Python面向对象语法精讲

嵩 天

实例2：图像的四则运算

嵩 天

Python面向对象语法精讲

"图像的四则运算"需求分析

需求分析

图像之间进行四则运算：加减乘除

- 加减法：两个图像相加减
- 乘除法：一个图像与一个数字之间的乘除法

需求分析

图像之间进行四则运算：加减乘除

- 通过面向对象程序设计方法，简化程序表达
- 让图像处理变得更加有趣、更加人性化
- 图像变换的抽象层次更加高级 （抽象是追求的目标）

需求分析

程序之前

基础图片：earth.jpg




辅助图片：gray.png、dark.png、blue.png、white.png

需求分析

numpy库和PIL库

- PIL库是一个能够简单读写图像文件的库
- numpy库是一个矩阵表示的数学库
- 图像可以理解为一个三维数据，像素是(r,g,b)，长宽是2个维度

Python面向对象语法精讲

"图像的四则运算"实例解析

Python123

代码纵览


```python
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image

    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码纵览

```python
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image

    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

重载4个运算

代码详解

```python
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image
```

初始化对象

将图片转换为np.array类型数组

PIL库中Image.open()用于打开文件

代码详解

```
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image
```

图像加法操作

建立一个新的图像对象

运算完成后返回新图像引用

符合加法操作意图，不修改加数

代码详解

```python
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image
```

图像加法操作

直接使用np.array对象的加法

利用np.mod()函数对元素取模

RGB颜色范围是0-255

任何异常则第一个加数的图像作为输出

代码详解

```
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image
```

图像减法操作

建立一个新的图像对象

运算完成后返回新图像引用

符合减法操作意图，不修改减数

代码详解


```python
import numpy as np
from PIL import Image

class ImageObject:
    def __init__(self, path = ""):
        self.path = path
        try:
            self.data = np.array(Image.open(path))
        except:
            self.data = None

    def __add__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data + other.data, 255)
        except:
            image.data = self.data
        return image

    def __sub__(self, other):
        image = ImageObject()
        try:
            image.data = np.mod(self.data - other.data, 255)
        except:
            image.data = self.data
        return image
```

图像减法操作

直接使用np.array对象的减法

利用np.mod()函数对元素取模

RGB颜色范围是0-255

任何异常则第一个减数的图像作为输出

代码详解

图像乘法操作

建立一个新的图像对象

运算完成后返回新图像引用

符合乘法操作意图，不修改乘数

```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码详解

图像乘法操作

直接使用np.array对象的乘法

第二个乘数是一个整数数字

利用np.mod()函数对元素取模

RGB颜色范围是0-255

任何异常则第一个乘数的图像作为输出

```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码详解

图像除法操作

建立一个新的图像对象

运算完成后返回新图像引用

符合除法操作意图，不修改除数

```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码详解

图像除法操作

直接使用np.array对象的除法

第二个除数是一个整数数字

利用np.mod()函数对元素取模

RGB颜色范围是0-255

任何异常则第一个除数的图像作为输出

```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码详解

保存文件方法

将np.array变成图像输出

```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

代码详解

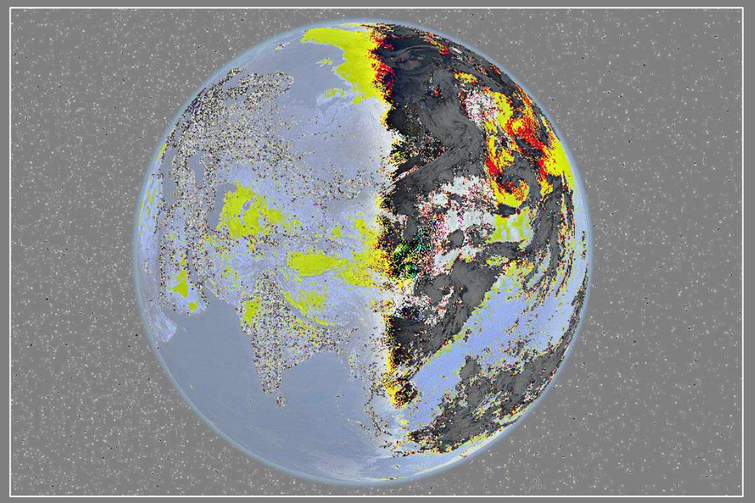
读入2个图像，进行四则运算

输出运行结果文件

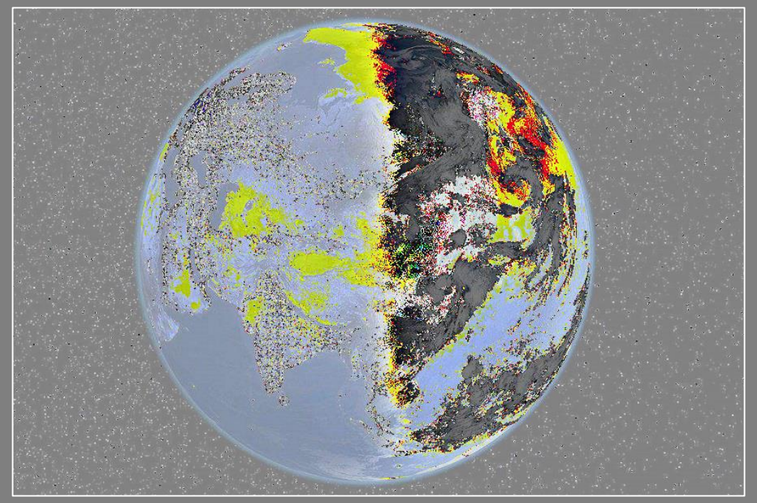
```
    def __mul__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data * factor, 255)
        except:
            image.data = self.data
        return image

    def __truediv__(self, factor):
        image = ImageObject()
        try:
            image.data = np.mod(self.data // factor, 255)
        except:
            image.data = self.data
        return image

    def saveImage(self, path):
        try:
            im = Image.fromarray(self.data)
            im.save(path)
            return True
        except:
            return False

a = ImageObject("earth.jpg")
b = ImageObject("gray.png")
(a + b).saveImage("result_add.png")
(a - b).saveImage("result_sub.png")
(a * 2).saveImage("result_mul.png")
(a / 2).saveImage("result_div.png")
```

程序结果

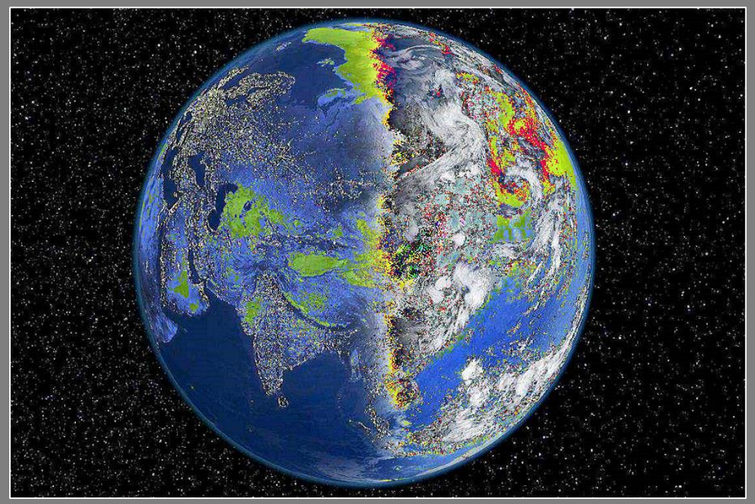
加法



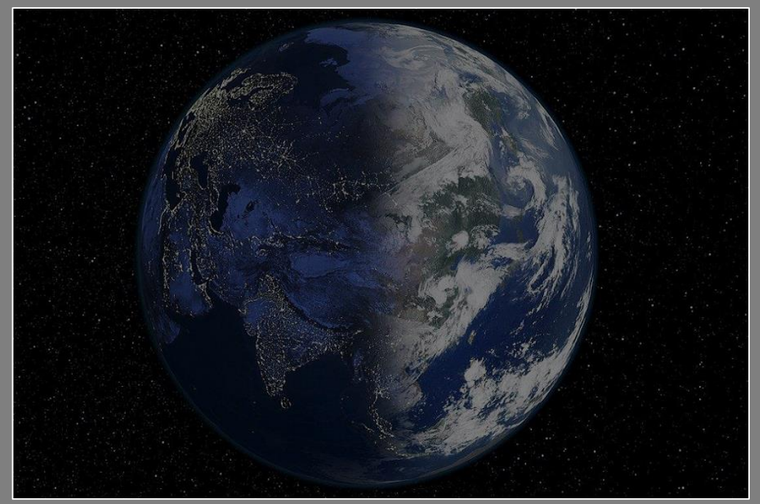
减法



乘法



除法




Python 123
Thank you