Use Python to write plug-ins for GIMP

Extend GIMP to do automated tasks

Nathan A. Good March 29, 2011

This article demonstrates how to automate GNU Image Manipulation Program (GIMP) with scripting to do two simple tasks: resizing all pictures in a folder and saving them all to the same format. You will learn how to build plug-ins for GIMP using the Python scripting language. You can use your new plug-in using the `gimp` command on the command line.

This article is about using GIMP-Python, which is a set of Python modules that allow you to do programming in Python to automate commands in GNU Image Manipulation Program (GIMP). These Python modules are wrappers around the libgimp libraries. GIMP-Python is different from the Script-Fu extensions. In Script-Fu, a plug-in is used to execute scripts. In GIMP-Python, the Python script takes center stage and does the work. You can instantiate the GIMP-Python scripts from inside GIMP itself, or you can use GIMP's batch mode to start it from the command line.

In this article, you learn how to write Python code that allows you to automate two different tasks in GIMP: resizing images and saving them as different formats.

You can install and use both GIMP and Python on many different platforms, including Linux®, Mac OS® X, and Microsoft® Windows®. The cross-platform nature of both GIMP and Python means you can write complex plug-ins for GIMP using Python and be able to run them on a variety of platforms.

Overview of GIMP

GIMP is an open source image manipulation program that many people use as a viable alternative to some of the commercial offerings. GIMP handles complicated features such as layers and paths. GIMP supports a number of different image formats and comes with relatively complex filters. GIMP has strong community support and involvement, so it is usually easy to find information about how to use or extend GIMP.

See Related topics for the link to download and install GIMP on your computer.

Overview of Python scripting

Python is an object-oriented scripting language that allows you to write code that runs on many different platforms. Python has been ported to both the .NET and Java™ virtual machines, so there are many different ways that you can execute Python. See Related topics to learn how to install Python on your computer.

Many modules exist for Python that provide functionality you can reuse without writing your own (the GIMP-Python modules are an example). An index of the Python modules lists many different pre-built modules that you can use to do a variety of tasks from dealing with Hypertext Markup Language (HTML) and Hypertext Transfer Protocol (HTTP) connections to working with Extensible Markup Language (XML) files (see Related topics). You can also build your own Python modules, allowing you to reuse parts of code within your enterprise.

Similar to GIMP, Python also has significant community support. This means that you can find information, as well as download and use relatively mature tools that help you in your Python development.

Before proceeding to the rest of the article, install Python on your operating system according to the instructions on Python's site. Make sure you have Python correctly installed by opening a command prompt and typing `python --version`. The results should look something like those in Listing 1.

Listing 1. Verifying the installation of Python

```
$ python --version
Python 2.6.6
```

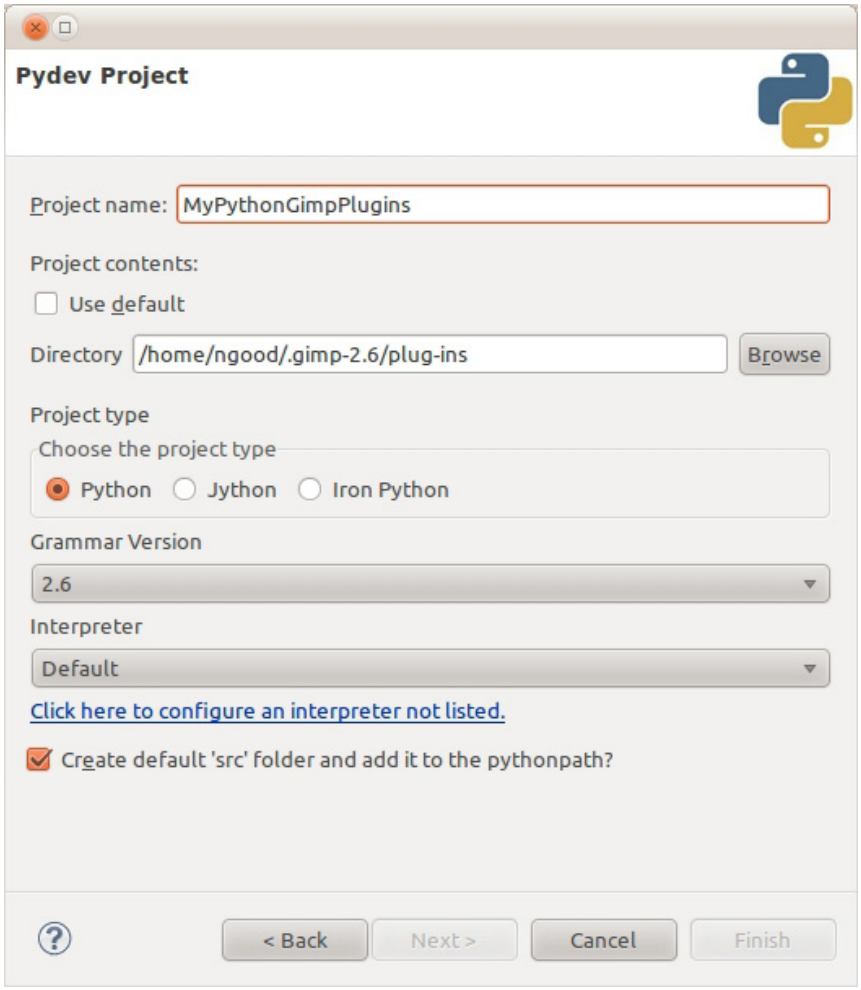
After you install the Python interpreter, you can create Python files in any text editor and run them with the interpreter. You can also use the PyDev plug-in for Eclipse, which offers syntax highlighting as well as some other features, such as catching syntax errors for you. Another option is to use the Python console directly in Eclipse, which is convenient for finding help.

The GIMP-Python modules should already be installed with newer versions of GIMP. To see if they are installed, open GIMP and look to see if you have a Python-Fu menu option under the Filters menu. If you see the option there, you are ready to start scripting. If you don't see that option, follow the links in the Related topics section to install the Python extensions for GIMP.

If you want to use the PyDev plug-in for Eclipse, follow these steps:

1. Install the PyDev plug-in by selecting **Help > Install New Software**.
2. Use http://pydev.org/updates as the update site (see Related topics).
3. Follow the rest of the installation and restart Eclipse when done.
4. After restarting Eclipse, select **File > New > Project** to create a new project.
5. Select **PyDev\PyDev Project** and click **Next**.
6. Enter your project's name (for example, *MyPythonGimpPlugins*).
7. Clear the **Use default** check box and enter the location of your GIMP directory for Python plug-ins as shown in Figure 1.

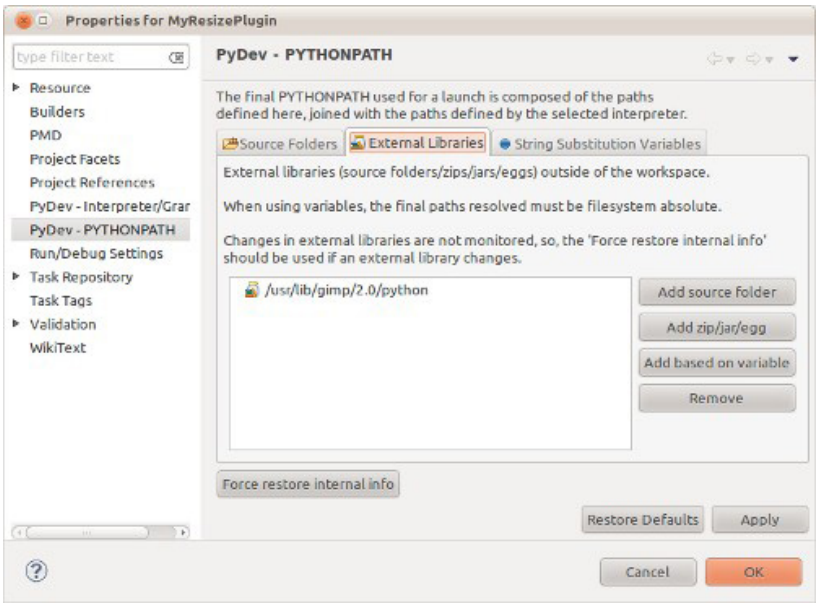
Figure 1. Creating a new project with the PyDev plug-in for Eclipse



8. Click the link to configure an interpreter. The **Auto Config** button should work as long as you have Python installed correctly and on your path.

For your project, make sure to add the folder that includes the GIMP Python modules, `gimp` and `gimpfu`. Add this directory to your Eclipse project (but don't add it to the base path) by using **Project > Properties** as shown in Figure 2.

Figure 2. Adding the GIMP-Python module directory to your project in Eclipse



Click **PyDev - PYTHONPATH**. Then select the **External Libraries** tab and click the **Add source folder** button to add the folder in which the GIMP Python modules are installed. The path will be something like /usr/lib/gimp/2.0/python/.

You can also run the Python console in Eclipse. With the console viewable, select **Pydev Console** from the list of consoles.

Registering your script

The Python files go in your user's home GIMP folder. On Mac and Linux systems, that folder is ~/.gimp-2.6/plug-ins. The Python script files should also be executable and have the Python interpreter on the first line, like the standard script declarations, as shown in Listing 2.

Listing 2. An elementary Python script that prints "Hello, world!"

```
#!/usr/bin/python

print "Hello, world!"
```

You need to register your Python script for GIMP to put the plug-in in one of the GIMP menus. Listing 3 shows the bare minimum script that you need to register a script in GIMP and print "Hello, World!" to the console when it is called.

Listing 3. Registering your plug-in with GIMP

```
#!/usr/bin/python

from gimpfu import *

def plugin_main(timg, tdrawable):
    print "Hello, world!"

register(
        "python_fu_resize",
```

```
        "Saves the image at a maximum width and height",
        "Saves the image at a maximum width and height",
        "Nathan A. Good",
        "Nathan A. Good",
        "2010",
        "<Image>/Image/Resize to max...",
        "RGB*, GRAY*",
        [],
        [],
        plugin_main)

main()
```

The `register()` method gives GIMP information about your plug-in.

The `register()` method has several paramters that tell GIMP how to display menu options for the plug-in, and what Python method to call when you start the plug-in from the menu. Table 1 shows the parameters for the `register()` method.

Table 1. The parameters and examples for the `register()` method

Parameter	Example	Description
proc_name	python_fu_resize	The name of the command that you can call from the command line or from scripting
blurb	Saves the image at a maximum width and height	Information about the plug-in that displays in the procedure browser
help	Saves the image at a maximum width and height	Help for the plug-in
author	Nathan A. Good	The plug-in's author
copyright	Nathan A. Good	The copyright holder for the plug-in (usually the same as the author)
date	2010	The copyright date
label	<Image>/Image/Resize to max...	The label that the plug-in uses in the menu
imagetypes	RGB*, GRAY*	The types of images the plug-in is made to handle
params	[]	The parameters for the plug-in's method
results	[]	The results of the plug-in's method
function	plugin_main	The name of the method to call in your Python code

You can get the most up-to-date information about the register method's parameters by opening the Python-Fu console (click **Filters > Python-Fu > Console**) and typing the commands shown in Listing 4.

Listing 4. Getting help using the Python console

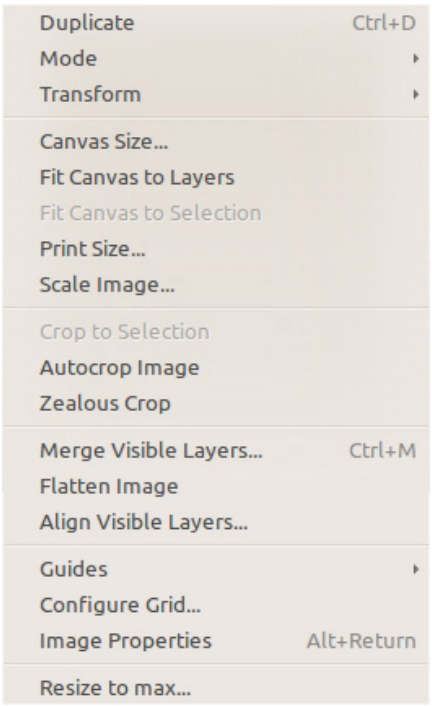
```
import gimpfu
help(gimfu.register)
```

After putting your values in the `register` method, save your script. Make sure that it is executable and is located in the .gimp2-6/plug-ins folder.

After you save the script, start GIMP from the command line using the `gimp` command. This should allow you to see any information that is printed by your plug-in, including the output of the print statement. Also, if you have any errors in your plug-in, you see them here.

With GIMP started, go to the **Image** menu where you can see the new **Resize to max** menu item as shown in Figure 3.

Figure 3. The new menu item for your plug-in



Now that your plug-in is registering itself properly with GIMP and you are able to click on a menu item for your plug-in, you're ready to proceed with adding the Python code for resizing the image.

Scripting the resize

After you have the Python plug-in in GIMP, you can add the real code into the `plugin_main` method inside your plug-in code.

Listing 5 demonstrates the code for resizing the image.

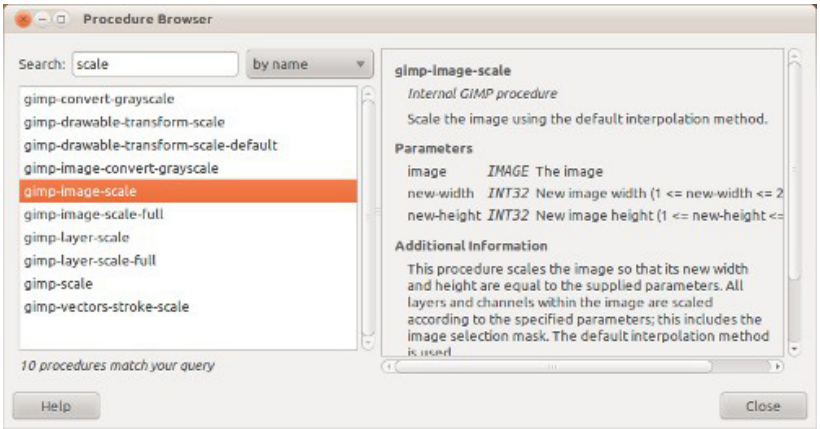
Listing 5. Adding the code in the `plugin_main` method

```
def plugin_main(timg, tdrawable, maxh=500, maxw=500):

    currentWidth = tdrawable.width
    currentHeight = tdrawable.height

    newWidth = currentWidth
    newHeight = currentHeight

    if (maxw < newWidth):
        newWidth = maxw
        newHeight = (float(currentHeight) / (float(currentWidth) / newWidth))

    if (maxh < newHeight):
        newHeight = maxh
        newWidth = (float(currentWidth) / (float(currentHeight) / newHeight))

    pdb.gimp_image_scale(timg, newWidth, newHeight)
```

The Python code is simply calling the `pdb.gimp_scale_image` method to resize the image after doing some elementary calculations to find what the values of the scaled image sizes should be. Because the values put into the box are maximum values, the script needs to check both the width and height of the current image to see if the image's dimensions need to be constrained. If either image dimension is larger than the maximum size, it sets the constrained dimension to the maximum size and then calculates the other dimension.

To find out more about other methods you can call inside your Python script, see the **Help > Procedure Browser** inside GIMP. The procedure browser for the `pdb.gimp_image_scale` method is shown in Figure 4.

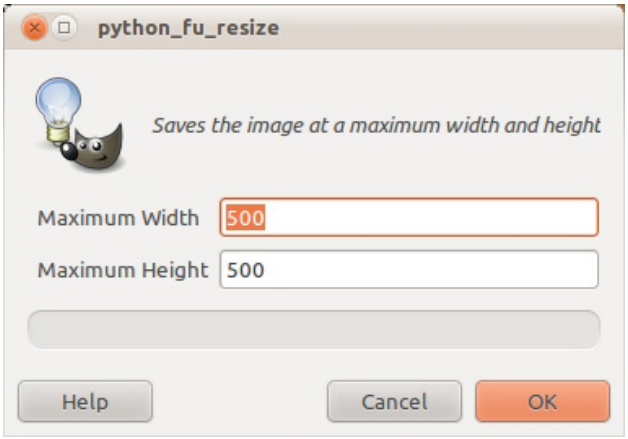
Figure 4. Viewing the gimp-image-scale method in the procedure browser



Running the resize plug-in script

After you add the code to perform the resize, open an image in GIMP. Click your new **Image > Resize to max** menu item. Your script asks you for the sizes as shown in Figure 5.

Figure 5. The input parameters for your plug-in



When you click **OK**, your `plugin_main` method executes and your script resizes your image.

Scripting the image transform

Now that you have the plug-in working to resize your image, you can update the Python script to also save the image in a different image format. This allows you to save the original image as a JPEG file as well as resize it to fit within the certain constraints.

The new additions to the script are shown in Listing 6.

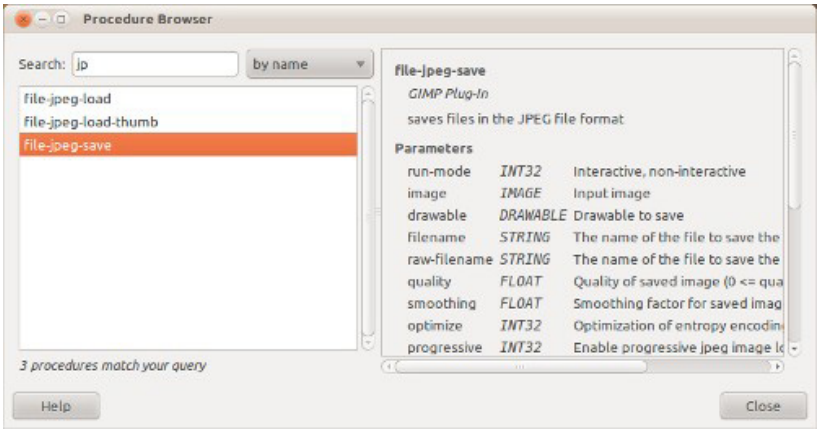
Listing 6. Adding code to save a JPEG copy of the original image

```
#!/usr/bin/python

from gimpfu import *

def plugin_main(timg, tdrawable, maxh=500, maxw=500, savecopy=TRUE):

    currentWidth = tdrawable.width
    currentHeight = tdrawable.height

    newWidth = currentWidth
    newHeight = currentHeight

    if (maxw > newWidth):
        newWidth = maxw
        newHeight = (float(currentHeight) / (float(currentWidth) / newWidth))

    if (maxh > newHeight):
        newHeight = maxh
        newWidth = (float(currentWidth) / (float(currentHeight) / newHeight))

    pdb.gimp_image_scale(timg, newWidth, newHeight)

    if savecopy:
        pdb.file_jpeg_save(timg, tdrawable, timg.name+".jpg", timg.name+".jpg",
                           0.9, 0, 0, 0, "", 0, 0, 0, 0)

register(
        "python_fu_resize",
        "Saves the image at a maximum width and height",
        "Saves the image at a maximum width and height",
        "Nathan A. Good",
        "Nathan A. Good",
        "2010",
```

```
        "<Image>/Image/Resize to max...",
        "RGB*, GRAY*",
        [
                (PF_INT, "max_width", "Maximum Width", 500),
                (PF_INT, "max_height", "Maximum Height", 500),
                (PF_BOOL, "copy", "Make a JPEG copy", TRUE),
        ],
        [],
        plugin_main)

main()
```

You can get the name of the method to use from the procedure database (the pdb variable) by using GIMP's **Help > Procedure Browser** as shown in Figure 6.

Figure 6. Looking up the method to save as a JPEG in the procedure browser



The constants used for the parameter input types come from the `gimpfu` library. You can get the list of available constants by typing the commands shown in Listing 7 in the Python console in either GIMP or Eclipse.

Listing 7. Getting help for the `gimpfu` constants

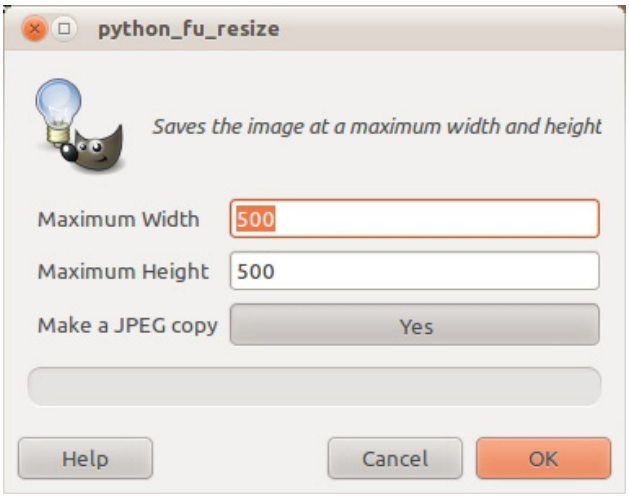
```
import gimpfu
help(gimpfu)
```

The constants begin with `PF_` and define data types that you can use for the controls on the input form.

Running the updated plug-in script

After adding the new code to the Python script to save the image as a JPEG, you can execute the plug-in by opening an image in GIMP and using the **Image > Resize to max** menu item. You see the updated parameter input box as shown in Figure 7.

Figure 7. The updated parameter input



Now that you've made the script and tried it on some images, you can run the plug-in on all of the images in a folder.

Running both on a folder

GIMP has a non-interactive batch mode that allows you to call GIMP commands from the command line. You can use the command-line feature to operate on all images in a folder using standard wildcards.

The method for saving the image as a JPEG, for instance, can be passed directly into GIMP's batch mode by using the command in Listing 8.

Listing 8. Using GIMP's non-interactive batch mode to save the image

```
gimp -i -b '(file-jpeg-save "Menu_006.png" 200 200 TRUE)' -b '(gimp-quit 0)'
```

However, this becomes a little more difficult to do when considering the calculations necessary for the size constraints. Therefore, this plug-in greatly simplifies both operations so you can call them from a single GIMP command.

Now that your plug-in is working and is registered in GIMP, the plug-in has its own command in GIMP's procedure database. You can see the command for your plug-in by going to the procedure browser (**Help > Procedure Browser** in GIMP) and typing the name that you gave your plug-in. For example, if you named it `python_fu_resize` in the register method as shown back in Listing 6, you will find it in the GIMP procedure browser as `python-fu-resize`. You call this command as it's shown in the GIMP Procedure Browser from the command line using the `gimp` command and the `-i -b` flags as shown in Listing 9.

Listing 9. Calling your plug-in from the GIMP non-interactive batch mode

```
gimp -i -b '(python-fu-resize "myimage.png" 200 200 TRUE)' -b '(gimp-quit 0)'
```

GIMP opens the image you specified, executes your command using the parameters that you provide, and then quits without saving any modifications made to the original image. By using the GIMP command in the non-interactive batch mode, you can script large-scale modifications to an entire folder full of images.

The command shown in Listing 10 operates your new plug-in's command on all Portable Network Graphics (PNG) images in a folder.

Listing 10. Calling your plug-in from GIMP on all of the images in a folder

```
gimp -i -b '(python-fu-resize "*.png" 200 200 TRUE)' -b '(gimp-quit 0)'
```

Summary

Python is an object-oriented scripting language that allows you to write scripts that you can execute on many different platforms, such as Linux, Mac, and Windows. The tool support for the Python scripting language is considerable — from simple syntax highlighting in text editors to Eclipse plug-ins.

GIMP is an application that provides sophisticated editing of graphics files on many different platforms. GIMP supports the notion of plug-ins, which provide extension points that you can use to automate even extremely complex tasks by using scripting. Because GIMP supports Python scripting in plug-ins, you can use Python to extend GIMP. Using the non-interactive batch mode, you can call your plug-ins from the command line in a method suitable for scripting.

Related topics

- Read about the different Python modules available to you.
- Learn more about the Python programming language.
- Get the Python interpreter, which allows you to run Python scripts on your computer.
- To develop Python code in an IDE, start with the Eclipse IDE.
- Download the PyDev plug-in for Eclipse to develop Python code in the Eclipse IDE.
- Download GIMP, an open source image manipulation program.
- Start developing with product trials, free downloads, and IBM Bluemix services.