、G、R的顺序排列的。





行补位：因为32位的Windows操作系统处理4个字节(32位)的速度比较快，所以BMP的每一行颜色占用的字节数规定为4的整数倍。这里一行颜色有两个像素，共占用6字节，如果要补齐4*2=8字节，就要再加两个0字节。

BMP文件的基本格式

BITMAPFILEHEADER （位图文件头）		bfType(位图文件的类型，必须为BM) bfSize(位图文件的大小，以字节为单位) bfReserved1(必须为0) bfReserved2(必须为0) bfOffBits(位数据起始位置距离文件开始的字节数) *这部分长度固定为14个字节*
BITMAPINFO （位图信息）	BITMAPINFOHEADER （位图信息头）	biSize(本结构大小) biWidth(按象素计算) biHeight(按象素计算) biPlanes(必须为1) biBitCount (每个像素所需的位数,1,4,8,16,24,32) biCompression (位图是否压缩，0或BI_RGB表示不压缩) biSizeImage (只在压缩图中使用) biXPelsPerMeter (水平分辨率，一般可设为0) biYPelsPerMeter (垂直分辨率，一般可设为0) biClrUsed (指定实际用到的颜色个数，为0时，表示所有颜色都用到了) biClrImportant (重要颜色数，一般可设为0) *这部分长度固定为40个字节*
	颜色表	一个数组表示，共有biClrUsed个元素（若该值为0，则有2的biBitCount次幂个元素） 单色DIB(2位)有2个表项，16色DIB(4位)有16个表项，256色DIB(8位)有256个表项或更少，真彩DIB(24位)无颜色表 每个表项为32位（4个字节），是一个RGBQUAD类型的结构，用R、G、B三原色定义一种颜色，整个颜色表实际是一个RGBQUAD类型的数组
DIB位数据		象素按每行每列的顺序排列 从最下面的行开始排列 每行扩展到4字节边界 按扫描行内从左到右,扫描行之间从下到上的顺序依次记录每个象素

Why Lena

在数字图像处理中，Lena（Lenna）是一张被广泛使用的标准图片。

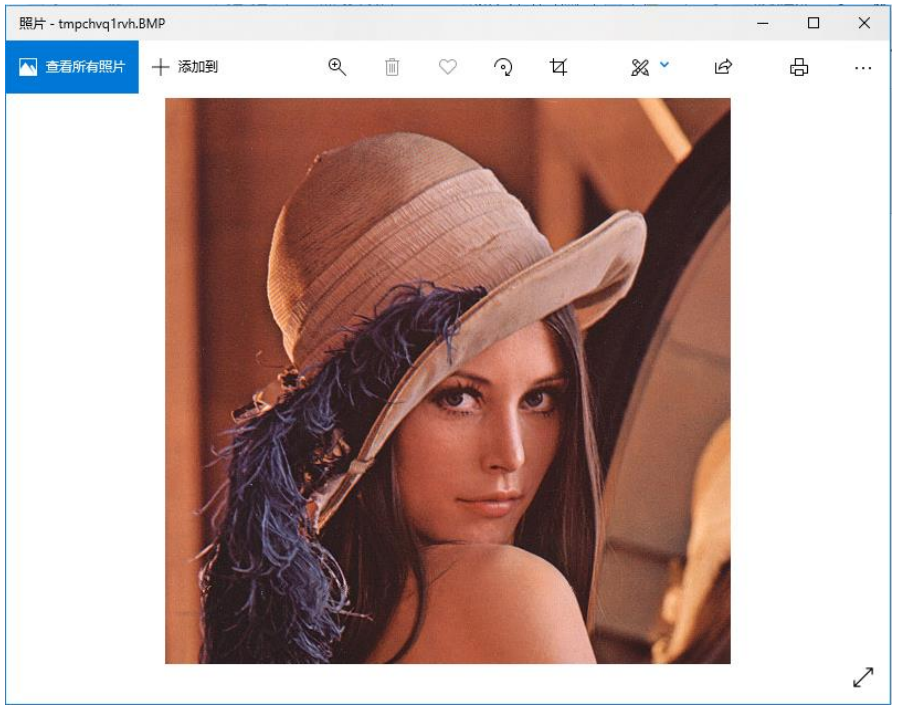
1973年6月，美国南加州大学的一名教授想找一幅图像来做图像压缩的测试，他已厌倦了手头繁杂的照片，想找张能让人眼前一亮的照片。恰好这时，一人拿着《花花公子》走了进来，Lena的照片确实够让教授眼前一亮了。教授便将《花花公子》的这期插页图用扫描了下来截取其中的一部分作为了他研究使用的样例图像。从此，这幅512*512的经典图像就诞生了。

该图适度的混合了细节、平滑区域、阴影和纹理，从而能很好的测试各种图像处理算法。

基本图像格式

- 打开JPG文件

```
#读取图像，请注意相对路径与绝对路径的关系
img = Image.open("../img/lena.jpg")
```



Python-image图像基本操作

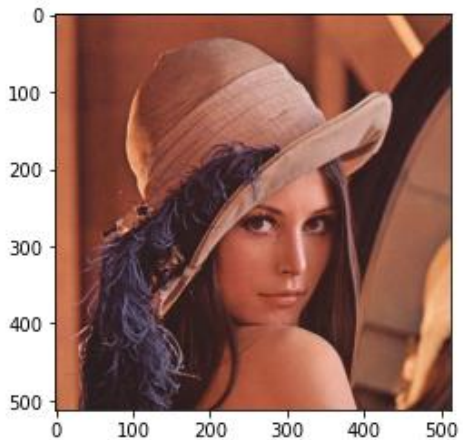
- 图像的读取、另存以及信息的查看

图像的读取与显示,与存储(Pillow)

```
[174]: #读取图像，请注意相对路径与绝对路径的关系
       img = Image.open("../img/lena.jpg")

[175]: #图像相关信息
       print(img.format,img.size,img.mode)

       JPEG (512, 512) RGB
```

图像信息

```
[176]: #用系统默认图片查看器查看图像
       img.show()

[177]: #inline方式查看图像-方案1
       plt.imshow(img)

[177]: <matplotlib.image.AxesImage at 0x1f360f38160>
```



Python-image图像基本操作

- 创建新图片

PIL.Image.new(*mode, size, color=0*) [source]

Creates a new image with the given mode and size.

Parameters

- **mode** – The mode to use for the new image. See: Modes.
- **size** – A 2-tuple, containing (width, height) in pixels.
- **color** – What color to use for the image. Default is black. If given, this should be a single integer or floating point value for single-band modes, and a tuple for multi-band modes (one value per band). When creating RGB images, you can also use color strings as supported by the ImageColor module. If the color is None, the image is not initialised.

Returns

An `Image` object.

The `mode` of an image defines the type and depth of a pixel in the image. The current release supports the following standard modes:

- `1` (1-bit pixels, black and white, stored with one pixel per byte)
- `L` (8-bit pixels, black and white)
- `P` (8-bit pixels, mapped to any other mode using a color palette)
- `RGB` (3x8-bit pixels, true color)
- `RGBA` (4x8-bit pixels, true color with transparency mask)
- `CMYK` (4x8-bit pixels, color separation)
- `YCbCr` (3x8-bit pixels, color video format)
  - Note that this refers to the JPEG, and not the ITU-R BT.2020, standard

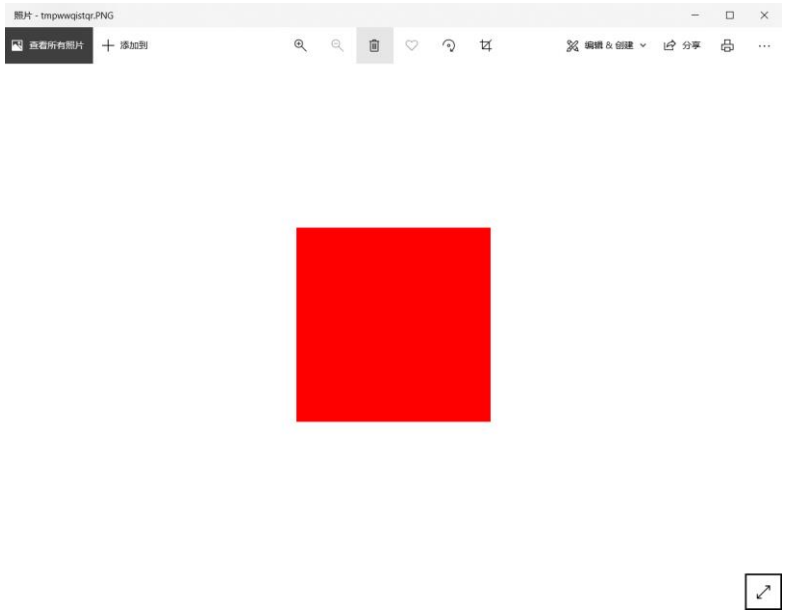
- `LAB` (3x8-bit pixels, the L*a*b color space)
- `HSV` (3x8-bit pixels, Hue, Saturation, Value color space)
- `I` (32-bit signed integer pixels)
- `F` (32-bit floating point pixels)

关于mode的信息，可以查看
https://www.jianshu.com/p/171ce1d0656e

Python-image图像基本操作

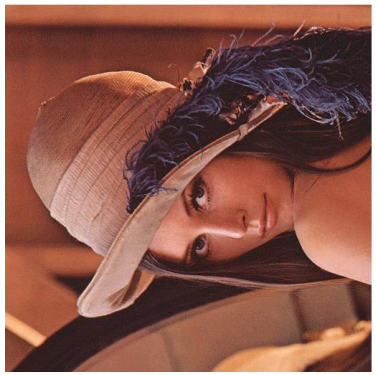
- 创建新图片

```
[181]: #这两行代码等效
       img_new = Image.new("RGB",(256,256),(255,0,0))
       img_new = Image.new("RGB",(256,256),"red")
```

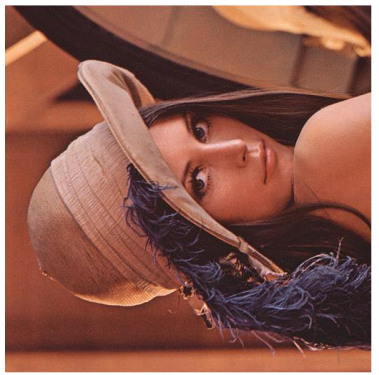


Python-image图像基本操作

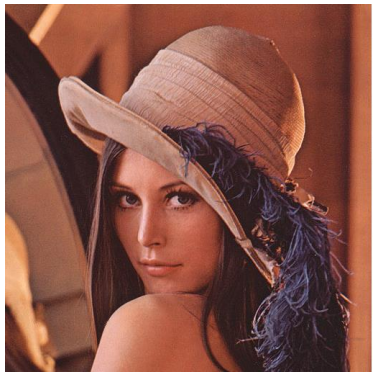
- 几何变换：镜像、旋转、缩放



转置



旋转



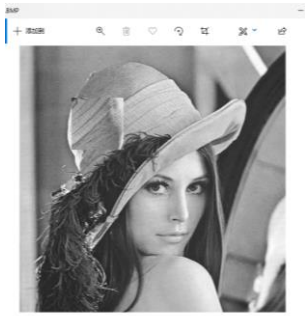
镜像

Python-image图像基本操作

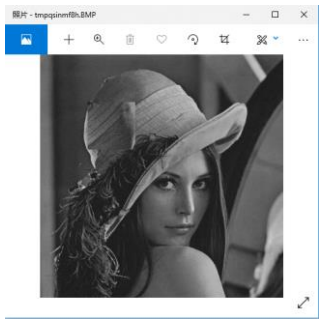
- 通道操作

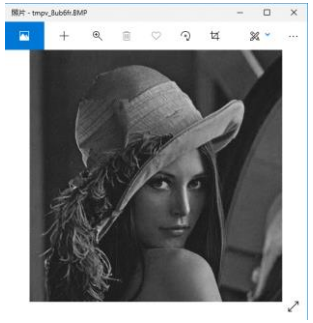
每一个RGB都是由三个通道的灰度图叠加的，PIL提供了将这三个通道分离的方法。

```
r, g, b = img_PIL.split()
```



	Column 0			Column 1			Column ...				Column n
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m





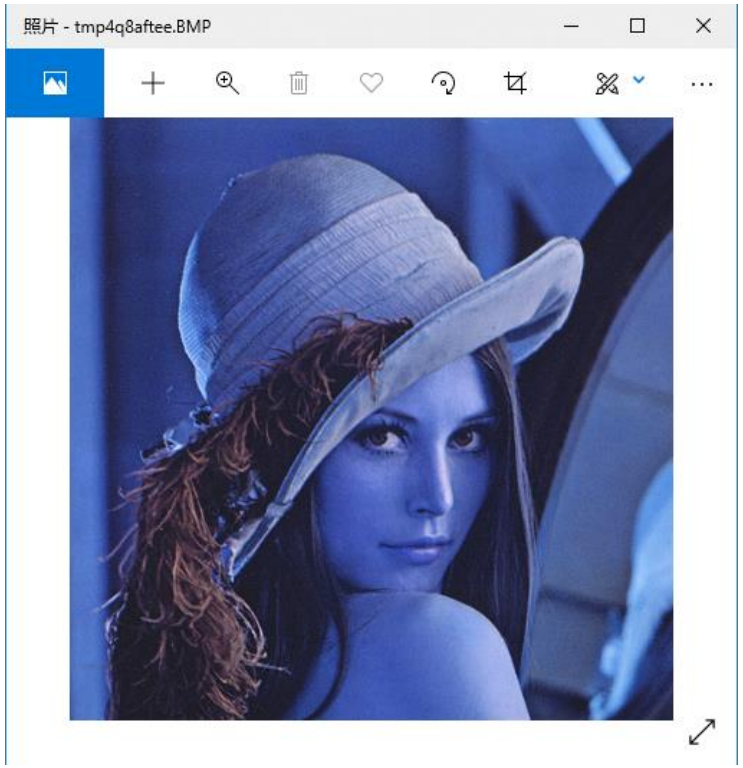
Python-image图像基本操作

- R、G、B通道融合

图像基本通道操作

通道融合（假彩色）

```
[120]: r,g,b = img.split()

[121]: img_bgr = Image.merge("RGB",(b,g,r))
       img_bgr.show()
```



Python-image图像基本操作

- 灰度变换

对于一张灰度图像，其每个像素点都用一个0-255之间的值表示，0表示黑色，越接近0越黑；255表示白色，越接近255越白。

灰度变换就是通过一个特定的函数，使灰度值从一个值转换成另外一个值。

Python-image图像基本操作

- 灰度变换之反相变换

反相变换

[183]:
```
img_gray = img.convert("L")
img_gray_np = np.array(img_gray) #读入数组
```

[184]:
```
img_neg = 255-img_gray_np
```

[185]:
```
img_neg = Image.fromarray(img_neg)
img_neg.show()
```

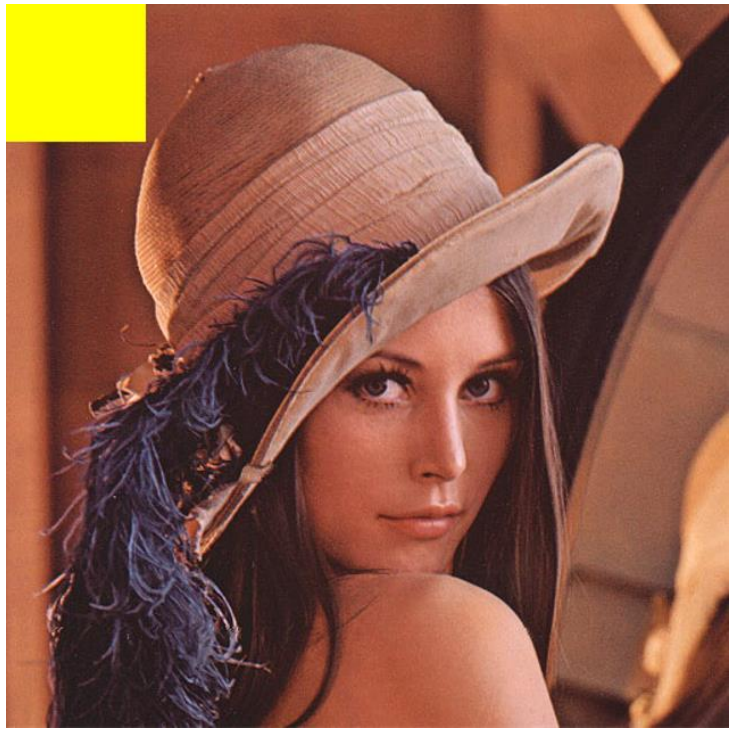
Python-image图像基本操作

- 对像素的操作

对图像像素直接操作

[188]:
```
img_np2 = np.zeros((h,w,c)).astype("uint8")
for ih in range(h):
    for iw in range(w):
        if ih< 100 and iw <100:
            img_np2[ih,iw] = [255,255,0]
        else:
            img_np2[ih,iw] = img_np[ih,iw]
```

[189]:
```
out = Image.fromarray(img_np2)
out.show()
```



作业1-Python-image图像基本操作

编程实现以下内容:

- 准备两张不同格式的彩色图片，打开并展示;(感兴趣的同学可以尝试根据图像格式，自己写程序解码读取图像文件)
- 将其中一幅红蓝波段对调,得到假彩色合成结果显示并保存;
- 将另一幅变成**64灰阶**的灰度图显示并保存;
- 将最终得到的结果保存成文件。
- 完成实验报告（**实验目的、意义，实验内容以及实验结果，实验感想**）

注：作业提交要求见课程介绍，内容为：代码文件、实验报告、实验结果图像

- 学习numpy的使用