PIL Intro - Basic PIL Processing

Basic PIL operations for image processing.

In [11]:
```
import PIL
from PIL import Image
from IPython.display import display
ifile = "misc/4.2.07.tiff"
image = Image.open(ifile)
```

Make a copy of the image to use for modification. Note the conversion to RGB when the image is being opened.

In [12]:
```
image.save("peppers-copy.png")
image = Image.open("peppers-copy.png").convert('RGB')
```

Now show the image.

In [13]:
```
display(image)
```

This image will be used to make one larger image named `gradimg` which will show the gradient of various effects from PIL's `ImageEnhance` module. To view the `gradimg` easily the copy will be scaled to half size. Each image with increased effect will be stored in the list `imglist` and modifed with an enhancer object. Let's start with brightness.

In [14]:
```
from PIL import ImageEnhance

# rescale image
image.resize((256,256))

# create image modifing object
enhancer = ImageEnhance.Brightness(image)

# make image gradients
imgbuf = []
for i in range(3, 8):
    imgbuf.append(enhancer.enhance(i/5))
```

Now create a new image that stitches the 5 above into one. The `Image.new()` method requires a mode (RGB in this case) and size, which will be retrieved from the first image of `gradimg`. The images will be offset (i.e. `xpos`) by the width after each paste.

In [15]:
```
# reference image
imgref = imgbuf[0]

# create new image with 5 images from imgbuf
gradimg = Image.new(imgref.mode, (imgref.width*5, imgref.height))
xpos = 0
for img in imgbuf:
    gradimg.paste(img, (xpos, 0))
    xpos += imgref.width
```

Now, print the result!

In [16]:
```
display(gradimg)
```

This can be done easily for other enhance methods, such as contrast.

In [17]:
```
# create image modifing object
enhancer = ImageEnhance.Contrast(image)

# make image gradients
imgbuf = []
for i in range(3, 8):
    imgbuf.append(enhancer.enhance(i/5))

# reference image
imgref = imgbuf[0]

# create new image with 5 images from imgbuf
gradimg = Image.new(imgref.mode, (imgref.width*5, imgref.height))
xpos = 0
for img in imgbuf:
    gradimg.paste(img, (xpos, 0))
    xpos += imgref.width

# display image
display(gradimg)
```

Or sharpness.

In [18]:
```
# create image modifing object
enhancer = ImageEnhance.Sharpness(image)

# make image gradients
imgbuf = []
for i in range(3, 8):
    imgbuf.append(enhancer.enhance(i/5))

# reference image
imgref = imgbuf[0]

# create new image with 5 images from imgbuf
gradimg = Image.new(imgref.mode, (imgref.width*5, imgref.height))
xpos = 0
for img in imgbuf:
    gradimg.paste(img, (xpos, 0))
    xpos += imgref.width

# display image
display(gradimg)
```

And finally, color. Here the enhance value was modified for the range $[0, 1]$.

In [19]:
```
# create image modifing object
enhancer = ImageEnhance.Color(image)

# make image gradients
imgbuf = []
for i in range(0, 6):
    imgbuf.append(enhancer.enhance(i/5))

# reference image
imgref = imgbuf[0]

# create new image with 5 images from imgbuf
gradimg = Image.new(imgref.mode, (imgref.width*5, imgref.height))
xpos = 0
for img in imgbuf:
    gradimg.paste(img, (xpos, 0))
    xpos += imgref.width

# display image
display(gradimg)
```

Another way to modify color is by manipulating each color layer. Here the image is adjusted within the RGB layers.

In [20]:
```
# split image into RGB
r,g,b = image.split()

# make image gradients
imgbuf = []
for i in range(1,10):

    factor = (0.1,0.5,0.9)

    if i < 4:
        mask = r.point(lambda j: j * factor[i-1])
        imagetemp = Image.merge(image.mode, (mask, g, b))
    elif i < 7 and i > 3:
        mask = g.point(lambda j: j * factor[i-4])
        imagetemp = Image.merge(image.mode, (r, mask, b))
    elif i > 6:
        mask = b.point(lambda j: j * factor[i-7])
        imagetemp = Image.merge(image.mode, (r, b, mask))

    imgbuf.append(imagetemp)

# create new image with 5 images from imgbuf
gradimg = Image.new(imgref.mode, (imgref.width*3, imgref.height*3))

xpos = 0
ypos = 0

for img in imgbuf:
    gradimg.paste(img, (xpos, ypos))
    xpos += imgref.width

    if xpos == 3*imgref.width:
        xpos = 0
        ypos += imgref.height

# display image
display(gradimg)
```