In [1]:
```
import PIL
```

In [2]:
```
PIL.__version__
```

Out[2]:
```
'5.4.1'
```

In [3]:
```
help(PIL)
```

```
Help on package PIL:

NAME
    PIL - Pillow (Fork of the Python Imaging Library)

DESCRIPTION
    Pillow is the friendly PIL fork by Alex Clark and Contributors.
        https://github.com/python-pillow/Pillow/ (https://github.com/python-p
illow/Pillow/)

    Pillow is forked from PIL 1.1.7.

    PIL is the Python Imaging Library by Fredrik Lundh and Contributors.
    Copyright (c) 1999 by Secret Labs AB.

    Use PIL.__version__ for this Pillow version.
    PIL.VERSION is the old PIL version and will be removed in the future.

    ;-)
```

In [4]:
```
dir(PIL)
```

Out[4]:
```
['PILLOW_VERSION',
 'VERSION',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 '_plugins']
```

In [5]:
```
from PIL import Image
help(Image)
```

```
Help on module PIL.Image in PIL:

NAME
    PIL.Image

DESCRIPTION
    # The Python Imaging Library.
    # $Id$
    #
    # the Image class wrapper
    #
    # partial release history:
    # 1995-09-09 fl   Created
    # 1996-03-11 fl   PIL release 0.0 (proof of concept)
    # 1996-04-30 fl   PIL release 0.1b1
    # 1999-07-28 fl   PIL release 1.0 final
    # 2000-06-07 fl   PIL release 1.1
    # 2000-10-20 fl   PIL release 1.1.1
    # 2001-05-07 fl   PIL release 1.1.2
```

In [6]:
```
help(Image.open)
```

```
Help on function open in module PIL.Image:

open(fp, mode='r')
    Opens and identifies the given image file.

    This is a lazy operation; this function identifies the file, but
    the file remains open and the actual image data is not read from
    the file until you try to process the data (or call the
    :py:meth:`~PIL.Image.Image.load` method).  See
    :py:func:`~PIL.Image.new`. See :ref:`file-handling`.

    :param fp: A filename (string), pathlib.Path object or a file object.
       The file object must implement :py:meth:`~file.read`,
       :py:meth:`~file.seek`, and :py:meth:`~file.tell` methods,
       and be opened in binary mode.
    :param mode: The mode.  If given, this argument must be "r".
    :returns: An :py:class:`~PIL.Image.Image` object.
    :exception IOError: If the file cannot be found, or the image cannot be
       opened and identified.
```

In [7]:
```
file='readonly/msi_recruitment.gif'
image=Image.open(file)
print(image)
```

```
<PIL.GifImagePlugin.GifImageFile image mode=P size=800x450 at 0x7FC6C42822B0>
```

In [8]:
```
import inspect
print('The type of the image is'+str(type(image)))
inspect.getmro(type(image))
```

```
The type of the image is<class 'PIL.GifImagePlugin.GifImageFile'>
```

Out[8]:
```
(PIL.GifImagePlugin.GifImageFile,
 PIL.ImageFile.ImageFile,
 PIL.Image.Image,
 object)
```

In [9]:
```
image.show()
```

In [10]:
```
from IPython.display import display
display(image)
```

In [11]:
```
help(image.copy)
```

```
Help on method copy in module PIL.Image:

copy() method of PIL.GifImagePlugin.GifImageFile instance
    Copies this image. Use this method if you wish to paste things
    into an image, but still retain the original.

    :rtype: :py:class:`~PIL.Image.Image`
    :returns: An :py:class:`~PIL.Image.Image` object.
```

In [12]:
```
help(image.save)
```

```
Help on method save in module PIL.Image:

save(fp, format=None, **params) method of PIL.GifImagePlugin.GifImageFile insta
nce
    Saves this image under the given filename.  If no format is
    specified, the format to use is determined from the filename
    extension, if possible.

    Keyword options can be used to provide additional instructions
    to the writer. If a writer doesn't recognise an option, it is
    silently ignored. The available options are described in the
    :doc:`image format documentation
    <../handbook/image-file-formats>` for each writer.

    You can use a file object instead of a filename. In this case,
    you must always specify the format. The file object must
    implement the ``seek``, ``tell``, and ``write``
    methods, and be opened in binary mode.

    :param fp: A filename (string), pathlib.Path object or file object.
    :param format: Optional format override.  If omitted, the
       format to use is determined from the filename extension.
       If a file object was used instead of a filename, this
       parameter should always be used.
    :param params: Extra parameters to the image writer.
    :returns: None
    :exception ValueError: If the output format could not be determined
       from the file name.  Use the format option to solve this.
    :exception IOError: If the file could not be written.  The file
       may have been created, and may contain partial data.
```

In [13]:
```
image.save('msi_recruitment.png')
image=Image.open('msi_recruitment.png')
import inspect
inspect.getmro(type(image))
```

Out[13]:
```
(PIL.PngImagePlugin.PngImageFile,
 PIL.ImageFile.ImageFile,
 PIL.Image.Image,
 object)
```

In [14]:
```
#filter functions to add effects
from PIL import ImageFilter
help(ImageFilter)
```

```
Help on module PIL.ImageFilter in PIL:

NAME
    PIL.ImageFilter

DESCRIPTION
    # The Python Imaging Library.
    # $Id$
    #
    # standard filters
    #
    # History:
    # 1995-11-27 fl   Created
    # 2002-06-08 fl   Added rank and mode filters
    # 2003-09-15 fl   Fixed rank calculation in rank filter; added expand cal
l
    #
    # Copyright (c) 1997-2003 by Secret Labs AB.
    # Copyright (c) 1995-2002 by Fredrik Lundh.
    "
```

In [15]:
```
#convert image to rgb mode
image=image.convert('RGB')
blurred_image=image.filter(PIL.ImageFilter.BLUR)
display(blurred_image)
```

In [16]:
```
#crop() function
print('{}x{}'.format(image.width, image.height))
```

```
800x450
```

In [17]:
```
help(image.crop)
```

```
Help on method crop in module PIL.Image:

crop(box=None) method of PIL.Image.Image instance
    Returns a rectangular region from this image. The box is a
    4-tuple defining the left, upper, right, and lower pixel
    coordinate. See :ref:`coordinate-system`.

    Note: Prior to Pillow 3.4.0, this was a lazy operation.

    :param box: The crop rectangle, as a (left, upper, right, lower)-tuple.
    :rtype: :py:class:`~PIL.Image.Image`
    :returns: An :py:class:`~PIL.Image.Image` object.
```

In [18]:
```
display(image.crop([50,0,190,150]))
```

In [19]:
```
from PIL import ImageDraw
drawing_object=ImageDraw.Draw(image)
drawing_object.rectangle((50,0,190,150), fill=None, outline='red')
display(image)
```

In [20]:
```
display(image)
```

In [21]:
```
from PIL import ImageEnhance
enhancer=ImageEnhance.Brightness(image)
images=[]
for i in range(0,10):
    images.append(enhancer.enhance(i/10))
print(images)
```

```
[<PIL.Image.Image image mode=RGB size=800x450 at 0x7FC656C2E390>, <PIL.Image.Im
age image mode=RGB size=800x450 at 0x7FC656C2E4A8>, <PIL.Image.Image image mode
=RGB size=800x450 at 0x7FC656C2E550>, <PIL.Image.Image image mode=RGB size=800x
450 at 0x7FC656C2E710>, <PIL.Image.Image image mode=RGB size=800x450 at 0x7FC65
6C2E7B8>, <PIL.Image.Image image mode=RGB size=800x450 at 0x7FC656C2EA20>, <PI
L.Image.Image image mode=RGB size=800x450 at 0x7FC656C2EA90>, <PIL.Image.Image
image mode=RGB size=800x450 at 0x7FC656C2EB00>, <PIL.Image.Image image mode=RGB
size=800x450 at 0x7FC656C2EB70>, <PIL.Image.Image image mode=RGB size=800x450 a
t 0x7FC656C2EBE0>]
```

In [22]:
```
help(PIL.Image.new)
```

```
Help on function new in module PIL.Image:

new(mode, size, color=0)
    Creates a new image with the given mode and size.

    :param mode: The mode to use for the new image. See:
       :ref:`concept-modes`.
    :param size: A 2-tuple, containing (width, height) in pixels.
    :param 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 :py:class:`~PIL.Image.Image` object.
```

In [23]:
```
first_image=images[0]
from PIL import Image
contact_sheet=PIL.Image.new(first_image.mode, (first_image.width,10*first_image.h
current_location = 0
for img in images:
    # Lets paste the current image into the contact sheet
    contact_sheet.paste(img, (0, current_location) )
    # And update the current_location counter
    current_location=current_location+450
contact_sheet = contact_sheet.resize((160,900) )
# Now lets just display that composite image
display(contact_sheet)
```


Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information
Master of Science in Information

In [25]:
```
contact_sheet=PIL.Image.new(first_image.mode, (first_image.width*3,first_image.h
x=0
y=0
for img in images[1:]:
    # Lets paste the current image into the contact sheet
    contact_sheet.paste(img, (x, y) )
    # Now we update our X position. If it is going to be the width of the image,
    # and update Y as well to point to the next "line" of the contact sheet.
    if x+first_image.width == contact_sheet.width:
        x=0
        y=y+first_image.height
    else:
        x=x+first_image.width
contact_sheet=contact_sheet.resize((900,900))
display(contact_sheet)
```

In [ ]: