GIS Tips & Tricks

YoLani Martin & Al Karlin, Ph.D.
CMS-L, GISP

You May Not Be The Only One Confused About Python Formatting

One of the continuing theses of this column is that with GIS software, there are always multiple ways to accomplish the same end goal. With Python scripting, it is even more true, but with a twist. When scripting there are 101 additional things to think about. Does the function have the right inheritance hierarchy? Are there too many comments? Was that fourth IF- statement indented properly? And then there is an entire additional list of items when geographic information system (GIS) software is thrown into the mix. What worked perfectly fine in the code editor you are using for your development environment suddenly doesn't cooperate when bringing it into another GIS interface. Then, a major factor to keep in mind when developing scripts is identifying what version of Python your software is using and what format a script needs to be written in for the software to understand a particular command. For this month's Tip & Trick, we will focus on formatting strings with the F-String function.

In Python, there are three methods of formatted strings (f-strings) that can be used to format syntax and change display expressions. Each method was developed as new versions of Python were released; the intent was to make formatting simpler, but of course, with different versions, new issues and confusion can arise.

For Python version 1.0

In Python 1.0, the f-string method involves using %formatting. The percent (%) operator acts as a placeholder in a statement while the variable being formatted is then added after the % (Example 1). If there is more than one variable needed in a statement, then the variables are included after the % operator using parenthesis and commas to separate each variable (Example. 2).

Example 1. Basic use of %formatting where variable "baker" is being incorporated into a print statement.

```
baker = "cleo"
print ("Welcome to %s's bakery"%baker)
```

Result: Welcome to Cleo's bakery

Example 2. Formatting with more than one variable.

```
food = "donuts"
num = 73
baker = "Cleo"
print ("Welcome to %s's bakery!. There are %s %s in stock."%(baker, num, food)
```

Result: Welcome to Cleo's bakery! There are 73 donuts in stock.

For Python version 2.0

In Python 2.0, the f-string method involves using the string format; where curly brackets "{}" are used to contain a string variable (otherwise known as "str.format())". The {} act as a placeholder for the variable "while .format()" follows after; the variable is contained within the parenthesis (Example 3). As many variables as needed can be added within the parenthesis, where even variables in a dictionary (Example 4) can be called and formatted in a statement.

Example 3. Basic use of string format where baker, number, and food variables are applied to the print statement.

```
Food = "donuts"
Num = 73
Baker = "Cleo"
Print ("Welcome to {}'s bakery! There are {} {} in stock."format(baker, num, food))
```

Result: Welcome to Cleo's bakery! There are 73 donuts in stock.

Example 4. Accessing a dictionary with string format for the print statement.

```
bakery = {"baker":"Daniel","food":"danishes}
print ("Welcome to {baker}'s bakery! We sell {food}.".format (**bakery))
```

Result: Welcome to Daniel's bakery! We sell danishes.

For Python version 3.0

In Python version 3.0, the f-string method is similar to str.format() but now it is written out as f "{}". By using lowercase f or uppercase F in the beginning of the statement, the curly brackets containing the variable can be placed more easily (Example 5). This method makes editing syntax more efficient and easier to follow.

Example 5. Basic use of f"{}" where variables are placed throughout the print statement.

```
baker = "Cleo"
rival = "Daniel"
b1 = "donuts"
b2 = "danishes"
print (f"{rival}'s bakery is {baker}'s rival. But their {b2} can't compare to our {b1}!")
```

Result: Daniel's bakery is Cleo's rival. But their danishes can't compare to our donuts!

Photogrammetric Engineering & Remote Sensing
Vol. 88, No. 3, March 2022, pp. 145-146.
0099-1112/22/145-146

doi: 10.14358/PERS.88.3.145

Table 1. Comparison of several popular GIS software programs and compatible Python versions.

GIS Software	Version	Python Version Used	Recommended F-string to use
ArcGIS Desktop	10.0-10.8.1	Python 2.6.5 - Python 2.7.18	• % formatting • str.format()
ArcGIS Enterprise	10.0 - 10.2.1, 10.5, 10.5.1, 10.6, 10.6.1, 10.7, 10.8, 10.9	Python 2.6.5 - Python 2.7.18	• % formatting • str.format()
ArcGIS Enterprise cont.	10.4 - 10.9 *(note: versions may include upgraded set of python libraries)	Python 3.4.1 - 3.7.9	• f "{}" formatting
ArcGIs Notebook	10.7 - 10.9 *(note: versions may include upgraded set of python libraries)	Python 3	• f "{}" formatting
ArcGIS Pro	1.0 - 1.2	Python installation required	• % formatting • str.format() • f "{}" formatting
ArcGIS Pro cont.	1.3 - 2.8.3	Python 3	• f "{}" formatting
Global Mapper	v22.0 - v23.0	Python 3.9	• f "{}" formatting
QGIS	v3.16 - v3.22	Python 3	• f "{}" formatting
QGRASS GIS	7.4.4 - 7.8.6	Python 2.7 - Python 3	• % formatting • str.format() • f "{}" formatting
Microsoft Azure Maps	2.0	Python 2.7 - Python 3.8	• % formatting • str.format() • f "{}" formatting

Understanding the different methods of f-strings is not only useful for formatting scripts, but also significant to formatting tools for different GIS software systems. Not all GIS systems may have Python incorporated into their back-end code; while some GIS software may use older versions of Python. The trick to formatting lines of code with f-strings is knowing what version of Python the software uses.

Every update to a piece of software is going to come with its own positives and negatives. One update may solve developing needs and make work much easier. On the other hand, an update may have the most up to date version of Python, but may take away a feature the last version had that was needed for a special kind of analysis or cartographic work. Table 1 summarizes popular GIS software and Python versions, there are a variety of packages that come in certain software versions.

Knowing how to adapt code for frameworks is part of an application and script's life cycle. Not every project requires legacy code to function, but having a basis of what formatting is required makes the adaptation and troubleshooting process run smoother.

It is important to note that there is no wrong way to write scripts and tools; there is always more than one way to accomplish the end-goal. Rather, it is a matter of how clear they are to users and developers that determines their usability. If a project is dependent on features of one system version over another, developing code that's understandable for that specific system is essential. But where there are limits, there also lies creative bounds. Once a system's syntax limits are understood, it becomes easier to adapt code for the best use of a project or task.

Below are some additional sources for help with Python scripting.

Sources

"Arcgis Notebook Server." *Available Python Libraries-ArcGIS Notebook Server | Documentation for ArcGIS Enterprise*, https://enterprise.arcgis.com/en/notebook/latest/python/windows/available-python-libraries.htm.

FAQ: What Version of Python Is Used in Arcgis?, https://support.esri.com/en/technical-article/000013224.

Real Python. "Python 3's F-Strings: An Improved String Formatting Syntax (Guide)." Real Python, Real Python, 19 Mar. 2021, https://realpython.com/python-f-strings/.

Jablonski, J. "Python 3's F-Strings: An Improved String Formatting Syntax (Guide)." *Real Python,* Real Python, 19 Mar. 2021, https://realpython.com/python-f-strings/.

"Microsoft Azure SDK for Python." *Microsoft Docs*, https://docs.microsoft.com/en-us/python/api/overview/azure/mgmt-maps-readme?view=azure-python.

"Python in Global Mapper." *Python Scripting*, https://www.bluemarblegeo.com/knowledgebase/global-mapper-23/Python.htm.

QGIS Documentation, https://qgis.org/en/docs/index.html.

Requirements to Compile Grass GIS 7, https://grass.osgeo.org/grass74/source/REQUIREMENTS.html.

Send your questions, comments, and tips to GISTT@ASPRS.org.

YoLani Martin is a Geospatial Analyst with Dewberry's Fairfax, VA office. She is a resource for open source tools and Python scripting. Al Karlin, Ph.D., CMS-L, GISP is with Dewberry's Geospatial and Technology Services group in Tampa, FL. As a senior geospatial scientist, Al works with all aspects of Lidar, remote sensing, photogrammetry, and GIS-related projects.