Things to discuss:

Read images with Pillow

Perform basic image manipulation operations

Use Pillow for image processing

Use NumPy with Pillow for further processing

In [1]:

```
pip install Pillow
```

```
Requirement already satisfied: Pillow in c:\users\dell\anaconda3\lib\site-
packages (6.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In this notebook, you will find

Basic Image Operations With the Python Pillow Library

-The Image Module and Image Class in Pillow

-Basic Image Manipulation

-Bands and Modes of an Image in the Python Pillow Library

In [82]:

```
from PIL import Image
filename = "flowers.jpg"
with Image.open(filename) as img:
    img.load()

type(img)
```

Out[82]:

```
PIL.JpegImagePlugin.JpegImageFile
```

In [83]:

```
isinstance(img, Image.Image)
```

Out[83]:

```
True
```

In [84]:

```
img
```

Out[84]:

In [8]:

```
# to see image specs
```

In [85]:

```
img.format
```

Out[85]:

```
'JPEG'
```

In [86]:

```
img.size
```

Out[86]:

```
(850, 478)
```

In [7]:

```
img.mode
```

Out[7]:

```
'RGB'
```

In [10]:

```
# The format of an image shows what type of image you're dealing with. In this case, the
# The size shows the width and height of the image in pixels.
# The mode of this image is 'RGB'.
```

In [87]:

```
cropped_img = img.crop((300, 150, 700, 1000))
cropped_img.size
```

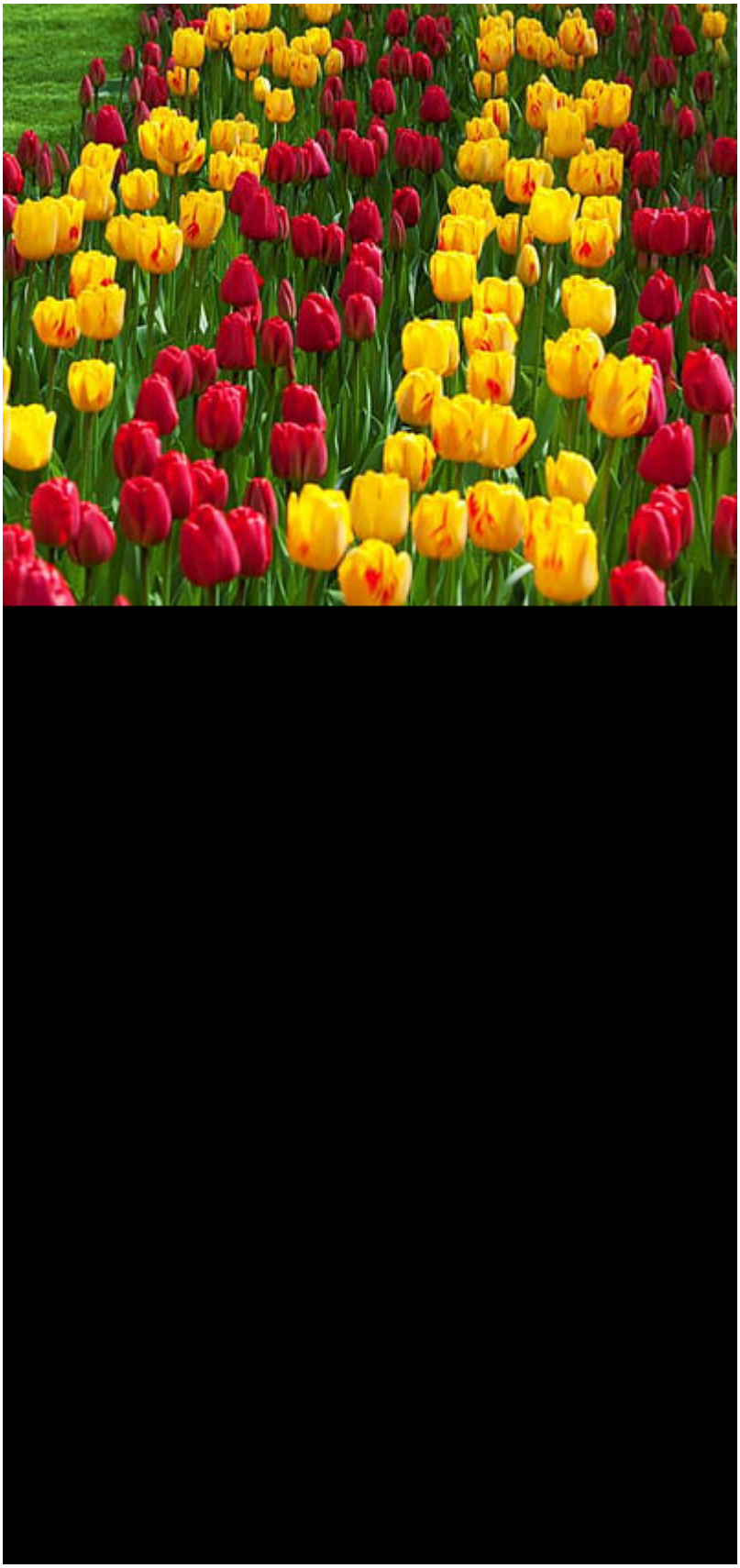
Out[87]:

```
(400, 850)
```

In [88]:

```
cropped_img
```

Out[88]:



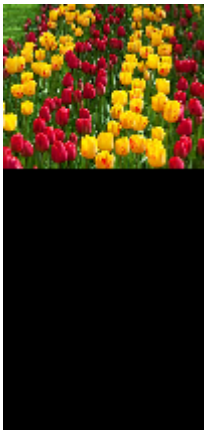
In [89]:

```
low_res_img = cropped_img.resize((cropped_img.width // 4, cropped_img.height // 4))
```

In [90]:

```
low_res_img
```

Out[90]:



In [91]:

```
cropped_img.save("cropped_image.jpg")
```

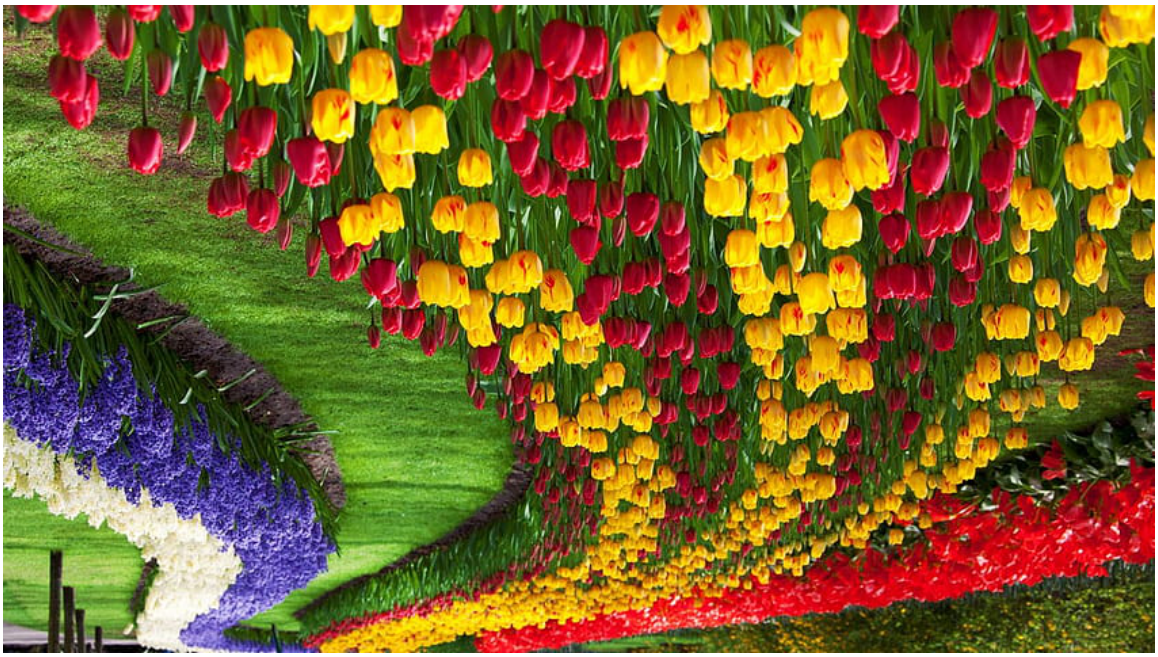
In [92]:

```
low_res_img.save("low_resolution_cropped_image.png")
```

In [93]:

```
converted_img = img.transpose(Image.FLIP_TOP_BOTTOM)
converted_img
```

Out[93]:



In [94]:

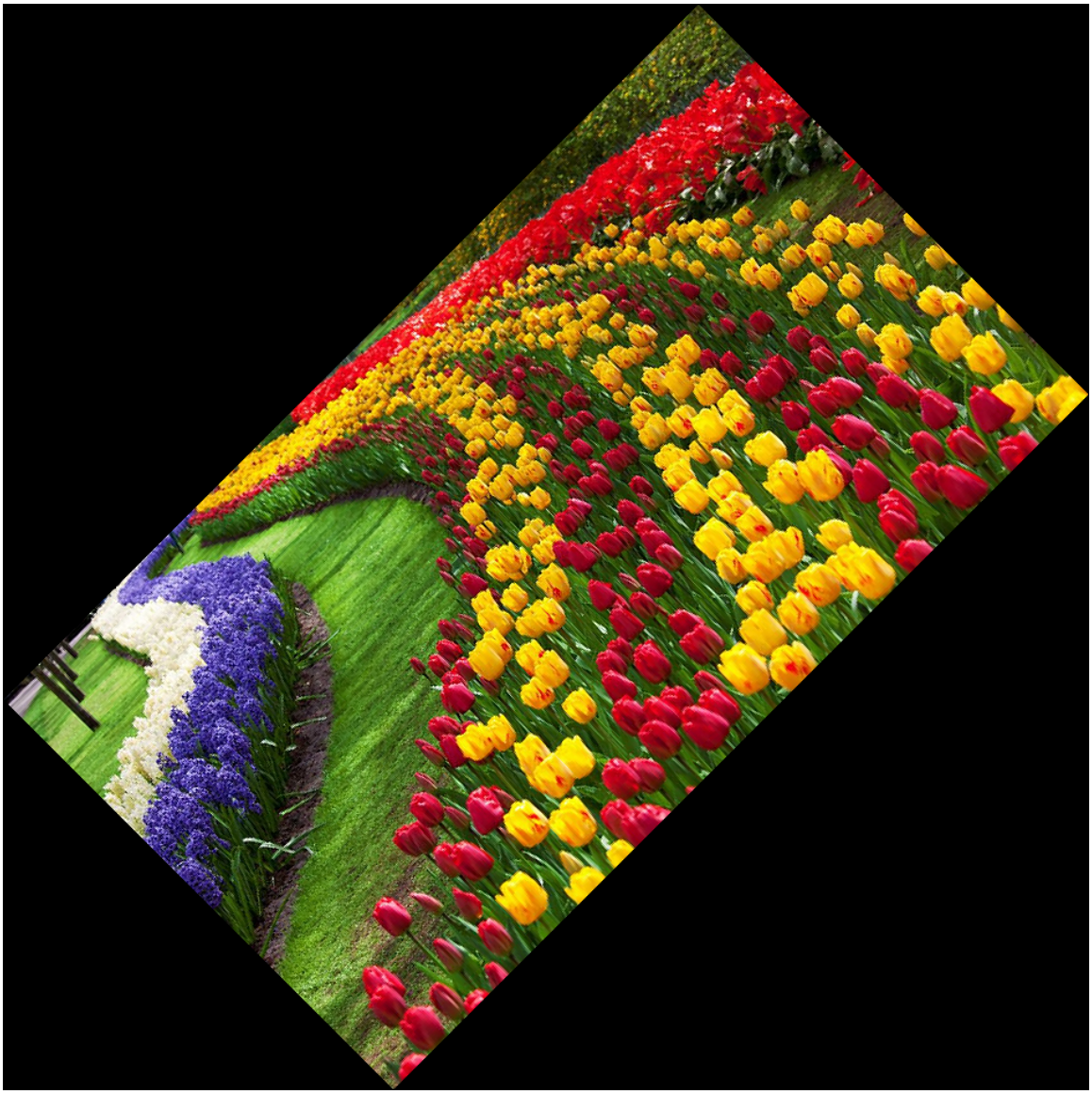
```
rotated_img = img.rotate(45) # This method call rotates the image by 45 degrees counterc
rotated_img
```

Out[94]:

In [95]:

```
rotated_img = img.rotate(45, expand=True) # This method returns a larger image that fully
rotated_img
```

Out[95]:



In [21]:

```
# Bands and Modes of an Image in the Python Pillow Library
```

In [96]:

```
filename = "flowers.jpg"
with Image.open(filename) as img:
    img.load()
```

In [97]:

```
cmyk_img = img.convert("CMYK")
gray_img = img.convert("L")  # Grayscale
```

In [101]:

```
cmyk_img.show()
```

In [99]:

```
gray_img
```

Out[99]:

In [102]:

```
img.getbands()
```

Out[102]:

```
('R', 'G', 'B')
```

In [103]:

```
cmyk_img.getbands()
```

Out[103]:

```
('C', 'M', 'Y', 'K')
```

In [104]:

```
gray_img.getbands()
```

Out[104]:

```
('L',)
```

In [105]:

```
red, green, blue = img.split()
red
```

Out[105]:

In [106]:

```
red.mode
```

Out[106]:

```
'L'
```

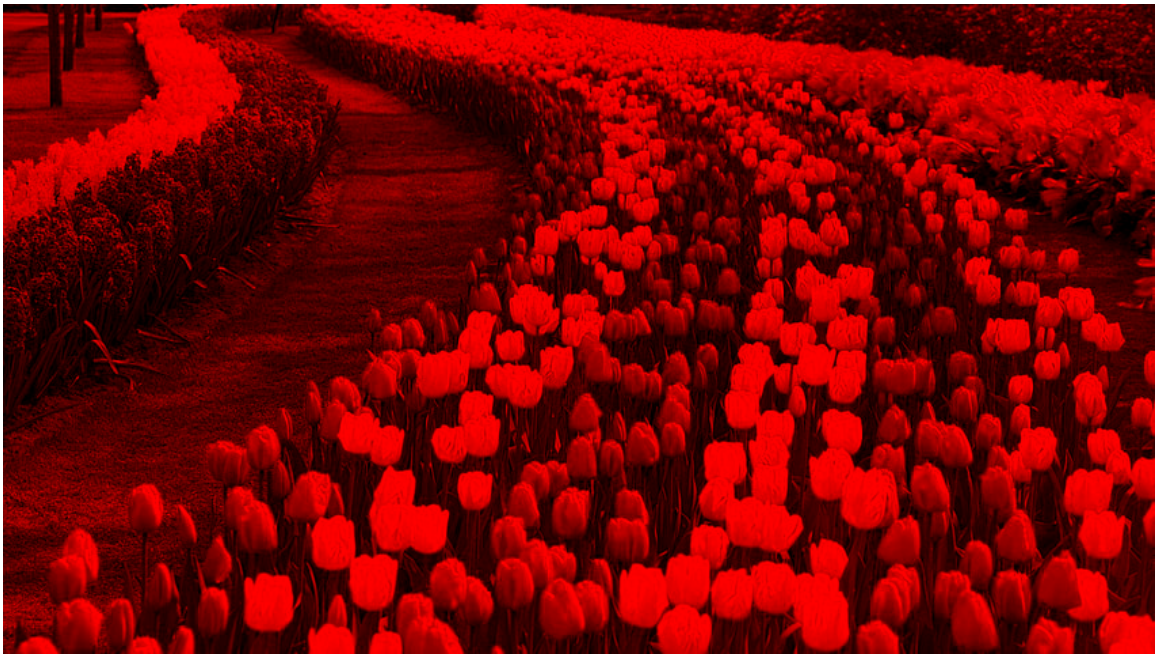
In [107]:

```
zeroed_band = red.point(lambda _: 0)

red_merge = Image.merge("RGB", (red, zeroed_band, zeroed_band))

green_merge = Image.merge("RGB", (zeroed_band, green, zeroed_band))

blue_merge = Image.merge("RGB", (zeroed_band, zeroed_band, blue))
```

In [108]:

```
red_merge
```

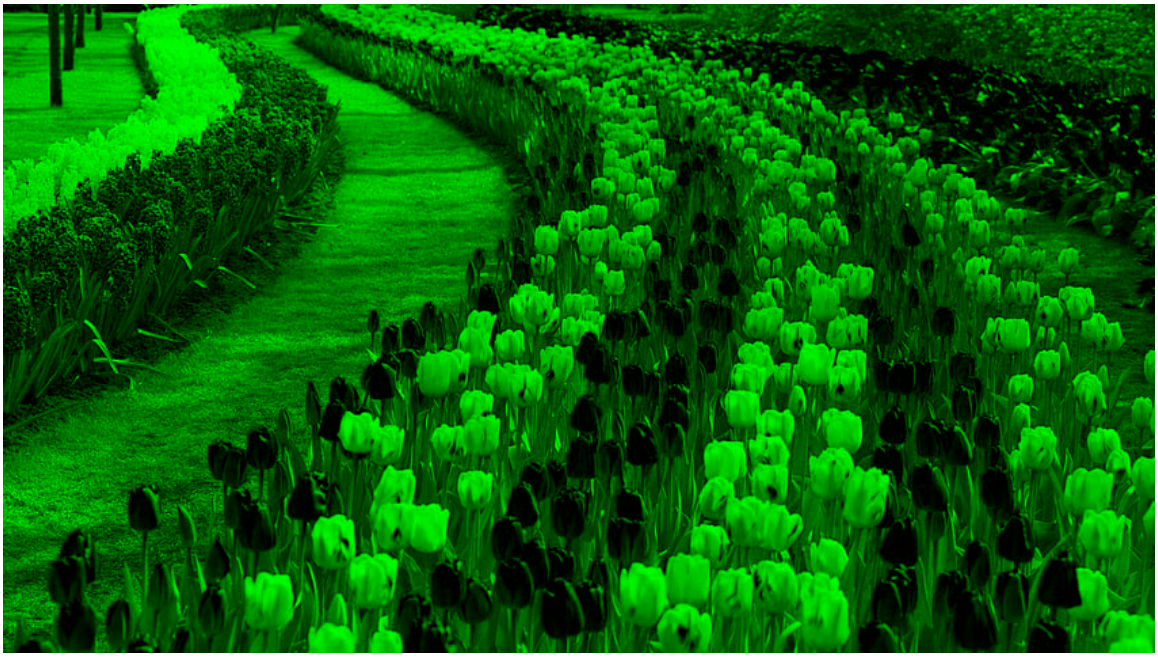
Out[108]:



In [109]:

```
green_merge
```

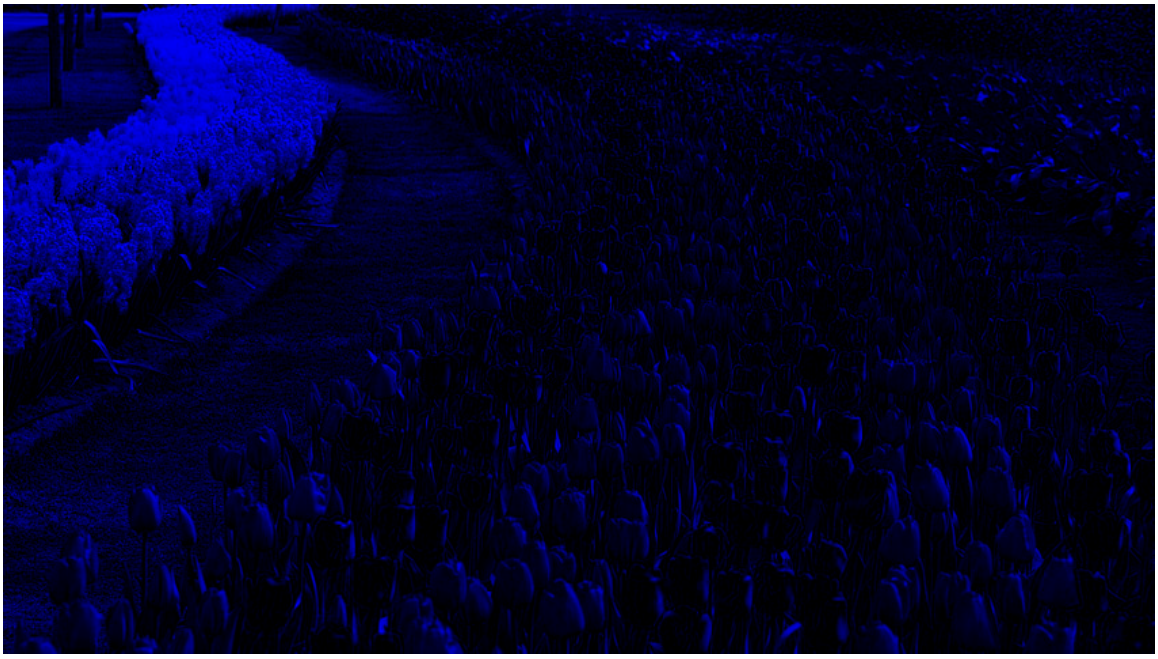
Out[109]:



In [110]:

```
blue_merge
```

Out[110]:



In [36]:

```
# Image Processing Using Pillow in Python
```

In [37]:

```
# Image Filters Using Convolution Kernels
```

In [112]:

```
from PIL import Image, ImageFilter
```

In [113]:

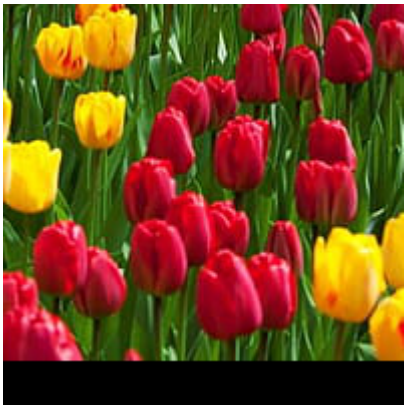
```
blur_img = img.filter(ImageFilter.BLUR)
blur_img
```

Out[113]:

In [114]:

```
img.crop((300, 300, 500, 500))
```

Out[114]:



In [115]:

```
blur_img.crop((300, 300, 500, 500))
```

Out[115]:

In [116]:

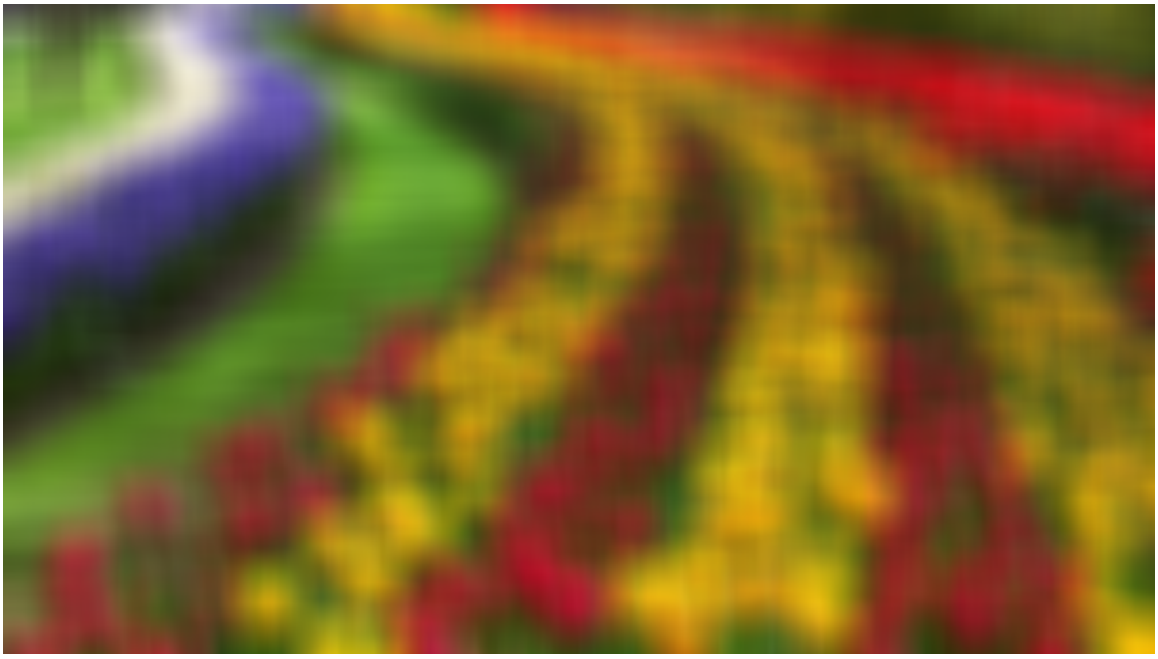
```
img.filter(ImageFilter.BoxBlur(5))
```

Out[116]:

In [117]:

```
img.filter(ImageFilter.BoxBlur(20))
```

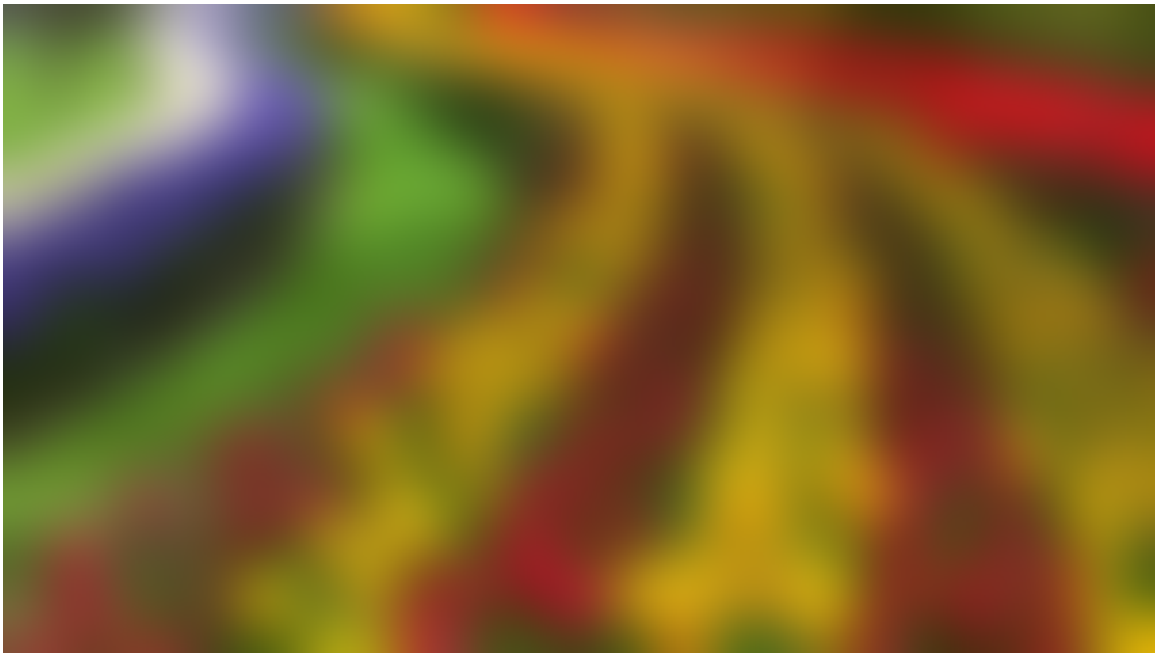
Out[117]:



In [118]:

```
img.filter(ImageFilter.GaussianBlur(20))
```

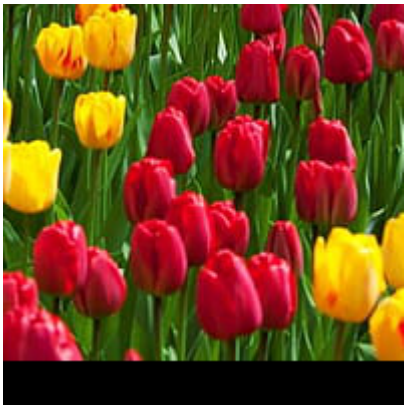
Out[118]:



In [119]:

```
sharp_img = img.filter(ImageFilter.SHARPEN)
img.crop((300, 300, 500, 500))
```

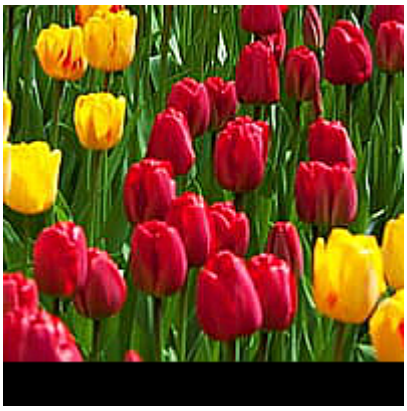
Out[119]:



In [120]:

```
sharp_img.crop((300, 300, 500, 500))
```

Out[120]:



In [121]:

```
smooth_img = img.filter(ImageFilter.SMOOTH)
img.crop((300, 300, 500, 500))
```

Out[121]:

In [122]:

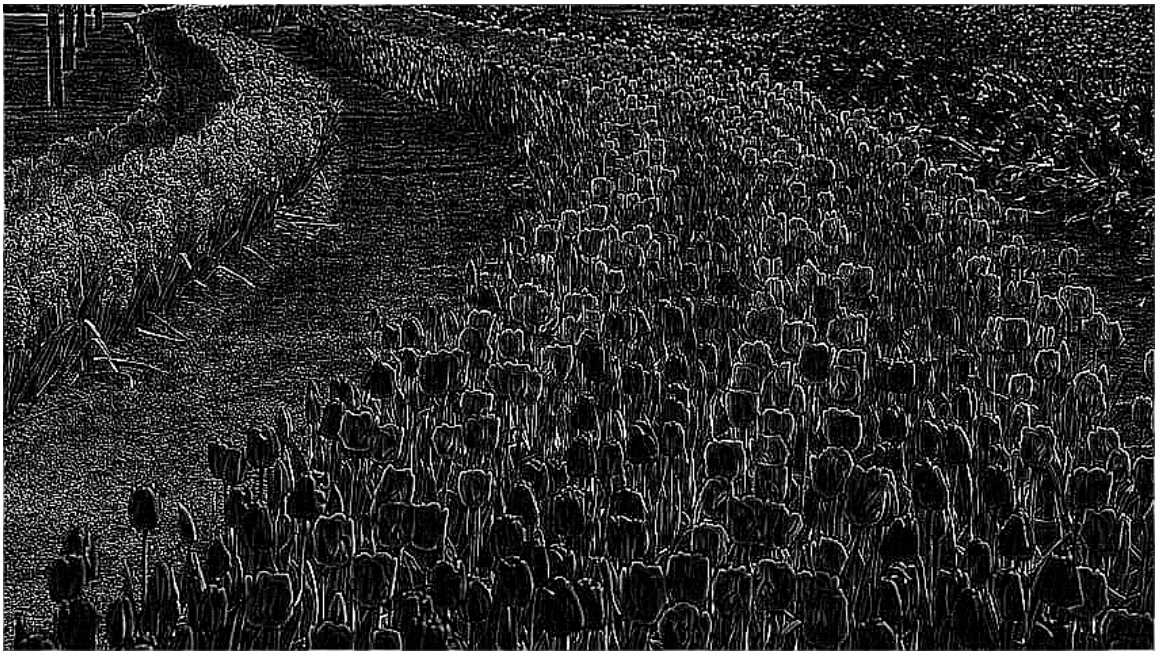
```
smooth_img.crop((300, 300, 500, 500))
```

Out[122]:

In [123]:

```
#showing the edges from the original image
img_gray = img.convert("L")
edges = img_gray.filter(ImageFilter.FIND_EDGES)
edges
```

Out[123]:



In [124]:

```
img_gray_smooth = img_gray.filter(ImageFilter.SMOOTH)
edges_smooth = img_gray_smooth.filter(ImageFilter.FIND_EDGES)
edges_smooth
```

Out[124]:

In [125]:

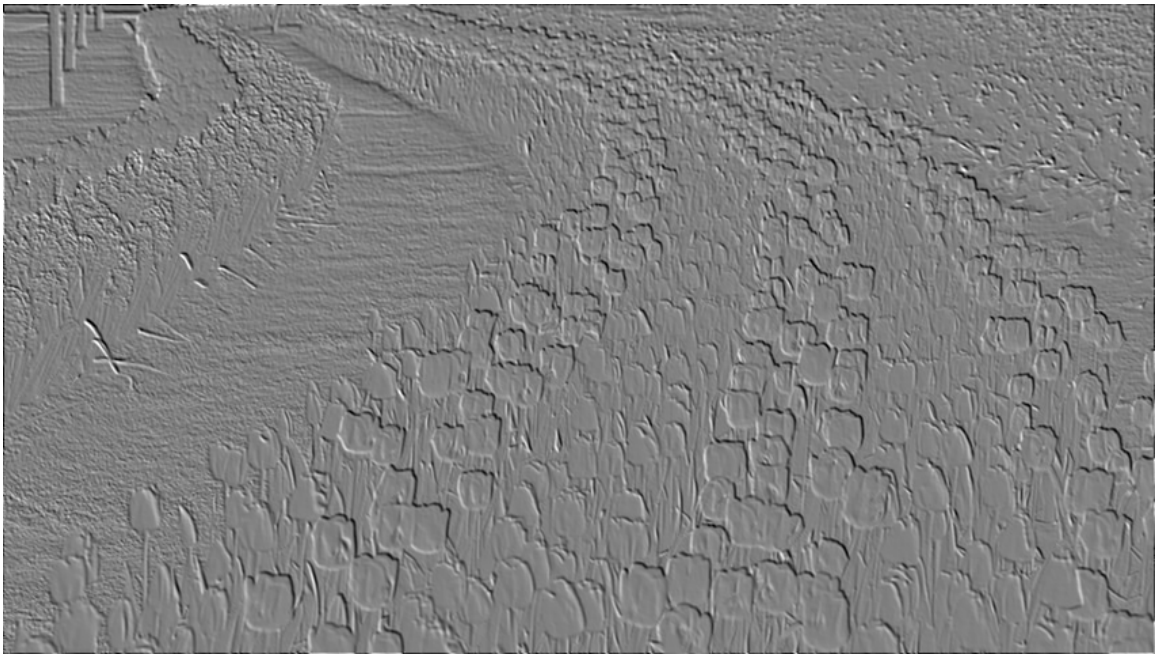
```
edge_enhance = img_gray_smooth.filter(ImageFilter.EDGE_ENHANCE)
edge_enhance
```

Out[125]:

In [126]:

```
emboss = img_gray_smooth.filter(ImageFilter.EMBOSS)
emboss
```

Out[126]:



In [ ]:

```

```