Python-fu Gimp Tutorial #1: Variables, simple math and types

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

This tutorial is designed to help people that have absolutely no experience with programming at all, in any language. The basic principles you need to understand before you can start to use Python-fu and start scripting in GIMP are actually quite easy, so it won't take long before you can understand everything you are doing in the simple scripts we'll explore in this series. Hopefully by the end of this series you should be able to write your own script from scratch that can automate a series of tasks to produce a desired effect.

But before we can get into writing those scripts, we have to learn some basic Python. If you know the basics of Python already, you can safely skip this tutorial.

We are going to look at variables, simple mathematical expressions and functions. Some of the vocabulary might be unfamiliar, but it's all pretty easy to pick up.

This blog post is a companion piece to this video tutorial:

Now, I've had both Python and GIMP separately installed on my computers for years, so I can't remember if you need to download Python separately, and I can't do a clean install of either without messing with some important customizations to both. So, the first thing you need to do is open up GIMP, go to Filters and look at the bottom end of the menu. If you can see the options 'Python-fu' which leads to Console, try running console, and if you get the console screen, which we will call the shell, then we are good to go. We don't need to download anything new for this tutorial, since GIMP has a Python console built in. If, however, this console is not available, you'll probably need to install Python. So go and download Python 2.7 (https://www.python.org/download/releases/2.7/). So, we Open up GIMP, go to Filters, Python-fu, and then Console. Each time I type something into the console, do it yourself too. Experiment with the principles and try to develop your understanding by hacking at the things we go over here. Learning a tiny bit of code and then experimenting with it will be the key to success, especially if you start modifying other people's plug-ins.

Variables

A variable is a label we can give to a value that might not always be the same everytime we run our program or script. For example, I might have a script that puts some copyright information on my photos. I could hardcode my name and the year of creation into a script so it always say 'Jackson Bates – 2015', but what about next year? Should I manually change the script every year? What if I publish the script for others to use – they don't all want to be me. So the sensible thing to do is store the name and year as seperate variables.

Note: In the examples below, the items preceded by >>> are what you type in the console. Bold text represents values returned by python console or program.

To do that I can simple write in the console:

```
>>>name = "Jackson Bates"
```

and

```
>>>year = "2015"
```

And if I want to see the results of that I can simply type:

```
>>>print name
Jackson Bates
>>>print year
2015
```

One of the cool things we can do with variables like these is add them together using the + operand (the plus sign):

```
>>>print name + year
Jackson Bates2015
```

But you can see that it jams them together because we didn’t add a space in between, so we could type this instead:

```
>>>print name + " " + year
Jackson Bates 2015
```

We can also create a new variable from the two old ones:

```
>>>copyright_info = name + " " + year
>>>print copyright_info
Jackson Bates 2015
```

Notice that each of my variables has a very easy to understand name. This is a good habit to get into. You should be able to tell exactly what a variable represents. Python is actually a pretty readable language, and you can make that easier for yourself if you add variables that are easily readable too. Notice that I also use underscores for spaces. A variable name can be split by a space in Python

So what happens if we change the year and then print the copyright_info again?
Well we can test that:

```
>>>year = "2016"
>>>print copyright_info
Jackson Bates 2015
```

You can see that the copyright_info variable didn’t change. Python doesn’t remember that copyright_info was initially built from name and year, and then assume that when one updates it should flow on. If you want the variable copyright_info to update you need to assign the values again:

```
>>>copyright_info = name + " " + year
>>>print copyright_info
Jackson Bates 2016
```

So the first thing to remember about variables is that they are an easy to remember name that represents a value that can change – i.e. it can vary (hence ‘variable’).

The second thing we have learnt is that we can perform operations on variables. We can stitch variables together when we print them, or we can stitch them together to make a new variable.
The proper word for this is 'concatenation', which is just a fancy word for sticking them together, like we did when we made the copyright_info variable by concatenating the name, a printed space, and the year.

What do you think happens if you write something like:

```
>>>print copyright_info - year
Traceback (most recent call last):
  File "", line 1, in
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Well, we get an error message. Don't be scared of these – you'll see plenty, and not because you suck, but just because that's what happens when you program. What it tells us here is that we can't use a minus sign for 'str.'

So what's a 'str'? Well there are 3 main types of variable that we'll focus on in this tutorial. There are more than that in Python, but for our purposes we'll get by with these three to begin with. We have Strings, Integers and Floats. A string, the 'str' we saw earlier, is a string of literal text. Anything you put in quotes is a literal string in Python. If you add a piece of text to another piece of text, you concatenate them – you stick them together. But Python doesn't know how to do other mathematical looking things to strings, as we saw with minus. You can divide a string by another string either, for obvious reasons. You can multiply a string by an integer though…

What effect do you think this has:

```
>>>print "hello!" * 5
hellohellohellohellohello
```

Now you may notice that 5 is not in quotation marks. When we wrote the year as "2015" earlier, we were treating it as a text string. But the 5 we just used is an integer. That means we can use it for real maths. The Python console can actually be used as a calculator:

```
>>>print 1 + 2 + 3
6
```

Which is different to:

```
>>>print "1" + "2" + "3"
123
```

An integer is a whole number, and we need to remember that because if we try to divide an integer by another integer, it will return an integer. That's ok when the first number is perfectly divisable by the second, but if you expect it to return a fraction or decimal, you will be disappointed, since they are not whole numbers.

So:

```
>>>print 10 / 5
2
```

is easy to understand

but:

```
>>>print 3 / 2
1
```

behaves oddly.

We can use the modulus operand (the percent sign) to see what is going on:

```
>>>print 3 % 2
1
```

This tells us that there is a remainder of 1. So the two together tell us that when we divide 3 by 2, there is 1 in each group and one left over.

All we really need to know about integers is that they are whole numbers and return whole numbers when used in calculations.

The final type we will look at is the floating point number, or simply float, which is a decimal number.

To set a variable as a float, we just use a decimal point – even if it is a whole number. Like this:

```
>>>number1 = 3.0
>>>number2 = 2.0
>>>print number1 / number2
1.5
```

Now we get the result we were probably expecting when we divided 3 by 2 before.

If we forget what type of variable we have set, we can use a built-in function, like this:

```
>>>type(number1)
```

This tells us it is a float. Try it yourself on variable of other types.

That’s all for this tutorial. In the next one we look at functions.

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