Python Fu #5: Automating Workflows, coding a complete plug-in

September 14, 2015November 30, 2015 | jacksonbates | gimp, gimp scripting, python, python-fu, tutorial
See the other tutorials in this series. (https://jacksonbates.wordpress.com/python-fu-gimp-scripting-tutorial-pages/)

If you haven't been following this series and any of this doesn't make sense, go back and watch the earlier videos. In this tutorial we create a plug-in that performs two GIMP effects in quick succession, so that it can be called from the menus like every other plug-in or filter. It's not a particularly desireable effect that we are making, but it will teach us the basics.

Now what we are doing here is automating a workflow. This is what programming is: You break down the task you want to acheive into the smallest steps you can think of, and then perform those steps in the correct order. The fancy name for the end product is an **algorithm** – but all it really means is the steps the program takes. We are going to use a process called incremental development for all of our scripts which means every time we add a step to our algorithm, we test it. We keep an eye on the error console and check that it has the effect we expected. If it works, great! If not, we debug that line until it does what we want it to. It feels slow doing it incrementally, but it is much slower to write what you think is the whole script and then run it and have no clue which line caused the error. If you progress by one tested step at a time, you will know exactly which line is causing problems, and should be able to debug much quicker. In the early days – maybe for months – you will find lots of errors as you learn how Python wants to recive the information. It's a slow process but you'll learn a ton about programming – and remember everything can be Googled, asked about on a GIMP forum, or on Reddit's 'Learn Python' subreddit, or on StackOverflow – a programmer's Q&A site.

Anyway, lets make the plug-in! The code for the completed plug-in can be found here: extreme_unsharp_desaturation.py (https://gist.github.com/JacksonBates/59512101e56657578854)

So what I want this to do is run the unsharp mask filter in a very distinctive way, and then make the image black and white.

I'll set up the register for the plug-in first by editing my template and saving it with the appropriate name.

```
def extreme_unsharp_desaturation(image, drawable):
# Function code goes here

register(
    "python-fu-extreme-unsharp-desaturation",
    "Unsharp mask and desaurate image",
    "Run an unsharp mask with amount set to 5, then desaurate image",
    "Jackson Bates", "Jackson Bates", "2015",
    "Extreme unsharp and desaturate",
    "RGB",
    [
        (PF_IMAGE, "image", "takes current image", None),
        (PF_DRAWABLE, "drawable", "Input layer", None)
    ],
    [],
    extreme_unsharp_desaturation, menu="/Filters/Enhance")

main()
```

What I do next is manually do the task myself and write down in the script file in comments what I am doing, taking note of all of the parameters I use.

So what I do for this effect is run the Unsharp mask from the filters menu.
Set the radius to 5.0, the amount to 5.0 and the threshold to 0 (even if I don't change the settings, I write down the defaults…we might need them.)
Then I run the desaturate command and set the option to Lightness.

And that's it! I write that in my code as comments and then add each of the functions under each comment. This means I am commenting my code as I go. If I come back to this months from now and can't remember what I did, the comments are there to remind me.

```
# Unsharp mask from filters menu
# Radius = 5.0
# Amount = 5.0
# Threshold = 0
# Desaturate, option = Lightness
```

Now I open up the PDB and start getting the actual commands I need, one at a time.
Firstly I look up the unsharp mask. Now I can see there are two, so I need to do some digging. I go back to the one I used and hover over it. The description tells me it's the most widely used one, and sure enough, the plug-in version tells me the same thing. So I double click that to snatch the code I can use, and then I read the information in the PDB about the arguments it takes.

Now, here's a gotcha that used to catch me out. **The first argument – run-mode – is not needed**. It is there for people that run Gimp procedures from the command-line, you and I might also refer to them as wizards, but the clue that we regular folk don't need it is that the next two arguments are image and drawable – those are the two that all plug-ins that work on an existing image always begin with. The important info we learn from the PDB is that the radius and amount are FLOATS, while the threshold is an integer. You paid attention in the very first video, so you know what that means!

Now we can either put that in as it is and set the variables in our function above it, or we can hardcode the values into the arguments. Essentially, when you make the choice for yourself, make it as readable as possible.

So image and drawable can stay the same. They are actually being supplied in the parameters below in the register function, PF_Image and PF_Drawable, so GIMP knows what those variables mean. But the other three aren't established anywhere so I'll create variables to hold the values to make it easy to understand.

```
# radius = 5.0
# amount = 5.0
# threshold = 0
pdb.plug_in_unsharp_mask(image, drawable, radius, amount, threshold)
```

I'll save that and run it to check that it worked. If it doesn't you need to go back and debug it. If your code looks the same as mine, but it doesn't run check all spellings and punctuation very carefully. A misplaced comma is all it takes to grind everything to a halt – so be very precise and careful. I spent an hour looking for a comma the other day…it happens.

So now that that works, we go through the same process for the next step. Look up desaturate, notice that there are two, but one lets us set the mode to lightness, so pick that one. We can either write in the value or the number that represents the mode. I'll use the name, since it's more readable. Remember that Python wants you to convert dashes to underscores! There's another bug you just avoided!

```
pdb.gimp_desaturate_full(drawable, DESATURATE_LIGHTNESS)
```

Now if we go back to the image and try to undo that, you notice that I can only undo each step at a time. This would be annoying for you if there were more steps – we'd rather just be able to undo the effect of the plug-in in one go, right?

So the way we do that is by creating an undo group. You'll want to do this all the time. Look up undo in the PDB and you'll see there is an undo group start and end function. So we'll stick the start on at the top of the function and the end at the end. These only take the image as an argument, and since that gets passed in already, save one last time and then test the undo functionality.

```
pdb.gimp_image_undo_group_start(image)
...
pdb.gimp_image_undo_group_end(image)
```

It works!

So now you have made your first useful GIMP plug in. Using these principals alone, and some careful think and detective work, you can pretty much make anything now. We haven't looked at getting new inputs from the user yet, so we'll do that in the next video.

Related Posts: Python Fu GIMP tutorials (https://jacksonbates.wordpress.com/python-fu-gimp-scripting-tutorial-pages/)

Here's the complete code:

```
#!/usr/bin/env python

# Tutorial available at: https://www.youtube.com/watch?v=uSt80abcmJs
# Feedback welcome: jacksonbates@hotmail.com

from gimpfu import *

def extreme_unsharp_desaturation(image, drawable):
    pdb.gimp_image_undo_group_start(image)
    radius = 5.0
    amount = 5.0
    threshold = 0
    pdb.plug_in_unsharp_mask(image, drawable, radius, amount, threshold)
    pdb.gimp_desaturate_full(drawable, DESATURATE_LIGHTNESS)
    pdb.gimp_image_undo_group_end(image)


register(
    "python-fu-extreme-unsharp-desaturation",
    "Unsharp mask and desaurate image",
    "Run an unsharp mask with amount set to 5, then desaurate image",
    "Jackson Bates", "Jackson Bates", "2015",
    "Extreme unsharp and desaturate",
    "RGB",
    [
        (PF_IMAGE, "image", "takes current image", None),
        (PF_DRAWABLE, "drawable", "Input layer", None)
    ],
    [],
    extreme_unsharp_desaturation, menu="/Filters/Enhance")

main()
```