目次へ

テンプレートから文字列をつくる

概要

ここでは、「%」演算子を使って文字列をつくることについて説明します。

```
>>> print "%s は、%s に %d のダメージをあたえた！" % ("大コウモリ", "Yama", 100)
```

って感じ。初めてこの例を見る人は意味が分からないでしょうけど。（でも、何が表示されそうか想像してみてね）

本編

文字列をうまい整形で表示できる方法のひとつとして、「%」演算子の使い方を説明しておきます。

今までは、この % 演算子は、「割り算の余りの値」だと説明してきました。これは、対象が数字のときは確かにそうです。

```
>>> 100 % 7
2
```

100を7で割ったときの商は14で、余りは2。

だけど、文字列を対象にしてこの演算子を使ったときには、まったく違う操作をすることを意味します。

```
>>> "Hello, %s" % "kitten"
'Hello, kitten'
```

「%」の左に現れるべき文字列は、「%s」という部分を含んだ文字列を書きます。「%」の右は、「%s」の代わりに入る文字列を指定しています。つまり、イメージとしては「穴埋め問題」ですね。

```
問．　法隆寺を建てたのは[　　]である。
```

こんな感じ。[　　]の代わりに、%sって書くんだと覚えれば早いです。（ちなみにこの問題の答えは「大工さん」ですね。）

こんな変なものを持ち出さないでも、別に今まで困ってなかったって？　まあそうですね。下のように書く方法をもうご存知のはずですし。

```
>>> n = "kitten"
>>> print "Hello,", n
'Hello, kitten'
```

コンマ記号「,」でつなげればいいじゃないか、と。

でもコンマ記号を使うと、自動的に半角スペース記号が一文字挟まってしまいます。これがイヤな場合というのもあるでしょうから、より汎用的な方法として、「%s」を使う方法を知っておいたほうがよいでしょう。（プラス記号で連結すれば半角スペースは混じらないぞ、という声も聞こえてきますが、後述するように、「%」をつかう方法だと、数字を扱うのも便利になるんですよ。）

それに、あとでどういう表示がされるかが、スクリプトを見たときにちょっと分かりやすいという利点もありますよ。特に、穴埋め部分が二箇所以上になっているような場合。

```
>>> n = "yama"
>>> i = "pen"
>>> "%s have a %s" % (n, i)
'yama have a pen'
```

"%s have a %s"という部分（ひながた、という意味で、テンプレートと呼ぶときがあります）を見るだけで、どんな表示結果になるかが予想しやすいと感じませんか。

```
n, " have a ", i
```

って書いてあると、ちょっと煩雑な気分でしょ。そのうちこのテンプレートを活用しながらHTMLを出力してみたりしますが、そのときになると有難味がもうちょっと分かるかも。

ああそうだ、二つ以上の穴あき部分があるときは、カッコで二つ以上の値をくるんで表現することを覚えておいてください。今はあえて「タプル」という用語は説明しませんが、いずれ。

もうひとつ、「%s」じゃなくて、「%d」という「穴埋め部分」もあると知っておいてください。こっちは文字列じゃなくて数値に使います。こんなふう。

```
>>> "The price is $%d." % 98000
'The price is $98000.'
```

テンプレートのそれぞれの穴には、文字列が入る予定か、数値が入る予定かをはっきり決めておいてください。そうしないと下のようなエラーが出ます。

```
>>> "The price is $%d." % "free"   ←数値が入るべきところに、文字列を入れようとした
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: %d format: a number is required, not str
```

このくらいの融通を利かせてくれないとは不便、でしょうか？　まあそうかも知れません。注意するしかありませんのでよろしく。

そのかわり、数値専用の「%d」はちょっと便利な機能を持っていますので、こいつもついでに紹介しておきます。「%4d」という書き方と、「%04d」という書き方です。4以外の数字でもいいですよ。

```
>>> "price is %d" % 980
'price is 980'
>>> "price is %4d" % 980
'price is  980'
>>> "price is %04d" % 980
'price is 0980'
```

それぞれなんだか微妙ですが、ちょっとした書き方の違いで結果がどう変わったか見ましょう。最初のは、いままで見たのと同じ。次のは、表示の「980」がひとつ右にずれているようですね。また最後のは、千の位までわざわざゼロがついて、0980なんて具合になっています。勘が鋭い人はおわかりでしょうが、あの指定した4という数字は、「表示を4桁にそろえる」という意味だったわけです。たくさんの行を表示したときに、最高の桁数を指定しておけば表示がきれいになるでしょ。たとえば下のように。

```
price is    33.
price is  1200.
price is   975.
...
```

こんな表示にしたいときは、「%nd」という指定が役に立つわけですね。（nは適当な数字）

また、次のような使い道も考えられます。たとえば2010年9月10日という年月日の数字を使って、'20100910' なんていう文字列を作りたい場合。なんらかの理由があって、9月も'09'としながら、年月日を必ず8桁で表示したいのだとします。年が4桁、月を2桁、日を2桁に固定してあらわしたい。つまりこうです。

```
>>> y = 2010
>>> m = 9
>>> d = 10
>>> '%04d%02d%02d' % (y, m, d)
'20100910'
```

ゼロを余分に表示したいので、「%0nd」という書き方がものを言いました。（nは適当な数字。）テンプレートの読み解きはちょっと面食らいますが、慣れればなんでもありません。

文字列テンプレートを使うためには、実際には「%s」と「%d」以外にもとてもたくさんの指定方法があります。でもまあ、この二つで基本的に充分ですから、まずは問題ありません。気になる人は、ちゃんとしたpythonの参考書をあたればいいのです。身も蓋もないんだけど。

ところで一応、最初の例が実際にはどんな表示になるかというと、

```
>>> print "%s は、%s に %d のダメージをあたえた！" % ("大コウモリ", "Yama", 100)
'大コウモリ は、 Yama に 100 のダメージをあたえた！'
```

って感じですね。対話シェルで日本語を打ち込むときに問題がある人もいるでしょうから実際に試せなくても結構ですが、スクリプトにしてから実行するときは、もちろん大丈夫ですよ。