目次へ

添付ファイル・BASE64

pythonでメールを送信する方法を前回やりました。今回は添付ファイルのくっついた電子メールを作る方法を紹介します。

電子メールってのは詰まるところテキストファイルだと、さきに述べました。でも、添付ファイルってのはいつもテキストファイルみたいなものと限りません。ワードやエクセルで作った文書とか、写真の画像とかといったものは、無理にエディタで開くとぐしゃぐしゃに壊れてしまう、いわゆる「バイナリファイル」というものです。どうやって電子メールにできるんでしょう。

ということで、まず踏まえるべき知識として、BASE64という技術を紹介します。

BASE64

エディタの上で内容がぐしゃぐしゃに壊れてしまうのは、まともな文字として扱われない文字コードがたくさん含まれているからです。ファイルの中身は、0 から 255 までの256種類のコードが色々な順番で並んでいるというのが実際の姿です。0 とか 1 とか 2 とかといったごく小さなコードは、それに対応する文字を持ちません。また、128を超えるものは、ASCII文字で表現できず、強いて言えば日本語をエンコードしたときに規則的に現れることがある程度です。これらが無秩序に交じっていたら、とてもまともな文字として表示ができないわけです。

なので、それを 64 種類だけの文字の組み合わせで表現しようというのが、BASE64 です。64文字くらいなら、まともな文字、しかも英数字だけで間に合いそうですね。

どうやって 64 種類の文字で 256 種類のコードを表すか。ちょっとだけ数学ぽい話になりますが、64の4乗と、256の3乗は同じであることに注意してください。なんならpythonの対話シェルで確かめてみましょうか。

```
>>> 64 ** 4
1677216
>>> 256 ** 3
1677216
```

- あ、「べき乗」を表す演算子が ＊＊ であることって、はじめて紹介しますね。この際覚えてください。「すごい掛け算」って感じの記号ですね。

ってことは、64種類の文字が4文字集まると、256種類の文字（コード）三つ分を表すことができることになりますね。（この理屈がわからなくても、別にいいですよ）

これをうまくやってくれるのが、今から紹介する base64 モジュールです。

まあ、対話シェルから例を試してみれば実感が持てるでしょう。

```
>>> import base64
>>> base64.b64encode('hello, world')
'aGVsbG8sIHdvcmxk'
```

base64モジュールの中にある、b64encode という関数が、文字列を受け取ってBASE64で処理してくれる機能を持っています。'hello, world' という文字列が、謎めいた文字の集合に化けてしまいました。

通常はまともに表示できないような文字列をつくって、こいつを処理したらどうなるか。

```
>>> s = ''.join([chr(i) for i in xrange(40)])
>>> print s
 （なんともいえない変な出力！）
```

最初の、s に変な文字列を入れるための命令は、理解しないでもいいです。リスト内包という表記テクニックなんですが、こんなのを実用的に使いこなす必要はそんなにないですから。ともあれ、文字コード 0 から 文字コード 39 までをつなげて、長さ40の変な文字列を作ったんです。

printした結果は無残なものでしょう。コンピュータによっては、「ピー」という音もなったりしませんでしたか。文字コード 7 って、表示どころか「ビープ音を鳴らす」っていう意味があったりして、とても目で確認できるものじゃありません。

この変な文字列も、BASE64すれば、変な文字列ではあるものの、曲がりなりにも読めるものになります。

```
>> s2 = base64.b64encode(s)
>> s2
'AAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJw=='
```

4文字で、3文字分を表そうとするのだから、BASE64でできあがった文字列は、もとのやつより若干長くなります。理論的には、1.333倍くらいになる感じかな。端数を扱うとそれよりもうちょっとだけ長くなりますけど。

```
>> len(s2)
56
```

で、このBASE64で出来上がった文字列を、もとの 0 から 255 までの本来のコードに戻すやつが、b64decode です。

```
>> s3 = base64.b64decode(s2)
>> print s3
 （もとどおりの変な出力！）
>> s == s3      ←もとの文字列と同じかどうか確かめてみる
True
```

こういうわけで、どんなファイルを電子メールにくっつけるにせよ、BASE64を通してしまえばちゃんと扱えるようになるというわけ。電子メールの本文は、文字コードが 0 から 127 までででないといけないといった話をさっきしましたが、BASE64後のデータはこの要件も満たします。

文字列そのものは、コードが 0 から 255 のうち、どんな文字でも正確に入れることができることには注意してください。無理に表示してみようとするからおかしくなっただけであって、中身自体はちゃんと秩序立ったものです。だから、ファイルの中身を(たとえそれがバイナリファイルでも)文字列として読み込むことはまったく自然なデータ操作です。

ちょっと練習でもしてみましょうか。練習問題としてナンバリングはしませんが。

下からダウンロードできるテキストファイルは、実はJPEG画像をBASE64処理したものです。こいつを元に戻してみましょう。

　hiddenpic.txt

手順としては、ファイルの中身を、改行とかを含めて全部をひとつの変数に読みこんでしまったあとで b64decode して、そいつを新しいファイルにいっぺんに書きだします。いっぺんに読み込む方法はたしか紹介したことがあります。ファイルをopenした目印（ファイルオブジェクト）を f に入れたとしたら、f.read() とすれば全部読み出せます。書き出すほうは、print を使う方法だけを今までやりましたが、あれは改行コードが勝手に混じってしまうのでちょっと嬉しくありません。改行なしで書きだしたい時は、下のような書き方をします。

```
o = open("some_file", "wb")
o.write(content)
```

read に対応して、write っていう書き方があるんです。ファイルを書き込み用に開いてから、こんな感じで write すれば、改行抜きで純粋にデータの部分だけを書きだすことができます。画像みたいなバイナリファイルは、途中で余計な改行コードがあるだけでも、変な動作を招きかねないものです。

あれ、書き出し用にファイルを開くときは、 "w" だけで足りるんじゃなかったっけ？　そうなんですが、出力されるデータがテキストファイルではない（ここでは画像ファイル）をきは、バイナリファイルですよという意味で "b" も一緒につけるという決まりになっています。（たぶん改行コードまわりの処理が違うんだと思いますが、ちゃんと追求したことはないです。ま、クセとして覚えてね）

さあ、元のファイルはJPEGですから、あとでダブルクリックとかで簡単に確認できるように、適当なファイル名に出力するにしても、最後が ".jpg" とかで終わるようにするとよいですよ。どうですか、できましたか。

で、添付ファイルのつくりかた

こんなBASE64の知識を得たところで、添付ファイルのくっつけかたの話に戻ります。

文字だけの電子メール（メッセージオブジェクト）のときは、メッセージオブジェクトをひとつ作れば済みました。

文字の電子メールに添付ファイルをひとつくっつけるときは、メッセージオブジェクトを全部で三つつくる必要があります。ふたつじゃなくて三つです。すなわち、「文字で構成されたメッセージオブジェクト」、「添付ファイルをBASE64化したメッセージオブジェクト」、そして「ふたつのメッセージファイルを束ねるための、親メッセージオブジェクト」です。

親メッセージオブジェクトが出てくるのがミソです。

で、例として、JPEG画像を添付したメッセージを作るためのサンプルを下に示しますから、見てみましょう。

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

msg_parent = MIMEMultipart()
msg1 = MIMEText('hello')
msg2 = MIMEImage('.....................', 'jpeg')

msg_parent.attach(msg1)
msg_parent.attach(msg2)

print msg_parent
```

msg_parent が、親メッセージです。これは中身を何も決めずに、単に MIMEMultipart() とだけして呼び出せば発生させることができて、このサンプルでは msg_parent 変数の中に入ります。

msg_1 が、さっき作ったのと同じような、文字が入ったメッセージオブジェクトです。ここでは一番簡単に、'hello' という文字を含むものを作りました。日本語の本文を含むときは、前にやったようなちょっとした小細工が必要になるんですが、ここでは単純に示すためにこんな感じに済ませています。

msg_2 に、画像ファイルを含むメッセージオブジェクトを作りました。'.....' ってのは、実際には画像ファイルの中身が入るべきです。今回はあくまで例として示すためにこうしましたが、本当はちゃんと意味のある画像データを入れないとだめですよ。画像ファイルを（"b"指定つきで）開いて、read() した内容を入れます。'jpeg' の部分は、あらかじめ画像がどんな形式のものなのかを調べておいて、'jpeg' とか 'gif' とか 'png' とかを適宜入れます。「自動判別」ってのが実はあるらしいんですが、今のところは人間があらかじめ書き込んでおきましょう。

- ご存じでない方のために、画像ファイルのいくつかの形式についてあらかじめ説明をしておかなければいけなかったですね。うーん、いつかやります…

msg_1 と msg_2 ができたら、親メッセージにこれを添付(attach)します。msg_parent.attach( ) という書き方でこれを行います。親メッセージにふたつのメッセージを添付する点に注意してください。テキストメッセージに画像メッセージを添付する、のではないのです。カラッポの内容をもつ親メッセージに、テキストメッセージと画像メッセージを兄弟関係のように添付するのです。

こんな風にしてできあがったメッセージオブジェクト（親）を print して確認しているのが最後の行です。実行結果はこんなふう。

```
From nobody Mon Feb  7 19:10:41 2011
Content-Type: multipart/mixed; boundary="===============0676479677=="
MIME-Version: 1.0

--===============0676479677==
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit

hello
--===============0676479677==
Content-Type: image/jpeg
MIME-Version: 1.0
Content-Transfer-Encoding: base64

Li4uLi4uLi4uLi4uLi4uLi4uLi4uLg==
--===============0676479677==--
```

なんか、==========0676479677== とかいう文字が、複数のメッセージを隔てる「仕切り」のようになっているように見えますね。数字の部分は違っているかも知れません。詳しいことは省きますが、大体こんなふうに添付ファイルのついた電子メールってのは表現されるのです。画像データの部分は、Li4uLi4uLi4uLi4uLi4u… とかいった感じに変換されて表記されています。これは元のデータがBASE64化されたからです。画像を扱うための MIMEImage は、自動でBASE64の処理も行ってくれます。今回の元データは意味がないものの上に短いですからこんなものですが、実際には大きなデータは長い長いメッセージになりますよ。

ここまでできたら、あとは電子メールを出すときの手順に従って、From とか To とか Subject とかを適宜設定して、smtplib を使って送ってしまえばおしまいです。あ、From とか To とかを設定するのは、親メッセージに対して行ってくださいね。テキストメッセージの部分に設定してもたぶんうまくいかないですから。

画像ファイルのときだけを例では示しましたが、音声ファイル、Excelシート、PDFなど、どれでも要領は同じです。ただし、インポートする関数がちょっとづつ違いますので、下にコードの例だけ書いておきます。

```
# 音声ファイル(mp3とかmidiとか)
from email.mime.audio import MIMEAudio
m = MIMEAudio('....data.....', 'x-mp3')

# エクセルシート(Office XP以前)
from email.mime.application import MIMEApplication
m = MIMEApplication('....data.....', 'vnd.ms-excel')

# PDF
from email.mime.application import MIMEApplication
m = MIMEApplication('....data.....', 'pdf')
```

ファイルの種類で、x-mp3 とか、vnd.ms-excel とか、こういう暗号じみた文字列はどうやって調べるんだ、と言われそうですが、これはもう標準として決まっている一覧表があるんです。

http://www.iana.org/assignments/media-types/
[http://www.iana.org/assignments/media-types/]

ここから探せばいいですよ。これを見ると、ファイルタイプごとに MIMEAudio とか MIMEApplication とかを使い分ける基準もわかってくると思いますよ。

これで添付ファイルのくっつけかたは説明したかな…と思ったけど、ひとつ抜けていました。このままだと、メールを受け取ったときにどういうファイル名で添付されているかが決まりません。添付ファイルって、ちゃんとファイル名が入っているものですからねえ。

さっきの画像を添付するサンプルをちょっと直します。

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

msg_parent = MIMEMultipart()
msg1 = MIMEText('hello')
msg2 = MIMEImage('.....', 'jpeg', filename="hotpicture.jpg")
msg2.add_header("Content-Type", "attachment", filename="hotpicture.jpg")

msg_parent.attach(msg1)
msg_parent.attach(msg2)

print msg_parent
```

MIMEImageを呼び出すときに、ちょっと引数を増やしました。filename="hotpicture.jpg" という部分です。なんだ、この引数の書き方は？　と思われるかもしれません。これはまだこのテキストで一度も説明したことのない、「名前つき引数」という技法です。詳細はともかく、こんな書き方をすると、添付ファイルのファイル名まで指定することができますよ。

さらにその下に一行書き足したとおりなんですが、画像メッセージに対して add_header という命令を発行します。これも同時に書き足すようにしましょう。"Content-Type" とか "attachment" とかいうところは変更する必要がなくて、filename="….." という部分がさっきと同じですね。

filename="….." の中身が日本語になるとき（例：今月の請求書.xls）は、ファイル名を丸ごと、前回説明した Header 関数でちゃんと扱える形に直しておく必要がありますよ。面倒ですね。面倒なので、あまり日本語のファイル名なんかつけないほうがいいですよ。

けっこう長々と説明してしまいました。電子メールの送信の仕方についての一連のお話でした。

最後にズバリ、日本語をちゃんと扱って、添付ファイルをちゃんと扱って、それで実際のメール送信までを行うようなサンプルスクリプトを書いておこうかなと思いましたが、やめときます。これをまじめに読んでくれる人には問題ないんですが、誰かがググってメール送信のサンプルだけを探してここを見つけて、それでスパムを送るツールなんかに使われたらいやだなあ、とも思ったり思わなかったりするもので。組み立てるべき要素はすべて説明したと思うので、あとは実際に必要なかた、または好奇心のあるかたは色々チャレンジしてみてください。

練習問題とかも、まあ、ないなあ。筆者になにかメールを送りなさい、とかいう課題じゃあおかしいしね。