Python String Format Cookbook

Posted on October 10, 2012 in Code

Every time I use Python's string format, version 2.7 and up, I get it wrong and for th
I can't figure out their documentation, I was quite familiar with the older % method
this string format cookbook as a quick reference for myself when wanting format n
anything. Thanks to other contributors I've learned and grown the examples over ti

Number Formatting

The following table shows various ways to format numbers using python's str.form
includes examples for both float formatting and integer formatting.

To run examples use `print("FORMAT".format(NUMBER));`
So to get the output of the first example, you would run: `print("{:.2f}".format(3.1`

Number	Format	Output	Description
3.1415926	{:.2f}	3.14	2 decimal places
3.1415926	{:+.2f}	+3.14	2 decimal places with sign
-1	{:+.2f}	-1.00	2 decimal places with sign
2.71828	{:.0f}	3	No decimal places
5	{:0>2d}	05	Pad number with zeros (left padding, width 2)
5	{:x<4d}	5xxx	Pad number with x's (right padding, width 4)
10	{:x<4d}	10xx	Pad number with x's (right padding, width 4)
1000000	{:,}	1,000,000	Number format with comma separator
0.25	{:.2%}	25.00%	Format percentage
1000000000	{:.2e}	1.00e+09	Exponent notation

13	{:10d}	13	Right aligned (default, width 10)
13	{:<10d}	13	Left aligned (width 10)
13	{:^10d}	13	Center aligned (width 10)

For even more power in formatting, you can use `{}` as a variable inside of the forma
brackets (Thanks to Peter Beens for this tip). Example:

```
pi = 3.1415926
precision = 4
print( "{:.{}f}".format( pi, precision ) )
~~ 3.1415
```

string.format() basics

Here are a couple of example of basic string substitution, the `{}` is the placeholder f
substituted variables. If no format is specified, it will insert and format as a string.

```
s1 = "so much depends upon {}".format("a red wheel barrow")
s2 = "glazed with {} water beside the {} chickens".format("rain", "white"
```

You can also use the numeric position of the variables and change them in the strin
gives some flexibility when doing the formatting, if you made a mistake in the orde
easily correct without shuffling all variables around.

```
s1 = " {0} is better than {1} ".format("emacs", "vim")
s2 = " {1} is better than {0} ".format("emacs", "vim")
```

Older % string formatter

Prior to python 2.6, the way to format strings tended to be a bit simpler, though limi
the number of arguments it can receive. These methods still work as of Python 3.3, b
are veiled threats of deprecating them completely though no time table. [PEP-3101]

Formatting a floating point number:

A comparison of % and format() for python number formatting.

```
pi = 3.14159
print(" pi = %1.2f " % pi)          # old
print(" pi = {:.2f}".format( pi )) # new
```

Multiple Substitution Values

```
s1 = "cats"
s2 = "dogs"
s3 = " %s and %s living together" % (s1, s2)
s4 = " {} and {} living together ".format(s1, s2)
```

Not Enough Arguments

Using the older format method, I would often get the error “TypeError: not enough
for format string” because I miscounted my substitution, do something like the follo
made it easy to miss a variable.

```
set = " (%s, %s, %s, %s, %s, %s, %s, %s) " % (a,b,c,d,e,f,g,h,i)
```

The new python string formatter you can use numbered parameters so you don’t ha
count how many you have, at least on half of it.

```
set = " ({0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}) ".format(a,b,c,d,e,f,g)
```

More String Formatting with .format()

The format() function offers a fair amount of additional features and capabilities, h
few useful tips and tricks using .format()

Named Arguments

You can use the new string format as a templating engine and use named argument
of requiring a strict order.

```
madlib = " I {verb} the {object} off the {place} ".format(verb="took", ob
~~ I took the cheese off the table
```

Reuse Same Variable Multiple Times

Using the % formatter, requires a strict ordering of variables, the .format() method a
to put them in any order as we saw above in the basics, but also allows for reuse.

```
str = "Oh {0}, {0}! wherefore art thou {0}?".format("Romeo")
~~ Oh Romeo, Romeo! wherefore art thou Romeo?
```

Convert Values to different Bases

You can use the following letters to convert a number to their bases, **d**ecimal, he**x**, **o**
binary

```
print("{0:d} - {0:x} - {0:o} - {0:b} ".format(21))
~~ 21 - 15 - 25 - 10101
```

Use Format as a Function

You can use .format as a function which allows for some separation of text and form
from code. For example at the beginning of your program you could include all you
and then use later. This also could be a nice way to handle internationalization whi
requires different text but often requires different formats for numbers.

```
## defining formats
email_f = "Your email address was {email}".format

## use elsewhere
print(email_f(email="bob@example.com"))
```

Hat tip to earthboundkids who provided this on reddit.

Escaping Braces

If you need to use braces when using str.format(), just double up

```
print(" The {} set is often represented as {% raw %}{{0}}{% endraw %}".fo
~~ The empty set is often represented as {0}
```

Reference

- Python String Library – Standard Library Documentation

- Python Essential Reference – book on Amazon
- My Python Argparse Cookbook – a similar cookbook for parsing command-line arguments

Like this:

Like

Be the first to like this.

Tags: python

xkcd graph style in d3

Python