目次へ

HTMLの自動生成ができたら、Webアプリまであと一歩

今回は練習問題ではなく、ちょっとしたデモを行います。

今まで、HTMLをpythonを使っていろいろ自動生成してみました。主に、ファイルに書き出してもらいましたね。

こいつの大枠は、まあ、大体下のような感じでした。

```
o = open("something.html", "w")
print >>o, "<html>"
print >>o, "<head>"
...
 (略)
...
print >>o, "これでおしまい　米ん団子"
print >>o, "</body>"
print >>o, "</html>"
o.close()
```

最初のopenについては、たしか、ファイルを書き出し用に開いたときの「目印」を変数oに格納する、とかそんな表現をしたと思います。

このとき変数oに入っているものは、数字でもなければ文字列でもない、リストでも辞書でもありません。やっぱりどう言っても「書き出し場所の目印」ってところでしょうかねえ。変なものですが、それでも変数に入ったのは確かです。

で、変数に入るようなものは、関数の引数にして渡してしまってもいいんですよ。だから下みたいに書き換えられます。

```
def output_html(o1):
   print >>o1, "<html>"
   print >>o1, "<head>"
```

```
  ...
   (略)
  ...
  print >>o1, "これでおしまい　米ん団子"
  print >>o1, "</body>"
  print >>o1, "</html>"


o = open("something.html", "w")
output_html(o)
o.close()
```

最初に、o1という引数を取る関数を定義します。その次からのprint文は、なんとそこで受け取ったo1に向かってモノをprintしてるではないですか。つまりここで定義したoutput_html関数は、「書き出し場所の目印」を受け取って、そこに何か出力する、という感じのものと読めます。

この関数を使う場面は、出力用ファイルを開いてそれをo変数に格納し、それを渡しながら「何か出力してね」という感じで関数を呼び、それが終わったらcloseしています。

- ところで、この関数って、何の返り値も戻してくれないんじゃない？　と思う方もいるでしょう。こういうのも実はアリです。関数の中で何か意味のあるこををしてくれるようになっていて、それだけが目当てで関数を呼んでもこれはこれで正当な使い方です。だから「関数」っていうのを数学か何かの用語だと思い込んでいると、混乱しそうですね。

さて、こう書き直してみると、何のいいことがあるのかな。

たとえばこんないいことがあると言えるでしょうか。二つの同内容のHTMLファイルを出力したいとき（下の例）。

```
def output_html(o1):
  print >>o1, "<html>"
  print >>o1, "<head>"
  ...
   (略)
  ...
  print >>o1, "これでおしまい　米ん団子"
  print >>o1, "</body>"
  print >>o1, "</html>"

# ファイルその1
o = open("something.html", "w")
output_html(o)
o.close()

# ファイルその2
o = open("anotherthing.html", "w")
output_html(o)
o.close()
```

書き出しの部分を一回だけ書いて、それを二回使いまわしました。まあ、便利かもしれないけど、そんなに大したことでもないですか。

もういっこ。できあがってファイルに書き出す前に、コマンドプロンプトの上だけで出力を確認しながらプログラムを作る、という役にも立ちます（下の例）。

```
def output_html(o1):
  print >>o1, "<html>"
  print >>o1, "<head>"
  ...
   (略)
  ...
  print >>o1, "これでおしまい　米ん団子"
  print >>o1, "</body>"
```

```
  print >>o1, "</html>"

# ファイル書き出し（完成してから生かす予定）
#o = open("something.html", "w")
#output_html(o)
#o.close()

# 動作確認中は、コンソールに出すだけ
import sys
output_html(sys.stdout)
```

importとか、sys.stdoutのことを紹介するのは初めてです。

えーと、今まで普通にprintしてた時、つまり＞＞なんちゃら, って書き方をしないで、素朴にprintしてた時のことを思い出してください。アレは実は、暗黙に sys.stdout っていう「書き出し場所の目印」に向かってprintしてたのです。つまり下の二つの文は同じなのです。

```
>>> print "hello"
hello
>>> print >>sys.stdout, "hello"
hello
```

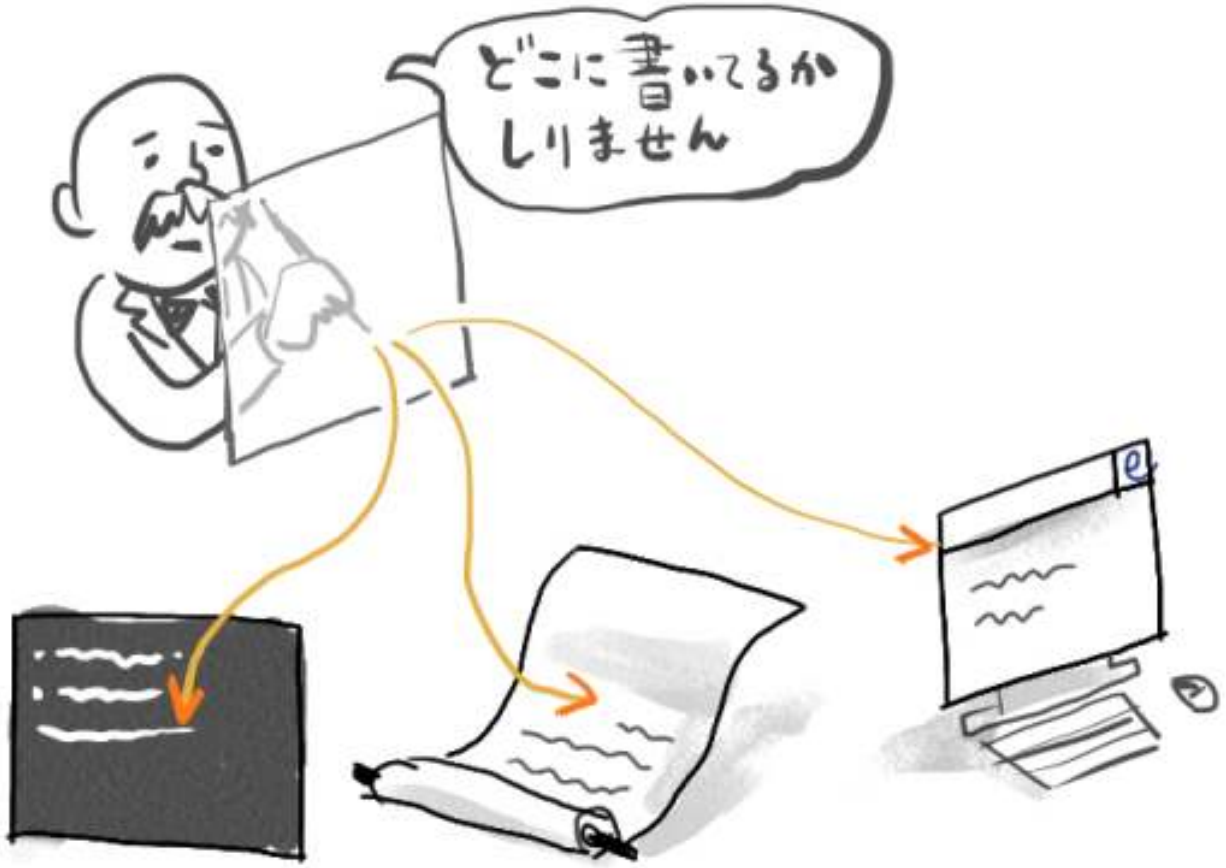
で、sys.stdoutっていう特殊な名前の変数を使うときは、あらかじめ「sys. で始まる機能を呼び出すからね」という宣言が必要なのです。これが import sys という文です。通常はスクリプトの一番最初に一回だけ書くものです。今はなんとなく真ん中辺に置きましたが。

さあ、上のスクリプト、最終的にはHTMLをsomething.htmlという名前のファイルに書き出す予定なんですが、まだ行頭に#をつけて実行できないようにしていますね。その代わり、動作確認用の部分が今のところは実行されます。output_html関数は、とにかく明示的に「書き出し場所の目印」を受け取って動くという仕様ですから、暗黙に、というわけにいきません。だから呼び出しのときに明示的に sys.stdout に書き出すように指定しているわけです。で、コマンドプロンプト上に出てくるHTMLが満足なものになったら、動作確認用の2行分を#で実行停止して、今まで止めていた3行の#を取り払えば見事ファイル書き出しに成功する、という仕掛けです。

このとき、output_html関数は、引数として与えられるものが「ファイル」だろうが「コンソール」だろうが構わずに、とにかくそいつに向かってprintしまくるだけです。なんだこいつ。不思議なやつだ。自分が何をしているのか自覚しないんだぜ。お構いなしだぜ。まさにこいつは組織の歯車ってやつだぜ。そう思いませんか。そうでもないですか、そうですか。

もっと面白いことに、このoutput_html関数は、そのまま「ウェブブラウザ」に向かってネットワーク越しにprintしてしまう、なんていう芸当までも（当人の知らない間に）こなしてしまうのです。

不思議なウェブサーバー



ってことで、今とほとんど同じような作りなのに、本物のウェブサーバーとして動いてしまうというデモスクリプトを作りました。

tiny_web_demo.py

```
# coding: utf8

def output_html(o1):
  print >>o1, "<html>"
  print >>o1, "<head>"
  print >>o1, '<meta http-equiv="Content-Type" content="text/html; charset=utf8" />'
  print >>o1, "</head>"
  print >>o1, "<body>"
  print >>o1, "<h1>俺は何をしてるんだ？</h1>"
  print >>o1, "</body>"
  print >>o1, "</html>"

#
# 極小ウェブサーバー(ここは理解しないでもOK!)
#
from BaseHTTPServer import *

class MyHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    self.send_response(200)
    self.send_header("Content-Type", "text/html")
    self.end_headers()
    output_html(self.wfile)

HTTPServer(('', 8700), MyHandler).serve_forever()
```

こいつを適当なところにダウンロードするかコピペして、アイコンをダブルクリックするなどして実行してみましょう。このwikiの仕様上、スクリプトはUTF-8エンコードでダウンロードされます。（さっそく出たぜ、エンコード）

ミスがなければ、何も字のないコマンドプロンプトがじっと表示されるでしょう。実はこの瞬間、超簡易ウェブサーバーが動き始めたのです。（あ、これ以外に、選択ダイアログが表示され

て、「アクセスを許可しますか？」とかいう質問が出てくるかもしれません。そしたら、「はい」と押してやってください。「はい」じゃなくて「今だけ許可する」とかそんなのでもいいです。）

で、この真っ黒い画面は**閉じずに**放っておいて、ここで最寄りのウェブブラウザ（今使っているやつでいいです）の新しいウィンドウを生成し、アドレス欄に下の文字列を入れてみましょう。

```
http://localhost:8700/
```

output_html関数の叫びが確認できたら、黒い画面を「閉じる」ボタンで終了してしまってよいです。超簡易ウェブサーバーが終了します。

…

あらためてデモスクリプトを眺めてみてください。HTMLを書き出している関数は、今までの例と全く同じでしょ。受け取った「書き出し用目印」に向かってHTMLをprintしただけです。下半分の「極小ウェブサーバー」以下のスクリプトを詳しく理解する必要はありませんが、大体書いてあることは「ウェブサーバーを立ち上げて、ブラウザがアクセスしてきたら、返事をする仕事を output_html 関数に振ってしまう」という程度のことです。けっこうなムチャ振りでしたが、ちゃんとこなしたでしょ。

今後、pythonとか他の言語でいわゆる「ウェブプログラム」「ウェブアプリ」みたいなものを書いたりする人もいるかも知れませんが、Webアプリのすごく原理的な話までさかのぼれば、結局これに似たことをするってことなんですよ。HTMLを書き出す。相手がなんだろうと、とりあえず書き出す。書き出したら、玄妙不可思議な仕組みがそれをウェブブラウザまで持ってってくれる。まずはそんな理解で、ここでいうoutput_html関数みたいなのを書けばいいだけなんです。

ってことで、この話はいったんここまでです。このノリを引き継いで、Webアプリの作り方とかにあらためて触れるときも、今後出てくるかもしれません。