目次へ

pythonで電子メールする

さあ、pythonを使って電子メールを送信する方法をやってみたいと思います。受信については、今はやりません。ほとんどの場合、何か自動でやりたいことって、メールの「送信」のほうでしょうし。

pythonでメールを送信するのは、基本だけならかなり簡単です。簡単すぎますので、くれぐれも迷惑メールを出しまくるような用途に使わないでくださいね。キリンさんアイコンとのお約束です。

まずは、やりかたそのものに先立って、電子メールの仕組みについてある程度の説明をしておこうと思います。

電子メールについて知っておくこと

まず、電子メールってのは、テキストファイルです。特定のルールに従うように書かれたテキストファイルではありますが、今まで扱ったテキストファイルと大差がありません。かならず個々

のファイルの形で格納されるとは限りませんが、やっぱり基本的な捉え方としては、メールの一通づつが、ひとつづつのテキストファイルです。

電子メールを送信するということは、誰かが作った「電子メール形式のテキストファイル」を、インターネットの向こうの誰かに届けるということです。この形式のテキストファイルを「メッセージ」とでも呼びましょうか。

あなたが使っている、電子メール用アプリケーション（Windows Mail, Outlook Express, Eudora, Thunderbird, Shuriken, 秀丸メール, 電信八号, 等々）は、メッセージファイル（またはファイル相当のもの）をたくさん保存してユーザーに見やすく提供したり、ユーザーが入力した文面を電子メール形式のテキストファイルに整えて、送り先の人にインターネット経由で送信するという仕事の補助をしているというわけです。

pythonからメールを送信したときは、この普段使っているメールソフトの「送信済みトレイ」とかにはメッセージが残りませんので注意してくださいね。これはあくまでこのメールソフトの固有の機能ですから。

で、メールを相手に送信するという行為は、通常、それの仲介をしてくれるサーバーに対してメッセージを送信することです。送信しようとする相手がいつもコンピュータの電源を入れて待ちうけてくれているとは限りませんし、仲介というものがあるのがミソです。「わたしが送って、あなたが受け取る」という経過をもうちょっと詳しく記述してみましょう。

- わたしが、電子メール（メッセージ）を、最寄りのSMTPサーバー（後述）に送信する。通常、この作業は電子メール専用アプリケーションを使って行う。
- そのSMTPサーバーが、送り先（あなた）が普段使っているメールボックス（後述）を管理するSMTPサーバーにメッセージを転送する。
- あなたは、自分のメールボックスを管理するPOP3サーバー（後述）等と通信し、自分あての最新のメッセージを手もとに受け取る。通常、この作業は電子メール専用アプリケーションを使って行う。

こんなところです。

特に注意すべきは、「わたし」から直接「あなた」に通信するわけでなく、それを仲介するSMTPサーバーとかメールボックスとかそういうものがある点でしょうか。

SMTP云々ってのはあとまわしにして、メールボックスってのをまず知りましょう。メールボックスってのは、あなた宛てのメッセージが溜まっている、どこかの場所のことです。メールサーバーとよばれる、どこか離れたコンピュータの中にあるのが普通でしょう。何にしろ、あなたは直接メールボックスの中身を調べることはできませんし、調べる必要もありません。メールボックスとあなたの間には、POP3サーバーとかIMAPサーバーとかそういう名前のプログラムが仲介に入ってくれて、あなたのリクエスト（最新のメールが見たい、どれそれのメールは消したい、等）に応えてメールボックスを操作してくれるわけです。

だから、電子メールソフトを設定するときって、「POP3サーバーの場所を設定してください」とかそんな指示があるでしょう。接続しているインターネットプロバイダが、そんなマニュアルを

くれませんでしたか。あれはメールボックスにアクセスするために必要な情報だったのです。（ついでに、POP3とかIMAPは、ユーザーIDとパスワードを内部的に尋ねてきます。これもちゃんと設定しないとメールは読めません）

まあ、メールを読むほうは、そんなもんか、くらいに考えておけばよいです。今回一番の問題にするのは、メールを出すほうです。

メールを読むほうに対応して、出すときに通信する相手は、SMTP(Simple Mail Transfer Protocol)サーバーです。あなたは、誰に電子メールを出すときでも、決められた一種類のSMTPサーバーに通信すればいいです。最寄りのポストに郵便物を放り込む感じに似ていますね。SMTPサーバーは、宛先のメールボックスをつかさどるSMTPサーバーを探して、そいつにメールを勝手に転送してくれますから、どこのSMTPサーバーから開始してもちゃんと宛先に届くという仕組みです。郵便局どうしが荷物をやりとりして担当の郵便局に届けてくれるのに似ていますね。

そんなわけで、pythonを使って電子メールを送信するとは、すなわち

- 一定の形式を守ったテキスト（ファイル）をこしらえて、
- あらかじめ指定されているSMTPサーバーと通信

すればいいんだということまで分かりました。POP3サーバーとかと同様に、インターネット接続プロバイダが、SMTPはここに設定してくださいね、というマニュアルなんかを渡してくれているでしょう。

SMTPという通信方法の特徴は、ユーザーIDとかパスワードとかが要らないという点です。最近はそうでないものも多いですが、それでも非常にたくさんのSMTPサーバーが、認証なしでだれでも郵便ポストに封筒を放り込み放題といった状態です。電子メールの仕組みはインターネットの中でも一番古いもののひとつであり、そのころはこれでも大丈夫だったんですが、今ではそれが 、電子メールというシステムの大きな欠点とみなされています。迷惑メール（スパムとも呼ばれますね）が届きまくってイヤで仕方がない方もいるでしょう。その問題の根っこはここらへんにあります。まあこの話は余談です。

pythonでやる

pythonの標準配布モジュールの中には、

- メッセージを楽につくる機能を提供してくれる機能(emailモジュール)と、
- できあがったメッセージをSMTPサーバーに送ってくれる機能(smtplibモジュール)

を提供してくれるモジュールがどっちもバッチリ入っています。今回やってみようというのは、これらのモジュールの使い方です。

メッセージをこしらえる

まず最初に、メッセージをつくる方法。

できあがる予定のメッセージテキストは、簡単なものなら、例えば下のような感じです。

```
From: pytext@giraffe.topaz.ne.jp
To: you@doko.zo.com
Subject: hello

You won 10000000000000$
access this web site NOW! http:/.......
```

テキストの前半は、メール送信者とあて先、タイトル、日付、その他のいわゆる「ヘッダ情報」というやつです。コロン記号がそれぞれの項目名のあとについているというのが特徴ですね。最初の空の行がみつかると、ヘッダの部分は終わりです。

で、テキストの後半は、メール本文です。他になにか呼び方があるのかもしれませんが、まあ「本文」でいいんじゃないでしょうか。

なんだ、こんな程度のテキストなら、大層なモジュールなんか使わなくても作れるな、とお考えになる人もいるでしょう。実際のところそのとおりで、簡単なメッセージテキストを作るくらいのことは、自力で文字列を組み立てて作ったってOKです。

でも添付ファイルとかを電子メールにくっつけたいときは、なかなかうまく作るのは面倒ですから、そういうときにはこのモジュールを使ってもいいですね。（そう、添付ファイルがくっついた電子メールも、それで丸ごとひとつのテキストファイルとして表現されるんです。ちょっと驚きでしょうか）

また、タイトルとか本文に日本語を含めるときも、いくつか気にすべき点がありますから、専用のモジュールを使って正確にメッセージを作るのがいいでしょうね。

で、具体的なモジュールの使い方に入ります。使うモジュールは、email.mime.text です。ここから MIMEText という名前の関数を使うことになりますから、まず行うべきインポートはこんなふう。（対話シェルから試してみましょう）

```
>>> from email.mime.text import MIMEText
```

MIMEってのはなんだい、といわれそうです。まあ、日本語の電子メールを作成したり、添付ファイルつきの電子メールを作成するときの書き方ルールみたいなもの、と言えばいいですか。あまり気にせずに、電子メールといえばMIME、くらいに覚えてもいいです。読み方は知りませんが、筆者は「まいむ」って言ってます。

で、この関数を実行すると、メッセージ型のオブジェクト（こういう言い方はもう大丈夫ですか）が発生します。使うときは、この結果を変数で受け止めるべきですね。

```
>>> m = MIMEText('hello')
```

MIMEText関数の引数は、電子メールの本文です。ここでは hello というひと言だけのメールってことにします。複数行のメールならば、改行文字 '\n' を途中に入れながら書くことになります

ね。または、""" をうまくつかうと、実際の改行をそのまま関数の引数に混ぜることができますね。下みたいに。

```
>>> m = MIMEText("""line 1,
... line 2,
... line 3""")
```

妙な感じでしょうが、""" はこんな風に使ってもいいのですよ。もっとも、ちょっと分かりにくくなりますかね。

なんせこの時点で、m という変数の中はメッセージファイルっぽいものとして準備されます。送信者も送信先も指定しないのではまだ不完全なんですけど、ちょっとそれっぽい感じですよ。下のようにして試してみましょう。

```
>>> print m
From nobody Wed Feb 02 19:44:59 2011
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit

hello
```

おや、m の中身は（複数行の）文字列なのかな、とお考えになりますか。m は MIMEText という種類のオブジェクトで、文字列とは異なります。でも、print されると文字列として表示されるという特徴を持っています。

それぞれのヘッダの意味はまだ分からないでしょうが、まずはこんなところ。

で、このメッセージに「送信者」「送信先」「メールのタイトル」をつけましょう。これは、m なら m というオブジェクトを、あたかも辞書のように扱って、下のような書き方でセットします。From は送信者、To は送信先、Subjectはメールの標題です。場合によっては、Cc とか Bcc なんかも同じように設定します。送信先が複数あるときは、コンマ「,」で区切って全部書けばいいようです。

```
>>> m['From'] = 'pytext@giraffe.topaz.ne.jp'
>>> m['To'] = 'dare@zo.ne.jp'
>>> m['Subject'] = 'Greeting'
```

へえ、なんでこんな辞書みたいな書き方ができるの？　ともし問われたら、それが MIMEText オブジェクトの特徴だからです。文字列のように見えたり辞書のように見えたり、なかなか不思議なシロモノですね。

こうしてから改めて m を print してみると、ヘッダの行が増えています。

```
>>> print m
From nobody Wed Feb 02 19:48:41 2011
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
From: pytext@giraffe.topaz.ne.jp
To: dare@zo.ne.jp
Subject: Greeting

hello
```

ここまでできれば、メッセージは完成です。あとでSMTPサーバーと通信するときは、こいつをそのまま送信してしまうことができます。今のところは、変数 m の中にそれが完成した、というところでまず満足しましょう。

さて、次は日本語が混じる場合の話です。ちょっぴり面倒ですが、慣れればたいしたことはありません。

まず、電子メールにおいて、日本語は JIS エンコード(iso-2022-jp)で書くのがほぼ暗黙の決まりとなっていることに注意してください。Windowsを使って普通にスクリプトを書くときは、cp932、すなわちシフトJIS系のエンコードで日本語を記しますから、素朴に日本語を本文に入れてしまうと、いくつかのメールソフトでは多分読めないものができあがってしまいます。たとえば下のようなのはダメです。

```
# coding: cp932

from email.mime.text import MIMEText
mail_body = '仏の顔もサンドバッグ'
m = MIMEText(mail_body)

...続きの処理は省略...
```

一見うまく動きますが、あとでこれを実際に送信したら、たぶん読めない人が出てくるでしょう。

なので、シフトJIS(cp932)でエンコードされている文字列を、JIS(iso-2022-jp)エンコードに変換してから使いましょう。説明はあとにして、書き方はこうです。

```
# coding: cp932

from email.mime.text import MIMEText
mail_body = '仏の顔もサンドバッグ'
mail_body = mail_body.decode('cp932').encode('iso-2022-jp')
m = MIMEText(mail_body)

...続きの処理は省略...
```

この自習テキストの中ではユニコード文字列の説明をずいぶん中途半端にやってしまったので、この書き方の真意を伝えるのは簡単でありません。まあ、decode( )の中が「元のエンコード」、encode( )の中が「変換後のエンコード」をそれぞれ表しているのだと理解してください。

これで、マナー通りの日本語電子メールになります。あ、いや、これだけでなくて、さらにMIMEText 関数を呼ぶときに、下のような引数も一緒につけるようにしてください。

```
m = MIMEText(mail_body, 'plain', 'iso-2022-jp')
```

これは、JIS エンコードを使う限り、いつもこうです。こうする理由はそれほど気にしなくてもよいですが、メッセージのヘッダ部に「これはJISエンコードだよ」という目印をちゃんとつけておくことができるようになるためのものと理解してください。

JISエンコードで書かなければいけない理由は、昔から電子メールの本文には「7ビット文字」でなくてはいけないという決まりがあったせいです。これはシステムが開発された当時の機械の事

情がそうだっただけであり、今となってはあまり意味がないはずなのですが、歴史的にこれで定着してしまっている以上、簡単に変えられないってところなんでしょうね。

- 7ビット文字ってのは、エンコードされたあとのそれぞれの数値が 0 から 127 の間におさまっているものをいいます。テキストのどこかでやったとおり、文字列ってのは 0 から 255 までの数字のカタマリとして表現できるんですが、JISエンコードってのは 0 から 127 におさまるという特徴も併せ持っているのでした。

さて、メール本文については、こういった事情で「ひと手間」かかるということなのでよろしくお願いします。

で、メールのタイトルにも日本語が使いたくなるところだけど、こっちも同じような事情かな？

すいませんが、こっちはこっちで、もうちょっとだけ小細工が必要です。メール本文は7ビット文字列なら何でもよかったですが、メールヘッダのほうはさらに制限があって、使える文字がさらに限られてくるのです。

まあ、これも、変換するやりかたを単におぼえておいて、必要になった時にコピペして使えばいいんですけどね。

やりかたはこうです。対話シェルから試しましょう。

```
>>> from email.header import Header
>>> print Header('こんにちわ'.decode('cp932'), 'iso-2022-jp')
=?iso-2022-jp?b?GyRCJDMkcyRLJEEkbxsoQg==?=
```

まず email.header モジュールから Header 関数を使えるようにインポートします。そうしたら、Header 関数を例のように書けばいいです。できあがった =?iso-2022-jp?b?GyRCJDMkcyRLJEEkbxsoQg==?= とかいう文字列が、「こんにちわ」というタイトルを表す文字列です。こうしてメールを送信すると、ちゃんとメールソフトには日本語のタイトルが出るんですよ。

ここまでを踏まえて、日本語の電子メールをつくるときの典型的な書き方を下にスクリプトとしてまとめておきましょう。

```
# coding: cp932

from email.mime.text import MIMEText
from email.header import Header

from_address = 'mail_sender@dokozo.ne.jp'
to_address = 'mail_receiver@sokozo.ad.jp'
orig_enc = 'cp932'
to_enc = 'iso-2022-jp'

mail_title = 'こんにちわ (^_^)'

message = """これを読んだら
後ろを振り向かずに
その場で両手を上げるんだ"""

message = message.decode(orig_enc).encode(to_enc)

msg = MIMEText(message, 'plain', to_enc)

msg['From'] = from_address
msg['To'] = to_address
msg['Subject'] = Header(mail_title.decode(orig_enc), to_enc)
```

```
print msg
```

ちょっと面倒ですかねえ。でも、ほとんどはお決まりの書き方なので、ぱっと見の印象ほど難しくはないんですよ。

このスクリプトの最後では、確認用に msg 変数の内容を確認することにしています。こんな結果が出てくるでしょう。

```
From nobody Wed Feb 02 15:09:36 2011
Content-Type: text/plain; charset="iso-2022-jp"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
From: mail_sender@dokozo.ne.jp
To: mail_receiver@sokozo.ad.jp
Subject: =?iso-2022-jp?b?GyRCJDMkcyRLJEEkbxsoQiAoXl9eKQ==?=

 $B$3$l$rFI$s$@$i (B
 $B8e$m$r?6$j8~$+$:$K (B
 $B$=$N>l$G<j$r>e$2$k$s$@ (B
```

いやあひどいバケかたですね。でも、MIMEの決まりに従えば、これで正しいんですよ。次はこいつを実際に送信してみましょう。

メッセージを送信する

で、次が、できあがったメッセージを実際に送信するやりかた。

これを試してみるには、使えるSMTPサーバーをひとつ知っていなくてはいけません。この情報は、サーバーの名前またはIP、そしてポート番号というものの組み合わせとして表現されているはずです。大抵の場合は接続プロバイダから教えられていると思います。ありますかー。

```
例：

  SMTPサーバー: smtp.ababaababab.ne.jp
  ポート番号  : 25

とか

  SMTPサーバー: 203.204.10.1
  ポート番号  : 25
```

とかそんな感じ。ポート番号は、通常、25だと思います。もしプロバイダが提供してるのが、よりセキュリティを高めた ESMTP とかそんなのだったら… 今はあきらめましょう。できないわけではないんですが、今回は割愛します。（この記事が全部読めるくらいだったら、適宜ググって情報源を探すと案外簡単にできるようになるかも？）

この情報がそろったら、次のような処理を書きます。

```
import smtplib

serv = smtplib.SMTP('smtp.abababababab.ne.jp', 25)
```

これで、serv 変数の中身に、郵便ポストのような機能をもつ「SMTPオブジェクト」がセットされたことになります。 この時点でもしSMTPサーバーの場所が間違っていたら、socket.error と

いう名前の例外が発生してエラーになります。

最後に、この serv 変数について、下のような命令を発行したらメール送信は完了！

```
smtp.sendmail(送信者, 宛先, メッセージ)
```

[送信者] には、送信者のメールアドレス(From)を文字列として入れます。[宛先]には、送信先のメールアドレス(To)を文字列として入れます（送信先が複数のときは、文字列のリストをここには指定します）。[メッセージ] は、さっき作ったメッセージオブジェクトを指定します。メッセージは、print したときに出てくる文字列を明示的に引き出すため、as_string() というメソッドを発行して使います。（下の例のとおり）

さっきのサンプルスクリプトを、メールの送信までしてしまう完全版に書きなおしましょう。

```
# coding: cp932

from email.mime.text import MIMEText
from email.header import Header
import smtplib

from_address = 'mail_sender@dokozo.ne.jp'
to_address = 'mail_receiver@sokozo.ad.jp'
smtp_server = 'your.smtp.server'
smtp_port = 25
orig_enc = 'cp932'
to_enc = 'iso-2022-jp'

mail_title = 'こんにちわ (^_^)'

message = """これを読んだら
後ろを振り向かずに
その場で両手を上げるんだ"""

message = message.decode(orig_enc).encode(to_enc)

msg = MIMEText(message, 'plain', to_enc)

msg['From'] = from_address
msg['To'] = to_address
msg['Subject'] = Header(mail_title.decode(orig_enc), to_enc)

serv = smtplib.SMTP(smtp_server, smtp_port)
#serv.sendmail(from_address, to_address, msg.as_string())
```

練習問題...ということのほどはありませんが、このスクリプトを、実際に動くように、自分の環境にあわせて書き換え、メールの送信ができることを確認してみるとよいでしょう。

書き換える場所は、最初にいろいろな変数に文字列を入れている場所になるでしょう。from_address, to_address, smtp_server, smtp_port のあたりを適切に書き換えれば動くと思います。自分から自分に送るのがつまらなければ、携帯のアドレスなんかに送ってみるとよいかもしれませんね。

ちなみに、最後にメールを送信する serv.sendmail の行は、サンプルではコメント化をして動作を止めています。メールの送信ってのは簡単に他人に迷惑をかけることができますので、本当に使うとき以外はこうやってコメント化しておくのが筆者のクセです。安全装置ってやつですね。

なにができるようになるかな

メールの送信のしかたは、手始めとしてはこんなものです。

一回のスクリプト実行で何回もメールの送信をしたければ、メッセージオブジェクトを作るのは何回もやりますが、SMTPオブジェクトは一回だけ作って変数に入れておき、それを何度も使いまわしてよいです。まあ、いろいろ試してみて。

これで何ができるようになったのか、ちょっと想像してみましょう。

たとえば1000人くらい会員がいる何かのグループがあって、下のようなメールを一斉に送信したいとします。

```
　○○　○○　さま
xx月分の獲得ポイントが確定しました。999,999,999 ポイントです！
よろしくね。
```

今までの色々な練習問題で、こういった文字列を作るのは簡単にできるようになっているはずです。名前とポイントは、別のファイルにシンプルなテキストファイルとして格納しているかもしれません。エクセルで管理しているかもしれませんが、これをテキストファイルに直すのは（別にpythonを使わなくても）簡単ですね。

だから、そんな仕事があったとして、pythonのスクリプトを一回だけ実行すれば、1000通だろうが2000通だろうが、データのある限り、こんな感じの定型のメールを一発で全部の会員に送れる、ということはもう可能です。今まで知ってることの「組み合わせ」次第で、だんだんと恐るべきことが可能になってきています。

楽しみましょう。

あ、添付ファイルをくっつける方法は説明しきれませんでした。まあ、BASE64とかの話題もありますし、次にまわすことにしましょう。