「Vocaloid UGC クリエーション支援システム」
実験テキスト (1): Python 入門〜バイナリ読み書き

角田孝昭（TA/シス情 CS 専攻）

2012 年 1 月 19 日

> There will now be a medium-sized intermission.
>
> *Monty Python Flying Circus*

1　プログラミング言語 Python とは

Python（ぱいそん）とは、1991 年に登場したオブジェクト指向プログラミング言語であり、インタプリタ上で動作する言語の一つである。（強い）動的型付け等を特徴とし、Perl, PHP, Ruby 等と言った言語に「近い」と言える。

Python の入門及びインストール方法は Web にある各種有用なドキュメントを参考にして頂くとして（但し、Google App Engine との互換性からバージョン 3.x や 2.7.x ではなく 2.5.x [*1] の方をインストールする事をお勧めする）[*2]、Python に触れる前に特に重要（っぽい）点のみを以下にいくつか挙げる。

- インデント **(**タブ**)** が文法的意味を持つ。例えば C 言語が {} に囲まれた部分をブロックと見做すのに対して、Python では同じ位置のインデントにある文をブロックと見做す。
- ソースコードの文字コードを各ファイルの先頭か二行目で指摘する必要がある。詳しくは「magic comment」で調べると良い。
- 配列のタイプが二種類ある。変更可能な配列をリスト型と呼び、変更不能な配列をタプル型と呼ぶ。例えば、リスト型はハッシュ（Python では「辞書」と呼ぶ）のキーになれないと言った制限がある。リスト型は角括弧 [] で囲み、タプル型は丸括弧 () で囲む。

Python 入門に関する文献案内

各自「Python 入門」等で検索すれば様々なものが出てくるが、本テキストでもいくつかお勧めしておく。

基本的な Python に関する勉強としては次のものが良いだろう:

- 初心者用 Python 講座
  `http://www.bioinfo.sfc.keio.ac.jp/class/genpro/Texts/Python1_4.htm`

[*1] Windows はここからダウンロードできる → http://www.python.org/ftp/python/2.5.4/

[*2] ちなみに、Visual Studio に慣れた人は「Python Tools for Visual Studio」を使うと、デバッグが容易になる（但し、Google App Engine に使えるかは分からない） → `http://pytools.codeplex.com/`

公式ドキュメントの物は長いが、必要な章のみを学習しても良い:

- Python チュートリアル - Python 2.7ja1 documentation
  `http://www.python.jp/doc/release/tutorial/`

「データ構造とアルゴリズム」の講義や実習で学んだ各種データ構造やアルゴリズムの C 言語での実装と比較しながら学習したい場合は、次のサイトも良い:

- M.Hiroi's Home Page / Lightweight Language
  `http://www.geocities.jp/m_hiroi/light/index.html`

2　バイナリファイルの扱い方

2.1　バイナリファイルとは？

バイナリ（binary）とは 2 進数の事である。（普通の）人間には訳のわからないバイナリの羅列が書かれているファイルをバイナリファイルと呼び、人間が読めて通常のテキストエディタで編集可能なプレーンテキストファイルと区別する。例えば、バイナリファイルである VSQ ファイルをメモ帳で読み込もうとすると次のようになる:

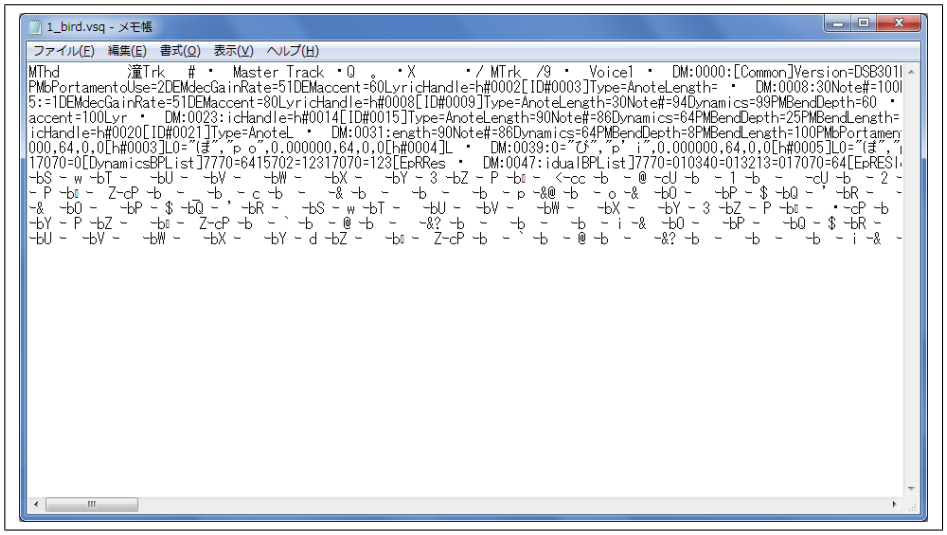

図 1　VSQ ファイルを読み込んだ例。読める部分もあるが所々文字化けしている

バイナリファイルにおいて一見文字が化けている部分は、実際にはある規則に従って書かれている。しかし、メモ帳はその規則が分からないため無理矢理テキストが書かれていると解釈して読み込んだために文字化けが発生してしまっている。これらのファイルは、今まで習ってきた方法（例えば C 言語の `fgets` 等）で読み込もうとするとメモ帳での文字化けが起こってしまうように正しく読み取れない。

これらを「コンピュータ上で記録されている数値」のままに読み込む事が可能なエディタをバイナリビューアと呼ぶ（編集可能な物はバイナリエディタと呼ぶ）。また、C 言語でも Python でもバイナリを正しく読み取れる命令が存在する。まずは、バイナリビューアの扱い方を説明しよう。

2.2 バイナリを見てみる

Windows で主に使われるバイナリビューアとしては、以下のようなソフトがある:

- Bz: シンプルで扱いやすい
  http://www.vector.co.jp/soft/win95/util/se079072.html
- Stirling: 高機能で強力
  http://www.vector.co.jp/soft/win95/util/se079072.html

また、見るだけであれば Linux コマンドの hexdump コマンドや od コマンドがある。基本的な見た目はどれも同じなので、以下は hexdump コマンドの出力を例にしながら進めていく。

さて、先程の VSQ ファイルをバイナリエディタで開くと次のような結果が得られる。

```
> hexdump -C test.vsq | head -10
00000000  4d 54 68 64 00 00 00 06  00 01 00 02 01 e0 4d 54  |MThd..........MT|
00000010  72 6b 00 00 00 23 00 ff  03 0c 4d 61 73 74 65 72  |rk...#....Master|
00000020  20 54 72 61 63 6b 00 ff  51 03 07 a1 20 00 ff 58  | Track..Q... ..X|
00000030  04 04 02 18 08 00 ff 2f  00 4d 54 72 6b 00 00 2f  |......./.MTrk../|
00000040  39 00 ff 03 06 56 6f 69  63 65 31 00 ff 01 7f 44  |9....Voice1....D|
00000050  4d 3a 30 30 30 30 3a 5b  43 6f 6d 6d 6f 6e 5d 0a  |M:0000:[Common].|
00000060  56 65 72 73 69 6f 6e 3d  44 53 42 33 30 31 0a 4e  |Version=DSB301.N|
00000070  61 6d 65 3d 56 6f 69 63  65 31 0a 43 6f 6c 6f 72  |ame=Voice1.Color|
00000080  3d 31 38 31 2c 31 36 32  2c 31 32 33 0a 44 79 6e  |=181,162,123.Dyn|
00000090  61 6d 69 63 73 4d 6f 64  65 3d 31 0a 50 6c 61 79  |amicsMode=1.Play|
```

これの見方は次の通りである。

左端　ファイルの位置（16 進数）。
中央　ファイルに書かれている数値（16 進数）。左から右へ 16 個（16 進数で 10 個）の数値が並ぶ。
右端　書かれている内容を文字として解釈するとどのようになるかが書かれている。

では、実際に VSQ ファイルのフォーマット解析結果と照らし合わせて、このファイルの情報を上のバイナリ列から読み取ってみよう。最初のブロックは次の通りである（16 進コードが書かれている物は固定、XX の部分はファイルによって変わる）。

表 1　VSQ ファイルのヘッダチャンク

16 進コード	長さ	意味
4D 54 68 64	4	ヘッダチャンクを表す (MThd)(固定)
00 00 00 06	4	ヘッダチャンクのデータ長 (固定)
00 01	2	フォーマット 1 を表す (固定)
00 XX	2	XX はトラック数を表す。2〜17 の値を取る。
XX XX	2	XX XX は 4 分音符の分解能を示す。

まず、ファイルの最初を見てみると、確かに「4D 54 68 64」「00 00 00 06」「00 01」の順で始まっている事が分かる。また、次の「00 02」を見れば、トラック数が 2 であること、「01 e0」を見れば 4 分音符の分解能 [*3] は $(01e0)_{16} = (480)_{10}$ である事が分かる。

実際には開発中に必ずしもバイナリ列を見る必要は無いが、うまく読めなかった際にファイルがどのようになっているかを実際に見ることでデバッグの助けになる事が多い。

2.3 Python でバイナリファイルを読み込む

では、実際に Python でバイナリファイルを読み込むプログラムを例に示す。が、ここで VSQ ファイルを読み込むコードを示してしまっては実験にならないので、画像ファイル（ビットマップファイル）の情報を読み込む例を示す。

表 2　ビットマップファイルのヘッダ（最初のいくつかのみ）

16 進コード	長さ	意味
42 4D	2	ファイルタイプ
XX XX XX XX	4	ファイルサイズ [byte]
00 00	2	予約領域 1
00 00	2	予約領域 2
XX XX XX XX	4	ファイル先頭から画像データまでのオフセット [byte]
XX XX XX XX	4	情報ヘッダサイズ [byte]
XX XX XX XX	4	画像の幅 [ピクセル]
XX XX XX XX	4	画像の高さ [ピクセル]

これらの情報を読み込む Python プログラムの例を示す。

ソースコード 1　ビットマップファイルの読み込み

```
1  #!/bin/env python
2  #-*- coding: utf-8 -*-
3  from struct import *;   # pack 命令に必要
4
5  fp = open("test.bmp","rb");
6
7  # 以下は全部文字列として読み込まれる
8  # read の引数はバイト数を表す
9  type = fp.read(2);
10 fsize = fp.read(4);
11 reserved1 = fp.read(2);
12 reserved2 = fp.read(2);
13 offbits = fp.read(4);
14 isize = fp.read(4);
15 width = fp.read(4);
16 height = fp.read(4);
17 fp.close();
18
19 # 整数に変換するには unpack("I",...) 等を使う ("I" は 4バイトの整数を示す)
20 # 変換後はタプル型になるので最初の要素を取り出すために [0] を付ける
21 print("ファイルサイズ:␣%d" % (unpack("I",fsize)[0]));
22 print("画像サイズ:␣%d␣x␣%d" % (unpack("I",width)[0], unpack("I",height)[0]));
```

[*3] パラメータを調整できる最小の単位時間だと思えば良い。

```
> python bmp.py
ファイルサイズ: 175734
画像サイズ: 244 x 240
```

特に重要なのは、open 命令（C 言語の fopen にあたる）の 2 つめの引数で「b」を併せて指定する点と、unpack によりバイナリから整数に変換する点である。このプログラムでは各データを 1 つ 1 つ読み込んでいるが、一度に一つの文字列に読み込ませた後、unpack で一つの文字列を各データを一気に分割する事もできる。

ソースコード 2　ビットマップファイルの読み込み (改)

```
#!/bin/env python
#-*- coding: utf-8 -*-
from struct import *;   # pack 命令に必要

fp = open("test.bmp","rb");
header = fp.read(26);

# 配列に展開
# <: リトルエンディアンである事を示す (通常これ)
# 2s: 長さ 2の文字列
# H: 2バイトの整数
# I: 4バイトの整数
info = unpack("<2sIHHIIII",header);
print(info);
```

```
> python bmp.py
('BM', 175734, 0, 0, 54, 40, 244, 240)
```

バイナリファイルの扱いに関する文献案内

ファイル読み書きのメソッドに関しての公式ドキュメント:

- 7. 入力と出力 - Python 2.7ja1 documentation
  http://www.python.jp/doc/release/tutorial/inputoutput.html#tut-files

謎に満ちたメソッド unpack の説明はこちら:

- 4.3 struct – 文字列データをパックされたバイナリデータとして解釈する
  http://www.python.jp/doc/2.5/lib/module-struct.html

3　課題

※提出の必要はありませんが参考までに。

1. Python 及びバイナリエディタのインストールを行う。
2. VSQ ファイルのヘッダチャンク及びマスタートラックチャンクをバイナリエディタで確認する。
3. Python で VSQ ファイルのヘッダチャンクを各データ毎に逐次読み込む。
4. VSQ ファイルのヘッダチャンク及びトラックチャンクの先頭部分をそれぞれ変換出来る pack テンプレート文字列を考えて、Python で読み込んでみる。