TSNET スクリプト通信

TSC 編集委員会

ISSN 1884-2798

目次

表紙写真: 分かれ道
撮影: jscripter
日時: 2011 年 4 月 10 日
場所: 広島市黄金山頂上より広島呉道路を望む。
メモ: 複雑な立体交差は現在の世界の迷いを象徴しているように感じた。

巻頭言

jscripter

東日本大震災の翌月、TSNET スクリプト通信第 12 号を刊行します。被災者の方々には心よりお見舞い申し上げます。

3 月末に機械伯爵さんから、「Python の文法」の第 4 回の原稿をいただいた。そして、Y ささんから「プラス的・エフェクト」の原稿を 4 月 15 日にいただいた。「よしおさんとロボ太」は海鳥さんから既にいただいているので、後は僕が原稿を書けば出せる状態になっていた。

プレッシャーを感じつつ、スクリプトを整備して、今日半日で記事にまとめたが、Perl の文法を説明しているわけではない。スクリプトがどのように動くのかを多少説明しているだけだ。スクリプトを読んで学習してほしい。前号の記事の参照は必須であると、記事の後書きに書くべきことをここに書いている。編集者の特典かもしれない。

次号で 3 周年、4 年目のサイクルに入る。世の中は混沌としてきた。アフリカも中東もヨーロッパも米国も中国も、そして日本も。さて、我々はどう生きていくのだろうか。TSNET スクリプト通信の運命や如何に・・・次回分に海鳥さんの「よしおさんとロボ太」の最後の 2 回を残しておいたので、まだ続けることはできるだろう。

カズオ・イシグロの特集の録画を先ほど torne で見ながら原稿を書いていた。5 歳まで日本にいて、イギリスに渡った。成長するにつれ、日本の記憶が英国の生活によって作られたものであることに気がつく。最初はミュージシャン志望だったのだそうだが、すべての記憶が失われないうちに書き留めておきたいとの焦りの気持ちで小説家になった。我々スクリプターはなぜスクリプトを書くのだろうか。その理由と意義についても考えていきたいものである。

（投稿: 2011 年 4 月 23 日）

Python の文法

TSNET スクリプト通信版　草稿　第４回

6-2 組み込みオブジェクト
6-2-1 組み込みオブジェクトとは？

Python では、システムを立ち上げた時点で、豊富な組み込みオブジェクト(built-in object)が使用可能になっています。
組み込みオブジェクト名の一覧は、dir 関数を用いてインタープリタ画面で

```
>>> dir(__builtins__)
```

とすると、識別子に登録してあるものは全て表示されます。
組み込みオブジェクトには、以下のようなものがあります。

■組み込み型(built-in type or class)
Python では便利なデータ型が、何の準備もせずにいきなり使えます。

※旧式の Python では『型(type)』と『クラス(class)』の概念は微妙に異なったのですが、現在では全く同じものになりました。即ち、組み込み型は全て組み込みクラスです。

組み込み型には、クラスオブジェクトとして提供されているもの、リテラルで書くもの、定義で書くものなどがあります。
組み込みのクラスオブジェクトとして提供されているもののうち、直接ユーザがデータとして使用するものには、以下のようなものがあります。

＜原子オブジェクト型＞
- int（整数型）
- bool（真偽型）
- float（浮動小数点数型）
- complex（複素数型）

＜コレクション型＞
＜不変コレクション＞
＜順序列＞
- tuple（タプル型）
- str（文字列型）
- bytes（バイト列）

＜集合＞
- frozenset（凍結集合型）

＜可変コレクション＞
＜順序列＞
- list（リスト型）
- bytearray（バイト配列）

＜写像＞
・dict（辞書型）

＜集合＞
・set（集合型）

以上の組み込み型については、「組み込み型」の項で説明します。
上記以外にも、始原オブジェクト型(object)、クラス型（＝メタクラス type）、定義とリテラルでのみ提供される関数型(function)、コンパイル済みコードオブジェクト型、モジュール型、ファイル型、例外型(例外クラス)、修飾子型、各種反復子型など、間接的にオブジェクトとして使用されるものを含めるとかなりの数になります。
これらについては、「その他の組み込みオブジェクト」の項で説明します。

■組み込み関数(built-in function or method)

組み込みオブジェクトの残りの大半は、『組み込み関数』と呼ばれるものです。

組み込み関数(<class 'builtin_function_or_method'>)は、関数(<class 'function'>)と働きは全く同じですが、違うクラスとして設定されています。関数のサブクラスでもありません。

ユーザーが使用することを想定されている組み込み関数は以下の通りです。

・abs（絶対値）
・all（全数真判定）
・any（一部真判定）
・ascii（アスキー表示）
・bin（二進法表示）
・chr（数値⇒文字）
・compile（コードコンパイル）
・delattr（属性消去）
・dir（検査）
・divmod（乗算・剰余）
・eval（評価）
・exec（実行）
・format（書式）
・getattr（属性取得）
・globals（グローバル辞書取得）
・hasattr（属性調査）
・hash（ハッシュ）
・hex（十六進法表示）
・id（オブジェクトＩＤ表示）
・input（標準入力取得）
・isinstance（クラス検査）
・issubclass（派生検査）
・iter（反復子生成）
・len（アイテム数）
・locals（ローカル辞書取得）
・max（最大値）
・min（最小値）

・next（反復子を先に進め、アイテムを取得）
・oct（八進法表示）
・open（ファイルオープン）
・ord（文字コード番号取得）
・pow（累乗）
・print（標準出力）
・repr（表現文字列取得）
・round（四捨五入）
・setattr（属性代入）
・sorted（整列済みリスト取得）
・sum（合計値）
・vars（ローカル辞書あるいは属性取得）

以下の二つは、構文に対して基本的に水面下で呼び出されるため、ユーザが直接使用する必要はありません。

・__build_class__（クラスを作る隠し関数）⇒'class'定義構文
・__import__（インポートする隠し関数）⇒'import'導入構文

■予約語（キーワード）として登録してあるオブジェクト

名前空間の識別子とは別個に、キーワードとして登録してあるオブジェクトが三つあります。

・False（bool クラスの偽値）
・None（否定オブジェクト）
・True（bool クラスの真値）

■その他

インタープリタを立ち上げた際、'exit'と'quit'という'site.Quitter'オブジェクトが登録されます。

この二つのオブジェクトは、関数呼び出しすることで、インタープリタを終了させることができます。

6-2-2 組み込み型(built-in type)

6-2-2-1 原子オブジェクト型(atomic object types)

原子オブジェクト型は全て数値型で、抽象基底クラス(ABCs = Abstract Base Classes)の'numbers.Number'(numbers モジュールの Number クラス)を基底クラスとして持っています。

```
>>> import numbers
>>> issubclass(int, numbers.Number)
True
>>>
```

数値クラスには、以下のようなメソッドや属性が共通して実装されています（オブジェクト全てに共通するものを除く）

●演算子に対応した特殊メソッド
・加算　　A + B　　__add__(A, B), __radd__(B, A)
・減算　　A - B　　__sub__(A, B), __rsub__(B, A)
・乗算　　A * B　　__mul__(A, B), __rmul__(B, A)
・実除算　　A / B　　__truediv__(A, B), __rtruediv__(B, A)
・フロー除算
　　A // B　　__floordiv__(A, B), __rfloordiv__(B, A)
・剰余算　　A % B　　__mod__(A, B), __rmod__(B, A)
・負　　-X　　__neg__(X)
・正　　+X　　__pos__(X)
・累乗算　　A ** B __pow__(A, B), __rpow__(B, A)

●関数に対応した特殊メソッド
・絶対値　　abs(X)　　__abs__
・商と剰余　　divmod(A, B)　__divmod__(A, B), __rdivmod__(B, A)

●クラスコンストラクタ引数に対応した特殊メソッド
・真偽化　　bool(X)　　__bool__(X)
・整数化　　int(X)　　__int__(X)
・実数化　　float(X)　　__float__(X)

●その他のメソッド
・共役複素数　num.conjugate()
・タプル変数化　　num.__getnewargs__()⇒(num,)

●属性
・虚数部　　num.imag
・実数部　　num.real

以下、各々のクラスについては、上記以外の組み込みメソッドや属性について、説明します。

6-2-2-1-1 整数型(int)

整数は電子演算に於いてもっとも基本的なデータです。
その為、ビット演算などハードウェア寄りの演算なども取り扱えるようになっています。

ただし、Pythonの整数型(int型)のオブジェクトは、ハードウェアのメモリ限界以外に限界を持たない上限下限無しの巨大な整数値を扱えますので、科学演算にも十分耐えられます。

整数型オブジェクトには、原子オブジェクト型に共通する以外に、次のようなメソッドや属性が実装されています（これ以外のものは「原子オブジェクト型」を参照してください）

［以下、X, Y は全て int 型］
●演算子対応メソッド
・ビット論理積 X & Y X.__and__(Y), Y.__rand__(X)
・ビット論理和 X | Y X.__or__(Y), Y.__ror__(X)
・ビット排他的論理和
X ^ Y X.__xor__(Y), Y.__rxor__(X)
・ビット左シフト X << Y X.__lshift__(Y), Y.__rlshift__(X)
・ビット右シフト X >> Y X.__rshift__(Y), Y.__rrshift__(X)
・ビット反転 ~X X.__invert__()

●組み込み関数、及び組み込みモジュールの関数に対応したメソッド
・上限値関数'math.ceil' ⇒ X.__ceil__()
・下限値関数'math.floor' ⇒ X.__floor__()
・切捨て関数'math.trunc' ⇒ X.__trunc__()
・四捨五入値関数'round' ⇒ X.__round__()

上の三つ(__ceil__, __floor__, __trunc__)は、'math.ceil', 'math.floor', 'math.trunc'に対応するためのものです。
しかし、これらの'math'モジュールの関数群は、 基本的に実数(=浮動小数点数 float)や分数(fractions.Fraction)の整数値概数を求めるものなので、整数値に適用するとそのまま整数を返します（ただし、float 型オブジェクトには、上記の三つのメソッドは実装されていません）

本来、浮動小数点数や分数を想定した値に整数値がイレギュラーで入った時のためのメソッドと考えられます。

四捨五入の'round'組み込み関数も基本的に似たようなものなのですが、これは桁数を指定できますので、整数でも適用できます（指定しなければ小数点以下を四捨五入しますので

```
>>> round(987654321, -2)
987654300
>>> (987654321).__round__(-2)
987654300
>>>
```

●特殊メソッド

■順序列のインデックス値
書式：X.__index__() ⇒ X
動作：順序列のインデックスとして利用可能になるためのメソッド。整数型ではそのままの値を返す（基本的には真偽型を整数値に変換する為に使用されるものと考えられるので、整数型では無意味）

■ビット長(bit length)を返すメソッド
書式：X.bit_length() ⇒ i
動作：整数"X"のビット長を示す整数"i"を返す

●属性
・分母を収納 X.denominator
・分子を収納 X.numerator

この二つの属性は基本的に分数値として使用する場合に使います。整数の分母(denominator)は当然1、分子はその整数そのものです。

6-2-2-1-2 真偽型(bool)

真偽(bool)型は、基本的には真（True）と偽（False）の２値しか持たないオブジェクトです。
その実体はほぼ整数で、真を1、偽を0として、数値として利用できます。
整数型と真偽型のメソッドや属性は完全に重なっています。
初期のPythonは真偽値の代わりに整数の0や1を用いていたので、真偽値は真偽判定の結果をわかりやすくするために整数にかぶせたクラスのラッパーのようなものだと思ってください。
本質的に整数値であるため、以下のような演算が可能です。

【bool型値を整数の１と０として計算する】

```
>>> t = True
>>> f = False
>>> t * 10
10
>>> t + t
2
>>> 10 * t + t
11
>>> 10 / (t + t)
5.0
>>> 500 * f
0
>>> 10 > 1   #【判定式の結果はbool】
True
>>> (10 > 1) * 555
555
>>> yesno = lambda q: input(q)[0] in ('y','Y')
>>> yesno("18歳以上ですか？:")
18歳以上ですか？:yes
True
>>> #【複数のチェックを一つの変数に収納する】
... check18 = 0b100
>>> checks = yesno("18歳以上ですか？:") * check18
18歳以上ですか？:yes
>>> print("OK" if checks & check18 else "No")
OK
>>>
>>> checks
4
>>> bin(checks)
'0b100'
>>>
```

6-2-2-1-3 浮動小数点数型(float)

浮動小数点数型は、実数を限られた情報単位の概数で表すオブジェクトです。

整数型は基本的にメモリが許す限り青天井ですが、浮動小数点数は、Pythonが実装されたハードウェアに依存します。

■整数であるかどうか
書式：f.is_integer() ⇒ b
動作："f"が整数ならば"b"は'True'、整数でないなら'False'を戻す

■分数表現
書式：f.as_integer_ratio() ⇒ t
動作："f"を整数による分数表現(タプル型の"t")として出力する(注意：ハードウェアの二進法表現の性質上、分母が二の乗数であるもの以外はあまり正確には出力されない)

```
>>> f = 100.0
>>> f.is_integer()
True
>>> f = 0.5
>>> f.is_integer()
False
>>> f.as_integer_ratio()
(1, 2)
>>>
```

■16進法変換
書式：f.hex() ⇒ hxstr
動作："f"を十六進法表現の文字列"hxstr"として戻す

■16進法逆変換
書式：float.fromhex(hxstr) ⇒ f
動作：十六進法表現の文字列"hxstr"を浮動小数点数"f"にして戻す

```
>>> f = 5.5
>>> f.hex()
'0x1.6000000000000p+2'
>>> float.fromhex('0x1.6p+2')
5.5
>>>
```

●組み込み関数、及び組み込みモジュールの関数に対応したメソッド

・四捨五入値関数'round' ⇒ X.__round__()
・上限値関数'math.ceil' ⇒ X.__ceil__()
・下限値関数'math.floor' ⇒ X.__floor__()
・切捨て関数'math.trunc' ⇒ X.__trunc__()

'math'モジュールの三つの関数は、整数値に適用するとそのまま整数を返します。これは基本的に分数(fractions.Fraction)や実数(=浮動小数点数 float)の整数値概数を求めるもので、'math.ceil'は「その数値を下回らない最小の整数」を、'math.floor'は「その数値を上回らない最大の整数」を、'math.trunc'は「切捨て値」を求めます。正の数では'math.floor'と'math.trunc'は同じ数値になりますが、負の数になると処理が異なります。

```
>>> import math
>>> x, y = 5.5, -5.5
>>> math.ceil(x)
6
>>> math.ceil(y)
-5
>>> math.floor(x)
5
>>> math.floor(y)
-6
>>> math.trunc(x)
5
>>> math.trunc(y)
-5
```

●使用を奨励されないメソッド

__getformat__及び__setformat__
※浮動小数点型オブジェクトの内部表現に関係するメソッドなので、通常使用しません

6-2-2-1-4 複素数型(complex)

複素数型のメソッドと属性は、原子オブジェクト型に共通なもののみです。
ただし、比較演算子を多重定義する'__ge__','__gt__','__le__','__lt__'は事実上使用できないようになっています。

複素数は、数直線上に無いので、大小関係が比較できません。

6-2-2-1-5 メソッド・プロパティ　一覧

整数型、真偽型、浮動小数点数型、複素数型のメソッドと属性の一覧を下に挙げます。

・複素数(complex)型のメソッドと属性＝全原子オブジェクトに共通
__abs__, __add__, __bool__, __class__, __delattr__, __divmod__
__doc__, __eq__, __float__, __floordiv__, __format__, __ge__
__getattribute__, __getnewargs__, __gt__, __hash__, __init__
__int__, __le__, __lt__, __mod__, __mul__, __ne__, __neg__
__new__, __pos__, __pow__, __radd__, __rdivmod__, __reduce__
__reduce_ex__, __repr__, __rfloordiv__, __rmod__, __rmul__,
__rpow__, __rsub__, __rtruediv__, __setattr__, __sizeof__
__str__, __sub__, __subclasshook__, __truediv__, conjugate
imag, real

・整数(int)型＝真偽(bool)型と全て共通（複素数型と共通なものを除く）
__and__, __ceil__, __floor__, __index__, __invert__, __lshift__
__or__, __rand__, __rlshift__, __ror__, __round__, __rrshift__
__rshift__, __rxor__, __trunc__, __xor__, bit_length, denominator
numerator

・浮動小数点数(float)型（複素数型と共通なものを除く。整数型とは重複あり）
__getformat__, __round__, __setformat__, __trunc__, as_integer_ratio
fromhex, hex, is_integer

6-2-2-2 コレクション型(collection object type)

コレクション型は、複数のオブジェクトを収納するオブジェクトという外見を持っています。
しかしほとんどのコレクション型オブジェクトは、実際には「収納している」オブジェクトに対するポインタを保持しているだけです。
このため、コレクション型オブジェクトは、収納するオブジェクトの種類を問わない（収納しているオブジェクトのメモリ領域を確保しているわけではないので）ものがほとんどです。

新しく追加されたコレクションオブジェクトの中には、メモリ節約及び高速アクセスを目的とするため、収納するオブジェクトの内容が特定されているものもあります。また文字列オブジェクトは、昔から（Python3 でリニューアルされた後も）文字コード情報のみを限定して収納するコレクションオブジェクトです。

コレクションオブジェクトには、その収納内容を変更できない不変オブジェクトと、収納内容が変更可能な可変オブジェクトとが存在します。

また、インデックスとして一定範囲の整数のみを扱う順序列(sequence)オブジェクトと、不変オブジェクトであればなんでもキーに使用可能な写像(mapping)オブジェクト、そしてキーやインデックスを持たず、集団への所属のみを表すための集合オブジェクトがあります。

その他、標準モジュールとして、'array'モジュールには配列('array')クラスが、'collections'モジュールには、その他の多数の追加コレクション型が用意されています。

6-2-2-2-1 可変／不変(mutable / immutable)

コレクション型には、収納するアイテムを変更できる可変コレクションと、収納アイテムが一切変更不可能な不変コレクションがあります。単純なオブジェクトの順序列であれば、リスト('list')は可変コレクションであり、タプル('tuple')は不変コレクションです。

可変コレクションは、アイテム数やオブジェクトの内容や種類に至るまで全て変更可能ですが、不変コレクションでは、登録されているオブジェクトは一切変更不可能です。

Python はデフォルトではＣ言語のような（アイテムオブジェクトの種類と数は固定で内容は変化できる）配列のようなコレクションは用意していませんが、標準モジュールの'array'モジュールには、数値限定ですが、オブジェクトの種類と個数を指定するコレクション'array'があります。

```
>>> import array
>>> a = array.array('I',(0,1,2,3)) # 'I'は符号無しの整数値のみを指定
>>> a
array('I', [0, 1, 2, 3])
>>> a[3] = 30
>>> a
array('I', [0, 1, 2, 30])
>>>
```

「変更不可能」の意味を詳しく説明すると「登録オブジェクト自体が変更不可能」なのであって、登録されているオブジェクトの内容……例えば属性やメソッド、あるいはコレクションであればアイテムなどは変更可能です（リストのタプルであれば、リストの内容は変更可能というわけです）

```
>>> imc = ([0], [1], [2])
>>> imc
([0], [1], [2])
>>> imc[0]=[100] # タプルのアイテムは変更不可能
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> imc[0][0] = 100 # タプルに登録されているリストのアイテムは変更可能
>>> imc
([100], [1], [2])
>>>
```

可変タイプと不変タイプの違いは、アクセス速度とメモリ消費量にありますが、繰り返しを多用するような場合を除き、資源節約的なメリットはあまりありません。

不変タイプは、不用意に登録アイテムを変更しないよう、ユーザ（プログラマ自身を含む）に注意を促すようなモジュールなどで使用すると効果があります。

6-2-2-2-2 コレクションのアイテムへのアクセス

順序列や写像などは、添え字(subscript)によってアイテムにアクセス／変更／追加できます。

```
>>> a = [0,1,2,3,4,5]
>>> a
[0, 1, 2, 3, 4, 5]
>>> a[3] # アイテムにアクセス
3
>>> b = {'x':100,'y':200}
>>> b
{'y': 200, 'x': 100}
>>> b['x']
100
>>> a[2] = 200 # アイテムを変更
>>> a
[0, 1, 200, 3, 4, 5]
>>> b['y']=2000
>>> b
{'y': 2000, 'x': 100}
>>> del a[4] # アイテム（リンク）を消去
>>> a
[0, 1, 200, 3, 5]
>>> del b['x']
>>> b
{'y': 2000}
>>>
```

「添え字」という言葉は、本来文字の右下(あるいは右上）に小さく添えられた文字を指します。しかし、フォントが変更できないタイプライターなどで、ブラケット('[]')で囲むことで添え字を表した慣習が残り、これがプログラミングコードのテキストの書き方まで残ったため、ブランケットで囲んだ数字(や文字)を添え字と呼んでいるようです

Python では'del'文を用いても消去されるのはラベルだけですので、'del'で消去しても可変／不変問わずコレクションに登録されているオブジェクトへのアクセスは可能です。

```
>>> a = object()
>>> a
<object object at 0x00BFD150>
>>> x = [a]
>>> x[0]
<object object at 0x00BFD150>
>>> del a # "a"というラベルを消去
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> x[0] # "x[0]"というラベルは生きている
<object object at 0x00BFD150>
>>>
```

アイテムへのアクセス／変更／消去は、'__getitem__','__setitem__','__delitem__'の三つのメソッドでフックしています。

自作でカスタムコレクションオブジェクトを作成する場合は、この三つのメソッドを実装すれば添え字でのアクセスの振る舞いをエミュレートすることができます。
コレクションは、'for'キーワードによる連続アクセス('for'文や生成子リテラル／内包表記)にも対応しています。

```
>>> a = [0, 1, 2, 3]
>>> for x in a: print(x)
...
0
1
2
3
>>>
```

'for'キーワードによる連続アクセスは、'__iter__'メソッドで反復子を返すことによって実現しています。
コレクションの'for'キーワードに対する振る舞いをエミュレートする場合は、'__iter__'メソッドを実装してください。

6-2-2-2-3 順序列型(sequence types)

順序列型(sequence types)は、アイテムに割り当てられたインデックスによりアクセス可能なコレクション型です。
インデックスは０から連続する整数で、'len'関数で確認できるアイテム数はインデックスの最大値＋１となります。
なお、アイテム数が不明なリスト"x"の最後尾のアイテムを指定する場合は"x[-1]"を使用します（以下、後ろから２番目なら"x[-2]"のように、マイナスで指定できます。

```
>>> x = [0, 10, 20, 30]
>>> len(x)
4
>>> x[3]
30
>>> x[-1]
30
>>>
```

可変の順序列型で、更新によって中のアイテムが抜けた場合、インデックスは連続する形に補正されます。

```
>>> x
[0, 10, 20, 30]
>>> x[2]
20
>>> del x[2]
>>> x
[0, 10, 30]
>>> x[2]
30
>>>
```

組み込みの順序列型コレクションには、以下のようなものがあります。

・list（リスト型）：可変型の汎用オブジェクト収納用
・tuple（タプル型）：不変型の汎用オブジェクト収納用
・str（文字列型）：不変型の文字オブジェクト収納用
・bytes（バイト列）：不変型のバイトデータオブジェクト収納用
・bytearray（バイト配列）：可変型のバイトデータオブジェクト収納用

また、組み込みの順序列型コレクションには、以下のようなメソッドが共通して実装されています（オブジェクト全てに共通するものを除く）

●通常のメソッド

■アイテムをカウントする
書式：s.count(itm) ⇒ n
動作：順序列型"s"を検索して"itm"と同じアイテムが何個あるかカウントし、個数(整数"n")を戻す（無ければ0を戻す）

■アイテムが最初に現れるインデックスの検索
書式：s.index(itm) ⇒ n
動作：順序列型"s"のアイテムを検索して、"itm"と同じアイテムが最初に現れるインデックス(整数"n")を戻す（無ければ'ValueError'を戻す）

●演算子や関数呼び出しをフックするメソッド

■アイテムの連結
書式：s.__add__(exS) ⇒ newS
動作："s + exS"で呼び出され、順序列型を連結した新しいコレクション"newS"を戻す。ただし"s"と"exS"は同じ型の順序列型(サブクラスを含む)でなければならない。

```
>>> s = [1, 2, 3]
>>> exS = [4, 5, 6]
>>> s + exS
[1, 2, 3, 4, 5, 6]
>>> s.__add__(exS)
[1, 2, 3, 4, 5, 6]
>>>
```

■アイテムの反復
書式：s.__mul__(i) ⇒ newS
書式：s.__rmul__(i) ⇒ newS
動作：順序列型"s"を整数"i"回数繰り返した新しい順序列型"newS"を戻す。上段の式は"s * i"で、下段の式は"i * s"で呼び出される（"i"は0や負の数でも構わないが、どちらも空の順序列型が戻る）

```
>>> s
[1, 2, 3]
>>> s * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> s.__mul__(3)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> 3 * s
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> s.__rmul__(3)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>>
```

■メンバーテスト
書式：s.__contains__(itm)
動作："itm in s"で呼び出され、メンバーテスト行う。"s"のアイテムに"itm"があれば'True'を、無ければ'False'を戻す。

```
>>> s = [0,1,2]
>>> 1 in s
True
>>> s.__contains__(1)
True
>>>
```

■インデックス取得
書式：s.__getitem__(i)
動作："s[i]"で呼び出され、"i"で指定されたインデックスをつけられたアイテムを戻す。

```
>>> s = [10,20,30]
>>> s[1]
20
>>> s.__getitem__(1)
20
>>>
```

■反復子取得
書式：s.__iter__() ⇒ itr
動作："iter(s)"で呼び出され、反復子"itr"を戻す。また、"for"キーワードによっても自動的に呼ばれて、全てのアイテムを検索する

■アイテム数取得
書式：s.__len__() ⇒ i
動作："len(s)"で呼び出され、アイテム数を整数型"i"を戻す

```
>>> s = [0,1,2,3,4]
>>> iter(s)
<list_iterator object at 0x00C776B0>
>>> s.__iter__()
<list_iterator object at 0x00C77810>
>>> len(s)
5
>>> s.__len__()
5
>>>
```

6-2-2-2-3-1 汎用の順序列型

汎用の順序型としてPythonは、可変順序型のリスト(list)と不変順序型のタプル(tuple)を用意しています。
どちらもカッコとアイテムを区切るコンマで書ける簡単なリテラルをもっています（タプルは通常、カッコすら要らない）ので、Pythonでの使用率はダントツに高くなります。
基本的にユーザがコンテナオブジェクトと意識して使用する場合、リストを使うのが普通でしょう。
タプルはどちらかといえば、文法的に組み込まれた部分で使用する頻度が多くなると思われます。

メモリ消費を意識してリストではなくタプルを使用することは、あまり意味は無いでしょう。リストクラスの本体はＣで書かれていますし、Python の製作者サイドで何度も最適化されています。あえてリストではなくタプルを使用するなら『不変』リストであることが有効である、といったコーディング上の積極的な理由で考えるべきです。

6-2-2-2-3-1-1 タプル(tuple)

■クラス名：'tuple'

タプルは不変のリストです。
タプルは主に文法の一部として使用されます。
一般のオブジェクトのコンテナとしては、後述のリストのほうが便利ですが、座標や IP アドレスのような「特定の値の組」を表現する際は、不変のタプルのほうが便利である場合もあります。
また、不変オブジェクトであるため、辞書オブジェクトのキーとしても利用できます。
タプルは不変ですが、'+'演算子を用いて連結したり、'*'演算子を用いて繰り返しを作ることができます（新しいタプルが作られるだけなので、元のオブジェクトとしてのタプルには変化がありません）
リテラルのところでも説明しましたが、タプルの書式は','(コンマ)によって作られるので、基本的に（空タプル以外は）カッコは必要たりません。
しかし、演算子としてのコンマの優先度は最低なので、内包表記や関数の中で使用する場合は、大体カッコが必要になります。

```
>>> T = 0,1,2
>>> T + (3,) #【このカッコは必要】
(0, 1, 2, 3)
>>> T
(0, 1, 2)
>>> T += 3,  #【こちらは不要】
>>> T
(0, 1, 2, 3)
>>> T * 2
(0, 1, 2, 3, 0, 1, 2, 3)
>>> T
(0, 1, 2, 3)
>>> T *= 2
>>> T
(0, 1, 2, 3, 0, 1, 2, 3)
```

タプルは、全ての順序列型のコレクションに共通なメソッド以外は持っていません（コンテナの内容を操作できないので、中身を変化させるためのメソッドが一切ありません）

6-2-2-2-3-1-2 リスト(list)

■クラス名：'list'

リストは、最も高頻度でコンテナとして使用されるコレクションオブジェクトでしょう。
リストは内容も大きさも可変なので、制約を考えることなく使用できます。
また、手軽な『参照渡し』として利用することもできます。
リストは、タプルが持つ全てのメソッドに加え、アイテムを操作する便利なメソッドがいくつか揃えてあります。

■リストアイテム削除(delete an item)
書式：L.__delitem__(i) ⇒ None
動作：リストアイテム削除をフックする。上記書式は"del L[i]"と同じ意味。"i"はインデックスとなる整数を指定する（範囲になければエラーとなる）

■リストの連結と更新(insert additional list)
書式：L.__iadd__(nL) ⇒ L
動作：リスト"nL"を"L"に追加する。動作的には"L += nL"と同じだが、更新された"L"が戻される。

■アイテムの反復と更新(insert multiplication list)
書式：L.__imul__(i) ⇒ L
動作：整数"i"で指定された回数だけLを反復して更新。動作的には"L *= nL"と同じだが、更新された"L"が戻される

```
>>> L = [0, 1, 2, 3]
>>> L.__delitems__(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute '__delitems__'
>>> L.__delitem__(1)
>>> L
[0, 2, 3]
>>> L.__iadd__([4, 5])
[0, 2, 3, 4, 5]
>>> L
[0, 2, 3, 4, 5]
>>> L.__imul__(2)
[0, 2, 3, 4, 5, 0, 2, 3, 4, 5]
>>> L
[0, 2, 3, 4, 5, 0, 2, 3, 4, 5]
>>>
```

■逆転リスト反復子を作成する(get reversed list)
書式：L.__reversed__() ⇒ Io
動作："reversed(L)"の動作をフックする。Lの順序を逆順で生成する反復子を作成。"Io"は'list_reverseiterator object'という特殊なオブジェクト

注意：リストクラスを継承して"__reversed__'メソッドを上書きすると、'reversed'関数は上書きされた'__reversed__'メソッドの動作を参照しますので、動作をフックしていることは確実です。しかし'__reversed__'メソッドを持たないタプルや文字列などの順序列型コレクションでも'reversed'関数は動作します。この場合、内部で一度リストに変換しているものと考えられます。

■リストアイテムの追加(append item)
書式：L.append(o) ⇒ None
動作：リストアイテムを追加する。"o"は任意のオブジェクトで、リストの末尾に追加される

■リストアイテムの取り出し(pop item)
書式：L.pop(i) ⇒ L[i]
動作："i"で指定されたインデックスのアイテムを取り出して返す（リストは変更される）。"i"を省略すると、末尾のアイテムを取り出す。引数なしの'pop'メソッドは、'append'メソッドとペアでLIFO式のスタックをエミュレートする時などに使用される

```
>>> L
[0, 1, 2, 3, 4]
>>> rlo = L.__reversed__()
>>> rlo
<list_reverseiterator object at 0x00C77810>
>>> list(rlo)
[4, 3, 2, 1, 0]
>>> L
[0, 1, 2, 3, 4]
>>> L.append(5)
>>> L
[0, 1, 2, 3, 4, 5]
>>> L.pop()
5
>>> L
[0, 1, 2, 3, 4]
>>>
```

■リストアイテムの延長(extend list)
書式：L.extend(eL) ⇒ None
動作："L"に"eL"を連結する。"L += eL"に等しい

■リストアイテムの挿入
書式：L.insert(i, o) ⇒ None
動作：整数"i"で指定された位置にオブジェクト"o"を挿入する（入れ替えではないので、以降のインデックスは一つずつ後ろにズレる

■リストアイテムを（アイテムを指定して）削除
書式：L.remove(o) ⇒ None
動作：リスト内のオブジェクト"o"を削除する。"o"に等しいオブジェクトが複数あった場合は、先頭から数えて最初に現れた"o"を削除する

■リストを逆転させる
書式：L.reverse() ⇒ None
動作："L"のアイテムの順序を逆にする。'reversed'関数や'__reversed__'メソッドと異なり、リスト本体が逆転する

```
>>> L
[0, 1, 2, 3, 4]
>>> L.extend([5])
>>> L
[0, 1, 2, 3, 4, 5]
>>> L.insert(2,100)
>>> L
[0, 1, 100, 2, 3, 4, 5]
>>> L.remove(100)
>>> L
[0, 1, 2, 3, 4, 5]
>>> L.reverse()
>>> L
[5, 4, 3, 2, 1, 0]
>>>
```

■リストを整列させる(sort items)
書式：L.sort(key=key_function, reverse=T_F) ⇒ None
動作：リスト"L"を整列する。'reverse'を省略した場合昇順に、'reverse'を指定した場合は'True'ならば降順、'False'ならば昇順に並び替える。'key'はアイテムを入れて整列に用いるキーを取り出す関数を指定する。

```
>>> L
[(0, 9), (0, 6), (1, 9), (1, 6), (2, 9), (2, 6)]
>>> L.sort(key=lambda x:x[1])    #【タプルの２つめのアイテムをキーとして整列】
>>> L
[(0, 6), (1, 6), (2, 6), (0, 9), (1, 9), (2, 9)]
>>> L.sort(key=lambda x:x[1], reverse=True)
>>> L
[(0, 9), (1, 9), (2, 9), (0, 6), (1, 6), (2, 6)]
>>>
```

（投稿：2011 年 3 月 31 日）

プラス的・エフェクト

written by Yさ

1. このゲームは

両手の手の指をある"法則"に従って立てて行き、立てた指の合計本数を10本にすれば勝ち、というゲームです。

その法則とは：
最先端の研究において科学者M・T・フーランパらによって発見された、2つの"ラッキーペア"同士をぶつけた場合の、まるで指が立つような突起形状変形反応において、反応時の突起形状の合計数が奇数になるか偶数になるかで異なる反応となる奇妙な法則が「プラス的・エフェクト」です。
一般的には"チックル効果"という名でよく知られ、歌にもなっています。

♪ゆびゆびたてたら～
...なんのこっちゃ d(^^;b

2. 動作環境は

例によって、最近のawkならOKだと思います。
ちなみに動作確認は、
GNU Awk 3.1.8, mawk 1.3.3 MBCS R27
で(WinXPのDOS窓で)行なってます。

3. 遊び方は

```
>gawk -f effect.awk[Enter]
~~~~~~~~~~~~~~~~~~~~~~~~~~
```

等で起動するとゲームスタートです。

1)最初にゲームの動作モードを選択してください。一人プレイか対戦プレイのどちらかを0～2を入力して選びます。

・一人プレイ ： 0
・対戦プレイ ： 先手=1, 後手=2

なお、

```
-v GAMEMODE=0
~~~~~~~~~~~~~
```

といった感じでオプションを付けて起動しても、動作モードを決定できます。

2)<一人プレイ>

盤面に左右の手の指の合計が偶数なのか奇数なのかを<od>/<ev>ガイド表示します。
左右どちらの手をどちらにぶつけるかを、文字で入力して[Enter]で決定してください。

入力する文字：
左手を右手に= LR or 12　　右手を左手に= RL or 32
（入力は大文字小文字を区別しません。）

左右の手の指の合計が奇数本となる場合、移動元の方の指が１つ増えます。
なお、既に５本の指が立っている手は、それ以上指を増やせないので、他の手にぶつけることはできません。

```
-----
    I     II
 ( 1 )   ( 2 )
  [L]     [R]
-----
  ↓  [R]を[L]へ = 合計3 = 奇数
-----
    I     III
 ( 1 )   ( 3 )
  [L]     [R]
-----
```

合計が偶数本の場合、ぶつけた先の指が１つ減り、元の指が２つ増えます。つまり指が１本移るイメージです。

```
-----
    I     III
 ( 1 )   ( 3 )
  [L]     [R]
-----
  ↓  [L]を[R]へ = 合計4 = 偶数
-----
  III     II
 ( 3 )   ( 2 )
  [L]     [R]
-----
```

ただし、合計が偶数本となる場合は、ぶつけた先の手に１本以上の指が無いとだめです。
また、既に４本の指が立っている手は、２本の指を増やせないので、合計が偶数となる他の手にぶつけることはできません。

両手とも５本の指を立てることができれば勝ちです。
また途中で両手とも４本指としてしまった場合は、それ以上何もできないのでゲーム終了となります。

3)<対戦プレイ>
盤面については、縦/横/斜め方向で手の指の合計が偶数なのか奇数なのかを<od>/<ev>ガイド表示します。
左右どちらの手を自分または相手のどの手にぶつけるかを、文字で入力して[Enter]で決定してください。
ただし、先手・後手それぞれの第１手目は相手にぶつけることはできません。自分の手にぶつける必要があります。
（自分の左右どちらかを自分にぶつける場合の文字は<一人プレイ>と同じです。）

入力する文字：
左手を相手の右手に= LU or 14　　左手を相手の左手に= LX or 15
左手を自分の右手に= LR or 12
右手を相手の右手に= RX or 35　　右手を相手の左手に= RU or 36
右手を自分の左手に= RL or 32
（入力は大文字小文字を区別しません。）

先手、後手が交互にいずれかの手をぶつけて指を立てて行きます。
ぶつけた手の指の合計が偶数・奇数となる場合の指の立ち方(or 移り方)については〈一人プレイ〉と同じです。
相手より先に両手の５本の指を立てることができれば勝ちです。

なお、自分の両手とも４本指となり、相手の両手とも偶数本の指だった場合は、相手に奇数本の指の手ができるまで自分は休みとなります。(自分の手をどこかにぶつけることができません。)
また途中で両者とも全ての手が４本指となってしまった場合は、それ以上何もできないのでゲーム終了となります。

4.開発環境など

ソースは awk 版として新規に作成したものです。ThinkPad の WinXP 上のテキストエディタとコマンドプロンプトな環境でやってます。
元ネタは、今年の正月(;^^ゞに甥っ子達がやっていた遊びを参考にして、ルールを適当にでっち上げて現在の形にまとめてみました。
説明の要領が悪いのか、はたまた単にオッサンの頭がついていけてないだけなのか、聞いても遊び方がさっぱり分からなかったので...

今回のバージョンのコンピュータ思考ルーチンはどうにか人間の相手ができるだけのレベルのため、先読みとかでもう少し強くできたらいいなぁと思ってます。
次回(;^^ゞ...なのか？

5. その他

本プログラムはフリーソフトです。
著作権は作者であるＹさにあります。
ただし転載、再配布、改造、消去は自由です。
(できましたら素晴らしい改造を加えた後に TSC 編集委員会宛に投稿してくださいませ)
また、このソフトを使用した事による損害が発生したとしても、損害に対しては一切の責任を負いかねます。

6. effect.awk

以下にスクリプトを掲載します。

```
##  Even-Odd Effect  written by Ｙさ

function rnd(N){ return int(N * rand()); }  ##  乱数

BEGIN{
  srand();      ##  乱数の初期化
```

```
# ゲームモード決定
gm=GAMEMODE;
if(gm!="0" && gm!="1" && gm!="2"){
  print " <Solitaire>   - 0"
  print " <Match-up>"
  print "  first move   - 1"
  print "  2'nd         - 2"
  printf("¥nWhich?[0-2]> ");
  do{
    gm=""; getline gm; gm=tolower(gm);
  }while(gm!="0" && gm!="1" && gm!="2");
}
GAMEMODE=gm+0;
## Turn;      ##  順番 0:人,1:COM
Turn=(GAMEMODE!=1)?(0):(-1);  #ゲーム開始時に+1
Times=0; #回数
## Sc[2];     ##  得点 0=人,1=COM
Sc[0]=Sc[1]=0;
## Bit[9];    ##  指 [人<1:左,3:右>, COM<9:左,7:右>]
Bit[1]=Bit[9]=0;
Bit[3]=Bit[7]=1;
## 場所判別 他
pNo[1,1]=7;  pNo[1,0]=9;
pNo[0,0]=1;  pNo[0,1]=3;
cnvPos[10]=pNo[0,1];  cnvPos[11]=pNo[1,1];  cnvPos[12]=pNo[1,0];
cnvPos[30]=pNo[0,0];  cnvPos[31]=pNo[1,0];  cnvPos[32]=pNo[1,1];
cnvPos[70]=pNo[1,0];  cnvPos[71]=pNo[0,0];  cnvPos[72]=pNo[0,1];
cnvPos[90]=pNo[1,1];  cnvPos[91]=pNo[0,1];  cnvPos[92]=pNo[0,0];
mv["LR"]=mv[12]=10;  mv["LU"]=mv[14]=11;  mv["LX"]=mv[15]=12;
mv["RL"]=mv[32]=30;  mv["RU"]=mv[36]=31;  mv["RX"]=mv[35]=32;
mv[78]=70;  mv[74]=71;  mv[75]=72;
mv[98]=90;  mv[96]=91;  mv[95]=92;

##  ゲームメイン
for(;;){
  #状況表示
  ++Times;
  dispHands(0);
  #GAME OVER / 1回休み判定
  if(gameOver()) exit;
  if(GAMEMODE!=0) if(++Turn>1) Turn=0;
  setSkip();

  do{
    #移動指定 -> 影響判定
    sw=(Turn==0)?(man()):(com());
    ev=canMove(sw);
    if(Turn==0 && ev<0){
```

```
        printf(" No effect, retry.");  pushEnter();
        dispHands(0);
      }
    }while(ev<0);
    #変化
    change(sw,ev);
  }
  exit;
}
END{
  print "¥n";
  if(finish(0)) printf("     Congratulations !!  (SCORE:%05d0)¥n", Sc[0]);
  else{
               printf("     ****** GAME OVER ******¥n");
               printf("     ...YOU LOSE (SCORE:%05d0)¥n", Sc[0]);
  }
  print;
}
function pushEnter(  tmp){ printf(" <PUSH ENTER>");  getline tmp; }
function yorn(  yn)
{
  do{ yn=""; getline yn; yn=toupper(yn); }while(yn!="Y" && yn!="N");
  return yn;
}


##
## 状況表示
##
function dispHands(sw,  d,ev,i)
{
  delete d;
  delete ev;
  makeDispTbl(sw,d,ev);

  print ;
  for(i=((GAMEMODE==0)?(6):(0)); i<=8; ++i) print d[i];
  print ;
}
function makeDispTbl(sw,d,ev,  s0,s1)
{
       #0....5....0....5....0....5....0...
  d[0]="  <s1> .<s2>   <s3>. <s4>";
  d[1]="      :             : ";
  d[2]="<s5> : zzzzz   xxxxx : COM [cccc0]";
  d[3]="     . zzzzz   xxxxx .";
  d[4]="     .               .";
  d[5]=d[4];
  d[6]="     . @@@@@   &&&&& .";
```

```
  d[7]="<s0> : @@@@@   &&&&& : YOU [yyyy0]";
  d[8]="        [L]     [R]   ";

  if(sw>=90){       s0=    mkBitStr(Bit[9],1);
                    s1=    mkBitStr(Bit[9],0);
  }else if(sw>=70){ s0=rev(mkBitStr(Bit[7],1));
                    s1=rev(mkBitStr(Bit[7],0));
  }else if(sw>=30){ s0=    mkBitStr(Bit[3],0);
                    s1=    mkBitStr(Bit[3],1);
  }else if(sw>=10){ s0=rev(mkBitStr(Bit[1],0));
                    s1=rev(mkBitStr(Bit[1],1));
  }
                          #0....5....0....5....0....5....0...
  if(sw==90){           d[2]="<s5> : zzzzz" s0 "    : COM [cccc0]";
                        d[3]="     . zzzzz" s1 "    .";             }
  if(sw==70){           d[2]="<s5> :    " s0 "xxxxx : COM [cccc0]";
                        d[3]="     .    " s1 "xxxxx .";             }
  if(sw==30){           d[6]="     . @@@@@" s0 "    .";
                        d[7]="<s0> : @@@@@" s1 "    : YOU [yyyy0]"; }
  if(sw==10){           d[6]="     .    " s0 "&&&&& .";
                        d[7]="<s0> :    " s1 "&&&&& : YOU [yyyy0]"; }
                          #0....5....0....5....0....
  if(sw==31 || sw==91){ d[4]="     .         " s0 " .";
                        d[5]="     .         " s1 " .";  }
  if(sw==32 || sw==92){ d[4]="     .    " s0 "      .";
                        d[5]="     .    " s1 "      .";  }
  if(sw==12 || sw==72){ d[4]="     .      " s0 "    .";
                        d[5]="     .      " s1 "    .";  }
  if(sw==11 || sw==71){ d[4]="     . " s0 "         .";
                        d[5]="     . " s1 "         .";  }
              #0....                                #0....
  cnvHandStr(7,"zzzzz",sw,1,d,3,2);   cnvHandStr(9,"xxxxx",sw,0,d,3,2);
  cnvHandStr(1,"@@@@@",sw,1,d,6,7);   cnvHandStr(3,"&&&&&",sw,0,d,6,7);

  setAllEval(ev);
  # YOU
  sub(/s0/, ev[0], d[7]);
  sub(/yyyy/, sprintf("%04d",Sc[0]), d[7]);
  # COM
  sub(/s1/, ev[1], d[0]);
  sub(/s2/, ev[2], d[0]);
  sub(/s3/, ev[3], d[0]);
  sub(/s4/, ev[4], d[0]);
  sub(/s5/, ev[5], d[2]);
  sub(/cccc/, sprintf("%04d",Sc[1]), d[2]);
}
function cnvHandStr(hand,regexp, dispSw,revSw,dTbl,ln1,ln2,  s0,s1)
{
                                                  #0....
```

```
  if(int(dispSw/10)==hand && (dispSw%10)!=0) s0=s1="     ";
  else{
    s0=mkBitStr(Bit[hand],0);
    s1=mkBitStr(Bit[hand],1);
    if(revSw){ s0=rev(s0);
               s1=rev(s1); }
  }
  sub(regexp, s0, dTbl[ln1]);
  sub(regexp, s1, dTbl[ln2]);
}
function mkBitStr(n,pos,  s,i)
{
  if(pos==0){
    for(i=1; i<=n; ++i) s=s "I";
    return substr("  " s "    ", 1, 5);
  }
  if(pos==1)
    return (n==5)?("- 5 "):(sprintf("( %d )", n));
}
function rev(src,  dst,len)
{
  for(len=length(src); len>0; --len) dst=dst substr(src, len, 1);
  if(substr(dst, 1, 1)==")")  dst="(" substr(dst, 2);
  if(substr(dst, 5, 1)=="(")  dst=substr(dst, 1, 4) ")";
  return dst;
}
function setAllEval(ev  )
{
  delete ev;
  ev[0]=evenOddStr(Bit[1], Bit[3]);
  ev[1]=evenOddStr(Bit[3], Bit[7]);
  ev[2]=evenOddStr(Bit[1], Bit[7]);
  ev[3]=evenOddStr(Bit[3], Bit[9]);
  ev[4]=evenOddStr(Bit[1], Bit[9]);
  ev[5]=evenOddStr(Bit[7], Bit[9]);
}
function evenOdd(b1,b2){ return ((b1+b2)%2); }
function evenOddStr(b1,b2){ return ((evenOdd(b1,b2))?("od"):("ev")); }


##
## 影響判定
##
function canMove(sw,  from,to)
{
  from=int(sw/10);
  to=cnvPos[sw];

  # 初手
```

```
  if(Times<=2){
    if(other(Turn,to)) return -1;
  }
  # 奇数
  if(evenOdd(Bit[from],Bit[to])) return ((Bit[from]<5)?(1):(-1));
  # 偶数
  if(Bit[to]<1) return -2;
  if(Bit[from]>=4) return -2;
  return 2;
}
function other(who,to,  oth){
  oth=((who==0)?(1):(0));
  if(to==pNo[oth,0] || to==pNo[oth,1]) return 1;
  return 0;
}
function finishLR(l,r){ return (Bit[l]+Bit[r]==10); }
function finish(who){ return finishLR(pNo[who,0],pNo[who,1]); }
function noEffectLR(l,r){ return (Bit[l]==4 && Bit[r]==4); }
function noEffect(who){ return noEffectLR(pNo[who,0],pNo[who,1]); }
function gameOver()
{
  if(finish(0) || finish(1)) return 1;
  if(noEffect(0) && (GAMEMODE==0 || noEffect(1))) return 1;
  return 0;
}
function setSkip(  s)
{
  s=-1;
  if(Turn==0 && noEffect(0) && evalOddEven(1)==4) s=1;
  if(Turn==1 && noEffect(1) && evalOddEven(0)==4) s=0;
  if(s>=0){ Turn=s;  printf(" Can't effect.");  pushEnter(); }
}
function evalOddEven(who){ return evalOddEvenLR(pNo[who,0],pNo[who,1]); }
function evalOddEvenLR(l,r,  minOE, maxOE)
{
  minOE=Bit[getMinPos(l,r)]%2;
  maxOE=Bit[getMaxPos(l,r)]%2;
  if(minOE==1 && maxOE==0) return 1; # 奇 < 偶
  if(minOE==0 && maxOE==1) return 2; # 偶 < 奇
  if(minOE==1 && maxOE==1) return 3; # 奇 <= 奇
  if(minOE==0 && maxOE==0) return 4; # 偶 <= 偶
  return 0;  # err
}
function getMinPos(l,r){ return ((Bit[l]<Bit[r])?(l):(r)); }
function getMaxPos(l,r){ return ((getMinPos(l,r)==l)?(r):(l)); }
function change(sw,ev,  from,to,l,r)
{
  from=int(sw/10);
  to=cnvPos[sw];
```

```
  if(ev==1) ++Bit[from];
  if(ev==2){
    --Bit[to];
    Bit[from]+=2;
  }

  #得点加算
  ++Sc[Turn];
  if(ev==2 && other(Turn,to)) Sc[Turn]+=2;
  if(finish(Turn))
    Sc[Turn]+=50;
  else if(Bit[pNo[Turn,0]]==5 || Bit[pNo[Turn,1]]==5)
    Sc[Turn]+=5;
}
function cnvMoveNo(from,to,  dir,n)
{
  n=from*10;
  for(dir=0; dir<=2; ++dir)
    if(cnvPos[n+dir]==to) return n+dir;
  return -1;  # err
}
function find(bit,who,  dir)
{
  for(dir=0; dir<=1; ++dir)
    if(Bit[pNo[who,dir]]==bit) return pNo[who,dir];
  return -1;  # not found
}
function findOEPos(dir,tgtOE,who,  list,p,st,iim)
{
  delete list;
  list[0]=getMinPos(pNo[who,0],pNo[who,1]);
  list[1]=findOtherPos(list[0],pNo[who,0],pNo[who,1]);
  st=(dir==1)?(0):(1);
  lim=(dir==1)?(2):(-1);
  for(p=st; p!=lim; p+=dir)
    if(Bit[list[p]]%2==tgtOE) return list[p];
  return -1;  # not found
}
function findOtherPos(tgt,l,r)
{
  if(tgt==l) return r;
  if(tgt==r) return l;
  return -1;  # not found
}
function move(from,to,  sw)
{
  sw=cnvMoveNo(from,to);
  dispHands(sw);
```

```
  printf("[#%02d] ... %s to %s.", Times, dirName(from), dirName(to));
  pushEnter();
  return sw;
}
function dirName(pos)
{
  return sprintf("%s %s",
    ((pos>=7)?("My"):("Your")), ((pos==1 || pos==9)?("Left"):("Right")));
}


##
##  人間 main
##
function man(  sw)
{
                                        #0...
  do{ sw=which(); dispHands(sw); printf("    OK?[Y/N]> "); }while(yorn()!="Y");
  return sw;
}
function which(  sw,tmp)
{
  printf("[#%02d] Which", Times);
  for(;;){
    printf("?> ");  tmp="";  getline tmp;
    sw=mv[toupper(tmp)];  if(sw+0==0) sw=mv[tmp+0];
    if(GAMEMODE==0){
      if(sw==10 || sw==30) return sw;
    }else{
      if((10<=sw && sw<=12) || (30<=sw && sw<=32)) return sw;
    }
  }
  return 0;  # err
}


##
##  COM main
##
function com(  ev,minP,maxP,yEv,yMin,yMax,me,you)
{
  me=1; you=0;
  ev=evalOddEven(me);
  minP=getMinPos(pNo[me,0],pNo[me,1]);
  maxP=findOtherPos(minP,pNo[me,0],pNo[me,1]);
  yEv=evalOddEven(you);
  yMin=getMinPos(pNo[you,0],pNo[you,1]);
  yMax=findOtherPos(yMin,pNo[you,0],pNo[you,1]);
```

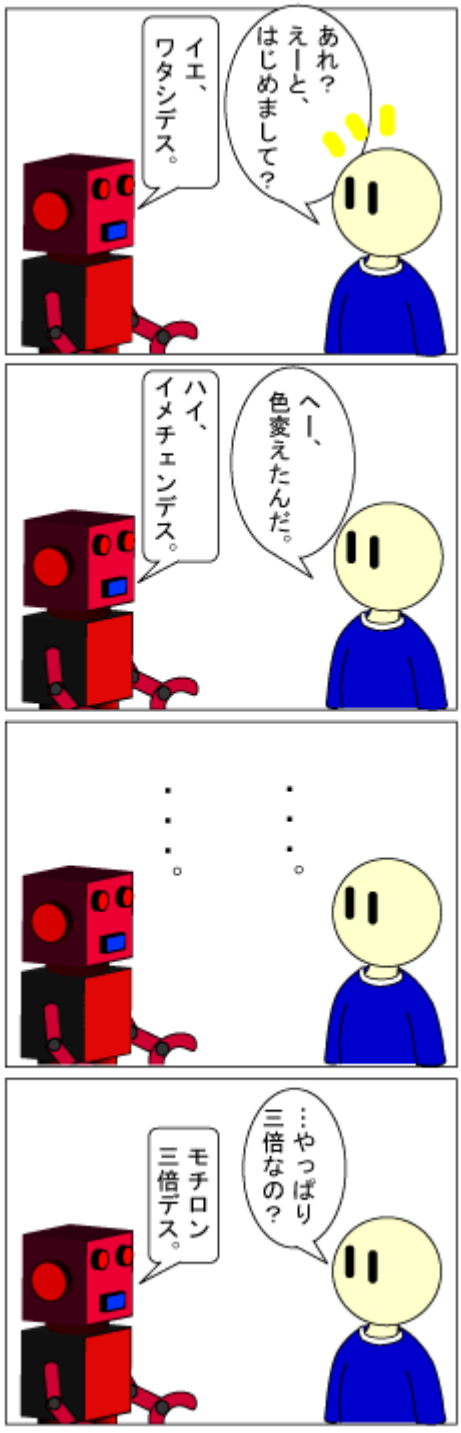
```
  if(yEv!=4)
    if((Bit[minP]==3 && Bit[maxP]==5) || (Bit[minP]==4 && Bit[maxP]==4))
      return move(minP, findOEPos(1,1,you));
  if((Bit[minP]==3 && (Bit[maxP]==4 || Bit[maxP]==3)) && (Bit[yMin]==3 && Bit[yMax]==5))
    return move(minP,yMin);
  if(Bit[minP]==2 && Bit[maxP]==5){
    if(yEv!=3 && Bit[findOEPos(1,0,you)]>0)
      return move(minP, findOEPos(1,0,you));
    return move(minP,maxP);
  }
  if(Bit[minP]==2 && Bit[maxP]==4 && Bit[yMin]==2 && Bit[yMax]==5)
    return move(minP,maxP);
  if(Bit[minP]==3 && Bit[maxP]==3 && Bit[yMin]==3 && Bit[yMax]==3)
    return move(minP,yMin)
  if(Bit[minP]==1 && Bit[maxP]==1 && Bit[yMin]==1 && Bit[yMax]==1)
    return move(minP,yMin);

  if(Times>2 && ev!=4 && yEv!=4){
    if(Bit[minP]%2)
      return move(minP,findOEPos(1,1,you));
    if(Bit[maxP]<5 && Bit[maxP]%2)
      return move(maxP,findOEPos(1,1,you));
  }

  if(Bit[yMin]==4 && Bit[yMax]==5)
    if((Bit[minP]==0 || Bit[minP]==2) && (Bit[maxP]==0 || Bit[maxP]==2 || Bit[maxP]==4))
      return move(minP,yMin);

  if(Bit[yMin]==4 && Bit[yMax]==5){
    from=find(3,1);  if(from>0) return move(from,yMax);
    from=find(1,1);  if(from>0) return move(from,yMax);
  }

  if(ev==1 || ev==2){
    from=findOEPos(1,0,me);
    to=findOtherPos(from,pNo[me,0],pNo[me,1]);
  }else{
    from=minP;
    to=maxP;
  }
  return move(from,to);
}
```

(投稿: 2011 年 4 月 15 日)
(改訂: 2011 年 4 月 29 日)

よしおさんとロボ太

海鳥

「専用」



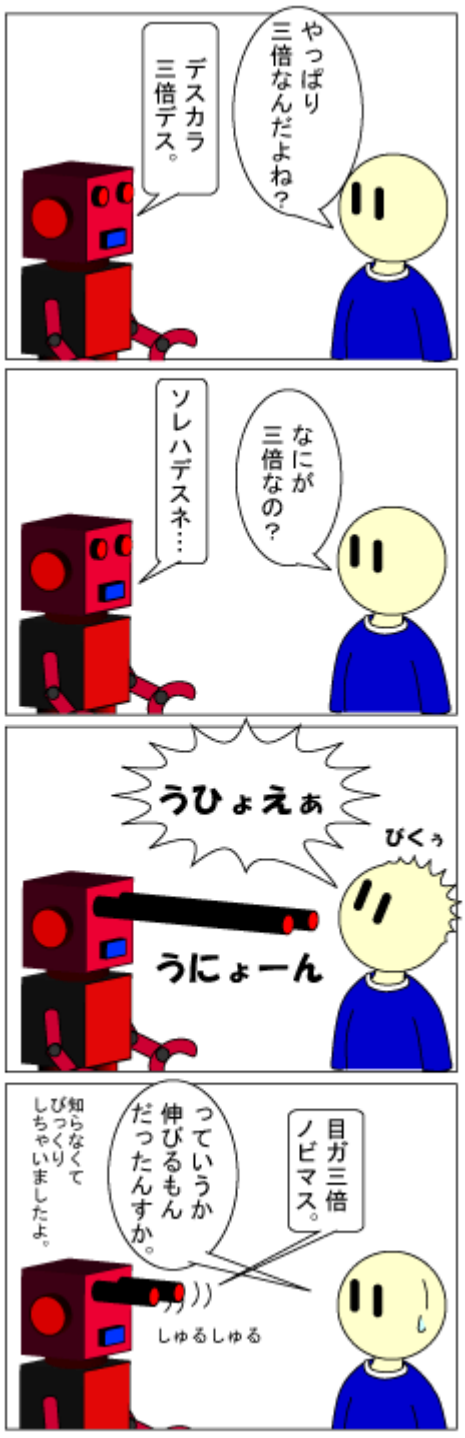
言い切っちゃったよ(さまあず三村風ツッコミ)。

(転載: 2011 年 4 月 23 日)

よしおさんとロボ太

海鳥

「三倍」



何でもアリです(ロボットだから)。

(転載: 2011 年 4 月 23 日)

スクリプト工作入門
- ツイッターを題材に II -

jscripter

1. 工作の構想

前回は、Twitter にログインせずに特定のユーザー名のタイムラインを 1 ページずつ手動で読み取り検索するスクリプト tw_read.pl を考えました。前号を刊行して少し後に、自動で読み取り検索する tw_cread.pl を考えて公開しました。今回は手動・自動両方の読み方でツイートを読んで、ツイート単位のテキストで保存するスクリプトと、保存したツイートを検索して表示するスクリプトを作ってみましょう。

2. HTML を読む問題

現在でも、http://twitter.com/username の URL が返す HTML はツイートの部分の構造は基本的には変わっていないはずです。なぜなら、まだスクリプトは意図通りに動いているからです。HTML を読み取るスクリプトの唯一の問題は、HTML が変化する可能性のあることです。なぜなら、HTML は内容や意味だけでなく表示にも関わるからです。表示を変えたくなると HTML を変更する可能性があります。tw_read.pl は、HTML の変化に追随することも容易なようにツイートの単位で切り出し、そのままテキストとして表示する方法を取っています。

3. ツイートテキストの保存と読み取りスピード制限

今回の読み取りスクリプトはツイート単位で切り出した HTML をテキストファイルとして保存します。それから、連続してツイートを読み取る tw_cread.pl は、読み取りページ数を制限するのではなくて、Twitter の 150 アクセス/時のレート制限に対応して、読み取りスピードに制限を加えることができるようにしました。現在の仕様では、約 120 アクセス/時のアクセスレートになるように設定してあります。

4. tw_read.pl

早速、新しくなったスクリプトを示します。

```
#!/Perl5.8/bin/perl.exe
use LWP::Simple;
use CGI qw(:cgi);
use URI::Escape;

### 初期設定(デフォルト項目 3 箇所のみ設定してください)

# 本 CGI を置くパスの設定
my $cgi_path = "http://localhost/cgi-bin";### デフォルト ###

# 検索パターンの取得
my $pattern = ".";#### デフォルト ###
if($ARGV[0]){
```

```
  $pattern = $ARGV[0];# コマンドライン第１引数から
}elsif(param('pattern')){
  $pattern = param('pattern');# CGI のパラメータ、pattern から
}

# username の取得
my $tw_user = "jscripter";### デフォルト ###
if($ARGV[1]){
  $tw_user = $ARGV[1];# コマンドライン第２引数から
}elsif(param('tw_user')){
  $tw_user = param('tw_user');# CGI のパラメータ、tw_user から
}

# ツイッタープロファイルの URL の取得
my $tw_url = "http://twitter.com/" . $tw_user;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合に username を取得
if(param('tw_url')){
  $tw_url = param('tw_url');
  if($tw_url =~ /^http:¥/¥/twitter.com¥/([^¥?]+)$/){
    $tw_user = $1;# CGI パラメータを URL が持たない場合
  }elsif($tw_url =~ /^http:¥/¥/twitter.com¥/(.+?)¥?/){
    $tw_user = $1;# CGI パラメータを URL が持つ場合
  }
}

# 現在のページの取得
my $ppage = 0;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合にページ数を取得
if($tw_url =~ /page=(¥d+)/){
       $ppage = $1;
}

# 次ページがない場合のフラグ(1: 次ページなし)
my $nopage = 0;

my $npage = 0;# 次ページ数
my $new_tw_url = "";# 次ページの URL
my $escaped_next_url = "";# 次ページの URL の値をエスケープしたもの

my $prevpage = 0;# 前ページ数
my $prev_tw_url = "";# 前ページの URL
my $escaped_prev_url = "";# 前ページの URL の値をエスケープしたもの

### CGI 最初の部分の出力

print <<HEADER;
Content-type: text/html; charset=UTF-8

<html>
```

```
<style type="text/css">
<!--
body { margin: 1em 2em; }
div,a { text-align: left; }
div.status-body { text-indent: 0em; font-family: monospace;
  width: 40em; line-height: 1.5em; background-color: lightyellow;
  border-width: 10px 20px 10px 10px; }
div.entry-content { text-indent: 1em; background-color: lightblue; }
div.meta { text-indent: 1em; background-color: lightyellow; }
div.shared-content { text-indent: 1em; background-color: lightyellow; }
-->
</style>
<body>
HEADER

###

# 読み込みURLのCGI出力
print "<p>${tw_url}</p>¥n";

### ツイートデータの取得

# LWP::Simpleのgetメソッドに入力するためのURL変換
$tw_url =~ s/¥&amp;/¥&/g;

# LWP::Simpleのgetメソッドで URLに対応するHTMLを得る
my $tw = get($tw_url);

# HTML中のツイート部分を個別に配列に格納
# <li class="hentry...>から</li>あるいは</li >までの範囲を取得
my @status_bodys = ($tw =~ /<li¥s+class="hentry.+?>(.+?)<¥/li.*?>/sg);

# 次ページのURLを取得、現在のページ数の取得
if($tw =~ /<a href="(¥/${tw_user}¥?.+?)"/s){
  $next_tw_url = $1;# 次ページのURL
  if($next_tw_url =~ /page=(¥d+)/){
    $npage = $1;# 次ページ数
    $ppage = $npage - 1;# 現在のページ数
  }
}else{
  if($tw_url =~ /(¥/${tw_user}.+)/){
    $next_tw_url = $1;
  }
  $nopage = 1;# 次ページのURLが取得できない
}
if($ppage == 0){
  $ppage = 1;# 現在のページを取得できない場合など
}
```

```
###

# ページ数の CGI 出力
print "<center>" . $ppage . "</center>¥n";

print "<hr>";

### ツイートの CGI 出力とファイル出力
my $year = "";
my %mon = ();
my $month = "";
my $tweet = "";
foreach (@status_bodys){
  if(/>.*?$pattern.*?</so){
    # 表示上、span タグは改行されないので、div タグに変換
    s/<span/<div/sg;
    s/<¥/span>/<¥/div>/sg;
    # @username や#hashtag の処理
    # "/username"や"/search?q=#hashtag"のハイパーリンクを補完する
    s/(<.*?)href="(¥/[^¥/]+?)"/${1}href="http:¥/¥/twitter.com$2"/sg;
    # published timestamp クラスから時間データを取得する
    if(/published timestamp.+?¥w{3} (¥w{3}) (¥d{1,2}) (¥d{1,2}):(¥d{1,2}):(¥d{1,2}) [¥-¥
+]¥d{4} (¥d{4}).+?>/)
    {
      $year = $6;
      $month = $1;
      %mon = (
        'Jan' => '01',
        'Feb' => '02',
        'Mar' => '03',
        'Apr' => '04',
        'May' => '05',
        'Jun' => '06',
        'Jul' => '07',
        'Aug' => '08',
        'Sep' => '09',
        'Oct' => '10',
        'Nov' => '11',
        'Dec' => '12',
      );
      unless (-e $year){
        mkdir $year;# 年フォルダがなければ作る
      }
      unless (-e "$year/$mon{$month}"){
        mkdir "$year/$mon{$month}";# 月フォルダがなければ作る
      }
      unless (-e "$year/$mon{$month}/$tw_user"){
        mkdir "$year/$mon{$month}/$tw_user";# ユーザー名フォルダがなければ作る
      }
```

```
      $tweet = $year . $mon{$month} . $2 . $3 . $4 . $5 . "_" . $tw_user . ".txt";
      if (-e "$year/$mon{$month}/$tw_user/$tweet"){
        print "*";# ツイートファイルが存在すれば、"*"を表示する
      }else{
        open(OUT, "> $year/$mon{$month}/$tw_user/$tweet");
        print OUT $_;# ツイートファイルが存在しなければ、出力する
        close OUT;
        # 経過時間の表示の代わりにpublished timestampの時間データを表示する
        s/(<div.+?published timestamp.+?{time:')(.+?)('}">).+?(<¥/div>)/$1$3$2$4/;
      }
    }
    print $_;
  }
}
###

print "<hr>¥n";
print "<center>";

### ページの移動リンクのCGI出力

# 戻るページへのリンク表示
if($ppage > 1){
  $prevpage = $ppage - 1;
  ($prev_tw_url = $next_tw_url) =~ s/page=¥d+/page=${prevpage}/;
  $escaped_prev_url = uri_escape("http://twitter.com${prev_tw_url}");
  print "<a href=¥"${cgi_path}/twitter_read.pl?pattern=${pattern}¥&tw_url=$
{escaped_prev_url}¥">${prevpage}ページへ戻る</a> <---> ";
}

# 進むページへのリンク表示
if($nopage){
  print "もうページはありません。¥n";# 最後のページの場合
}else{
  $escaped_next_url = uri_escape("http://twitter.com${next_tw_url}");
  print "<a href=¥"${cgi_path}/twitter_read.pl?pattern=${pattern}¥&tw_url=$
{escaped_next_url}¥">${npage}ページへ進む</a>¥n";
}

###

# CGIの最後の部分の出力
print "</center>¥n</body>¥n</html>¥n";
__END__
```

前回のtw_read.plと違う部分は、ツイートのCGI出力の部分が、加えて、ツイート単位でファイル出力するように変わったところです。ファイル出力する際のディレクトリは「4桁年/2桁月/username」、例えば、「2011/04/jscripter」というディレクトリに、ファイル名は「4桁年2桁月

2桁時2桁分2桁秒_username.txt」、例えば、「20110422120000_jscripter.txt」というファイル名で出力されます。

ディレクトリやファイル名の時間は、ツイートの時間経過や時間を表示している部分の「published timestamp」の span クラスの HTML から切り出します。時間は世界標準時のままになります。具体的な HTML は次のようになっています。

```
<span class="published timestamp" data="{time:'Sun Dec 19 02:47:38 +0000
2010'}">約1時間前</span></a>
```

ここから、正規表現によるパターンマッチで必要な部分を抜き出して使っています。詳細はスクリプトを参照すること。ほとんど、Perl の基本的な機能を組み合わせてスクリプトは作られています。

次のような HTML フォームと組み合わせて、CGI としてセットして使うことができます。

```
<!-- Twitter 検索用 CGI - tw_read.pl -->
<FORM action="cgi-bin/tw_read.pl" target="main">Twitter プロファイル検索(ページ単位)<br>
  username: <INPUT type="text" name="tw_user" value="jscripter" size="20"><br>
  検索パターン: <INPUT type="text" name="pattern" value="." size="20"><br>
  <INPUT type="submit"><INPUT type="reset">
</FORM>
```

5. tw_cread.pl

次に、一括してプロファイルタイムラインを読み込む tw_cread.pl を示します。tw_read.pl と同じ部分が多いので、比較しながら違いを見つけてください。新しい tw_read.pl と同様に、ツイート単位でテキストを切り出し、ファイルとして格納します。

```
#!/Perl5.8/bin/perl.exe
use LWP::Simple;
use CGI qw(:cgi);
use URI::Escape;

### 初期設定(必要な場合、デフォルト項目3箇所のみ設定してください)

# 本 CGI を置くパスの設定
my $cgi_path = "http://localhost/cgi-bin";### デフォルト ###

# 検索パターンの取得
my $pattern = ".";#### デフォルト ###
if($ARGV[0]){
  $pattern = $ARGV[0];# コマンドライン第1引数から
```

```
}elsif(param('pattern')){
  $pattern = param('pattern');# CGI のパラメータ、pattern から
}

# username の取得
my $tw_user = "jscripter";### デフォルト ###
if($ARGV[1]){
  $tw_user = $ARGV[1];# コマンドライン第 2 引数から
}elsif(param('tw_user')){
  $tw_user = param('tw_user');# CGI のパラメータ、tw_user から
}

# 追加モードの設定
my $addmode = 1;# 2: 全取得モード
if($ARGV[2]){
  $addmode = $ARGV[2];
}elsif(param('addmode')){
  $addmode = param('addmode');
}

# ツイッタープロファイルの URL の取得
my $tw_url = "http://twitter.com/" . $tw_user;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合に username を取得
if(param('tw_url')){
  $tw_url = param('tw_url');
  if($tw_url =~ /^http:¥/¥/twitter.com¥/([^¥?]+)$/){
    $tw_user = $1;# CGI パラメータを URL が持たない場合
  }elsif($tw_url =~ /^http:¥/¥/twitter.com¥/(.+?)¥?/){
    $tw_user = $1;# CGI パラメータを URL が持つ場合
  }
}

# 現在のページの取得
my $ppage = 0;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合にページ数を取得
if($tw_url =~ /page=(¥d+)/){
  $ppage = $1;
}

# 次ページがない場合のフラグ(1: 次ページなし)
my $nopage = 0;

my $npage = 0;# 次ページ数
my $new_tw_url = "";# 次ページの URL
my $escaped_next_url = "";# 次ページの URL の値をエスケープしたもの

my $prevpage = 0;# 前ページ数
my $prev_tw_url = "";# 前ページの URL
my $escaped_prev_url = "";# 前ページの URL の値をエスケープしたもの
```

```
### CGI 最初の部分の出力

print <<HEADER;
Content-type: text/html; charset=UTF-8

<html>
<style type="text/css">
<!--
body { margin: 1em 2em; }
div,a { text-align: left; }
div.status-body { text-indent: 0em; font-family: monospace;
  width: 40em; line-height: 1.5em; background-color: lightyellow;
  border-width: 10px 20px 10px 10px; }
div.entry-content { text-indent: 1em; background-color: lightblue; }
div.meta { text-indent: 1em; background-color: lightyellow; }
div.shared-content { text-indent: 1em; background-color: lightyellow; }
-->
</style>
<body>
HEADER

###

while(!$nopage){

# 読み込み URL の CGI 出力
print "<p>${tw_url}</p>¥n";

### ツイートデータの取得

# LWP::Simple の get メソッドに入力するための URL 変換
$tw_url =~ s/¥&amp;/¥&/g;

# LWP::Simple の get メソッドで URL に対応する HTML を得る
my $tw = get($tw_url);

# HTML 中のツイート部分を個別に配列に格納
# <li class="hentry...>から</li>あるいは</li >までの範囲を取得
my @status_bodys = ($tw =~ /<li¥s+class="hentry.+?>(.+?)<¥/li.*?>/sg);

# 次ページの URL を取得、現在のページ数の取得
if($tw =~ /<a href="(¥/${tw_user}¥?.+?)"/s){
  $next_tw_url = $1;# 次ページの URL
  if($next_tw_url =~ /page=(¥d+)/){
    $npage = $1;# 次ページ数
    $ppage = $npage - 1;# 現在のページ数
  }
}else{
```

```
  if($tw_url =~ /(¥/${tw_user}.+)/){
    $next_tw_url = $1;
  }
  $nopage = 1;# 次ページの URL が取得できない
}
if($ppage == 0){
  $ppage = 1;# 現在のページを取得できない場合など
}

###

# ページ数の CGI 出力
print "<center>" . $ppage . "</center>¥n";

print "<hr>";

### ツイートの CGI 出力
my $year = "";
my %mon = ();
my $month = "";
my $tweet = "";
foreach (@status_bodys){
  if(/>.*?$pattern.*?</so){
    # 表示上、span タグは改行されないので、div タグに変換
    s/<span/<div/sg;
    s/<¥/span>/<¥/div>/sg;
    # @username や#hashtag の処理
    # "/username"や"/search?q=#hashtag"のハイパーリンクを補完する
    s/(<.*?)href="(¥/[^¥/]+?)"/${1}href="http:¥/¥/twitter.com$2"/sg;
    # published timestamp クラスから時間データを取得する
    if(/published timestamp.+?¥w{3} (¥w{3}) (¥d{1,2}) (¥d{1,2}):(¥d{1,2}):(¥d{1,2}) [¥-¥
+]¥d{4} (¥d{4}).+?>/)
    {
      $year = $6;
      $month = $1;
      %mon = (
        'Jan' => '01',
        'Feb' => '02',
        'Mar' => '03',
        'Apr' => '04',
        'May' => '05',
        'Jun' => '06',
        'Jul' => '07',
        'Aug' => '08',
        'Sep' => '09',
        'Oct' => '10',
        'Nov' => '11',
        'Dec' => '12',
      );
```

```
      unless (-e $year){
        mkdir $year;
      }
      unless (-e "$year/$mon{$month}"){
        mkdir "$year/$mon{$month}";
      }
      unless (-e "$year/$mon{$month}/$tw_user"){
        mkdir "$year/$mon{$month}/$tw_user";
      }
      $tweet = $year . $mon{$month} . $2 . $3 . $4 . $5 . "_" . $tw_user . ".txt";
      if (-e "$year/$mon{$month}/$tw_user/$tweet"){
        print "*";
        if($addmode == 1){
          print "新規ツイートの取得を完了しました!¥n";
          $nopage = 1;
          last;
        }
      }else{
        open(OUT, "> $year/$mon{$month}/$tw_user/$tweet");
        print OUT $_;
        close OUT;
        # 経過時間の表示の代わりに published timestamp の時間データを表示する
        s/(<div.+?published timestamp.+?{time:')(.+?)('}">).+?(<¥/div>)/$1$3$2$4/;
      }
    }
      print $_;
  }
}
###

print "<hr>¥n";

### 進むページへの URL 設定

if($nopage){
       print "もうページはありません。¥n";# 最後のページの場合
       last;
}else{
       $tw_url = "http://twitter.com${next_tw_url}";
}

###

# while loop interval
sleep int(rand(10)) + 25;# アクセス頻度: 120 回/時間程度

}# while loop end

# CGI の最後の部分の出力
```

```
print "</body>¥n</html>¥n";
__END__
```

tw_cread.pl はプロファイルタイムラインを読んだ時に次のページの URL が取得できれば、続けてそのページを読むループを備えています。tw_read.pl は新しい URL を読むためのリンクを生成して表示し、ユーザーのクリックを待つプログラムです。

Twitter は時間あたりのアクセス頻度を無認証アクセスの場合、150 アクセス/時までに制限しています。tw_cread.pl は下記のループの最後の部分で、適当な時間だけさぼります。

```
# while loop interval
sleep int(rand(10)) + 25;# アクセス頻度: 120 回/時間程度
```

この sleep は、約 30 秒程度のループ間隔をもたらすので、120 回/時のアクセス頻度に抑制されることになります。

最初の方の「追加モード」の設定は、ユーザープロファイルタイムライン全体を取得するか、新しい部分だけを取得するかを設定できるようにしています。例えば、tw_read.pl で不定期にデータを取得している場合は、データの抜けがある場合が想定されるからです。初めて取得する場合や最新の部分だけを取得するのでは足りない場合は全体を取得するモードを使います。tw_cread.pl だけで運用する場合は、追加モードだけで十分です。全取得モードにするためには、0 と 1 以外の数字、例えば、2 にするとよいでしょう。

```
# 追加モードの設定
my $addmode = 1;# 2: 全取得モード
if($ARGV[2]){
  $addmode = $ARGV[2];
}elsif(param('addmode')){
  $addmode = param('addmode');
}
```

例によって、コマンドラインからでもパラメータを取得できるようになっています。それでは、パラメータ入力用の HTML フォームの HTML を示しておきましょう。

```
<FORM action="cgi-bin/tw_cread.pl" target="_blank">Twitter プロファイル検索(一括)<br>
  username: <INPUT type="text" name="tw_user" value="jscripter" size="20"><br>
  検索パターン: <INPUT type="text" name="pattern" value="." size="20"><br>
  検索モード(追加: 1 全: 2): <INPUT type="text" name="addmode" value="1" size="3"><br>
  <INPUT type="submit"><INPUT type="reset">
</FORM>
```

6. tw_aread.pl

さて、次は tw_read.pl や tw_cread.pl で切りだして保存したツイートを検索して表示するスクリプトを作ってみましょう。Twitter のアクセス制限を気にすることなく、自由に検索して表示することができます。

```
#!/Perl5.8/bin/perl.exe
use LWP::Simple;
use CGI qw(:cgi);
use URI::Escape;

### 初期設定(必要な場合、デフォルト項目のみ設定してください)

# baseurl の設定
$baseurl = "http://localhost/cgi-bin";### デフォルト ###

# 検索パターンの取得
my $pattern = ".";#### デフォルト ###
if($ARGV[0]){
	$pattern = $ARGV[0];# コマンドライン第1引数から
}elsif(param('pattern')){
	$pattern = param('pattern');# CGI のパラメータ、pattern から
}

# username の取得
my $tw_user = "jscripter";### デフォルト ###
if($ARGV[1]){
	$tw_user = $ARGV[1];# コマンドライン第2引数から
}elsif(param('tw_user')){
	$tw_user = param('tw_user');# CGI のパラメータ、tw_user から
}

# ツイートの格納ディレクトリの設定
my $basedir = "C:/anhttpd/cgi-bin";### デフォルト ###

###

# 西暦年4桁のディレクトリ名の取得
my @yeardirs = ();
opendir(DIR, $basedir) || die "Can't open: $!";
my @yeardirs = grep(/^¥d{4}$/, readdir(DIR));
closedir(DIR);

### CGI 最初の部分の出力

print <<HEADER;
Content-type: text/html; charset=UTF-8
```

```
<html>
<head>
<base href="$baseurl" />
<style type="text/css">
<!--
body { margin: 1em 2em; }
div,a { text-align: left; }
div.status-body { text-indent: 0em; font-family: monospace;
       width: 40em; line-height: 1.5em; background-color: lightyellow;
       border-width: 10px 20px 10px 10px; }
div.entry-content { text-indent: 1em; background-color: lightblue; }
div.meta { text-indent: 1em; background-color: lightyellow; }
div.shared-content { text-indent: 1em; background-color: lightyellow; }
-->
</style>
</head>
<body>
HEADER

###

my $year = "";
my $mon = "";
my $user = "";
my @mondirs = ();
my @users = ();
foreach $year (@yeardirs){
# 西暦年ディレクトリ下、月2桁のディレクトリ名の取得
  opendir(YDIR, $basedir . "/" . ${year}) || die "Can't open: $!";
  @mondirs = grep(/^¥d{2}$/, readdir(YDIR));
  closedir(YDIR);
  foreach $mon (@mondirs){
    if($tw_user =~ /^all$/i){
# 西暦年4桁月2桁のディレクトリ下、ユーザーのディレクトリ名の取得
      opendir(MDIR, $basedir . "/" . $year . "/" . $mon) || die "Can't open$!";
      @users = grep(/^¥w+$/, readdir(MDIR));
      closedir(MDIR);
      foreach $user (@users){
        $udir = $basedir . "/" . $year . "/" . $mon . "/" . $user;
        &twsearch($udir, $pattern);
        }
    }else{
      $udir = $basedir . "/" . $year . "/" . $mon . "/" . $tw_user;
      &twsearch($udir, $pattern);
    }
  }
}
```

```
# CGI の最後の部分の出力
print "</body>¥n</html>¥n";

# 西暦年 4 桁月 2 桁ユーザー名ディレクトリ下、ツイートテキストの取得
sub twsearch{
  my($udir, $pattern) = @_;
  my @txts = ();
  my $txt = "";
  my $html = "";
  if(opendir(UDIR, $udir)){
    @txts = grep(/^¥d{14}/, readdir(UDIR));
    closedir(UDIR);
    # ツイートテキストの検索出力
    foreach $txt (@txts){
      open(IN,  $udir . "/" . $txt) || die "Can't open: $!";
      while(<IN>){
        $html .= $_;
      }
      close(IN);
      if($html =~ />.*?$pattern.*?</iso){
        print "<p>User Directory: $udir; Pattern: $pattern</p>¥n";
        print "[" . ++$num . "]¥n";
        print $html;
      }
      $html = "";
    }
  }
  1;
}
__END__
```

このスクリプトはデスクトップで動く単純な検索スクリプトです。ディレクトリ操作やファイル名の取得など基礎的な技術の積み重ねになっています。スクリプトを参考にしてください。サブルーチンの書き方なども参考になるでしょう。

サブルーチンの twsearch はユーザーディレクトリのパスと検索パターンを引数としますが、ディレクトリが開けない場合は、true を返すだけで何もしません。存在する年月のすべてのディレクトリに保存してあるすべてのユーザーのディレクトリが存在するとは限らないからです。

HTML フォームの HTML を次に示しておきます。

```
<FORM action="cgi-bin/tw_aread.pl" target="_blank">Twitter プロファイルアーカイブ検索<br>
  username: <INPUT type="text" name="tw_user" value="jscripter" size="20"><br>
  検索パターン: <INPUT type="text" name="pattern" value="." size="20"><br>
  <INPUT type="submit"><INPUT type="reset">
</FORM>
```

7. 最後に

最後にといっても、この記事の最後であって、スクリプト工作はさらに発展していくだろうと思います。

この三つのスクリプトは僕のデスクトップにセットされていて毎日動いています。次のステップでは読みに行くユーザー名を選べるようしたり、ツイートの統計を表示したり、夜間に複数のユーザーのプロファイルタイムラインを連続的に取得したりするシステムを構築するかもしれません。

読者の皆さんの中から、スクリプト工作をされる方が出て、TSNET スクリプト通信に記事を投稿されるようになることを期待しています。

それではまた次号でお目にかかりましょう。

（投稿：2011 年 4 月 23 日）
（改訂：2011 年 5 月 3 日）

編集後記

jscripter

iPhone や iPod touch、あるいは iPad のような高機能モバイルデバイスが登場して以来、コンピューティングの主役は、明らかにデスクトップからモバイルに移りつつある。既に Apple の売上の半分は iPhone によるものとの報道がなされている。1990 年代に汎用機やミニコン・ワークステーションから PC へとコンピューティングの主役が変わったところを再度目撃しているような気分になる。

ユービキタス・コンピューティング(Ubiquitous computing)という用語は、Wikipedia によれば、Xerox PARC の Mark Weiser が 1991 年に提唱したそうだが、ようやく相応しい時代に入ったということができるだろう。

ユービキタス・コンピューティング時代に、Twitter は単なる使いやすいコミュニケーションツールあるいは SNS であるだけでなく、大変便利な情報ツールとしても機能する。タイムラインに見つけた情報はリツイート・ワンクリックで簡単にメモできる。自らもインターネットにつながればどこからでもツイートでメモできる。無論、オープンにできるような事柄でないと扱えないかもしれない。それさえ許せば大変便利である。

今では、パーソナル・コンピューターはオフィスツールなどではなく、個人の能力を増強するものとして意義付けられる。スクリプティング言語も今後はそのような役割をはたすものとして使われていくことだろう。

(投稿: 2011 年 4 月 23 日)

改訂版後記として、少し書き加えておこう。5 月目前に刊行したため、プレ公開にして執筆者の最終確認を取っていたら、Ｙささんから、スクリプト修正の連絡が入った。僕自身も本格的に使い始めると不具合が見つかり始めた。それらの修正を含めた 3.4.002 版を刊行します。スクリプトのどこが違うか確認するのもご参考になるかも^^;)そうそう、ツイッターのツイート保持件数の保証が 3200 件ぐらいまでという仕様も、1 例ですが、tw_cread.pl で確認したことを報告しておきます。ツイートを残したい場合にはご自分で対策が必要です。

(投稿: 2011 年 5 月 3 日)

ＴＳＮＥＴスクリプト通信

ISSN 1884-2798　出版地: 広島市

2011 年 5 月 3 日　　3.4.002 版
2011 年 4 月 24 日　　3.4.001 版

投稿規程

TSNETWiki ： 「投稿規程」のページを参照のこと

編集委員会(投稿順)

機械伯爵　kikwai[at]livedoor[dot]com
Ｙさ　saw[at]mf-nokuchi2pho[dot]ne[dot]jp
海鳥　kaityo256[at]nifty[dot]ne[dot]jp
jscripter　jscripter9[at]gmail[dot]com

著作権

使用許諾・配布条件

免責事項

編集ソフトウェア

OpenOffice.org 3.2.1 Writer

発行所

一次配布所: TSNET スクリプト通信刊行リスト
http://text.world.coocan.jp/TSNET/?TSNET%E3%82%B9%E3%82%AF%E3%83%AA%E3%83%97%E3%83%88%E9%80%9A%E4%BF%A1%E5%88%8A%E8%A1%8C%E3%83%AA%E3%82%B9%E3%83%88

TSNET スクリプト通信 第3巻第4号(通算第12号)
発行: TSC編集委員会 発行日: 2011年4月24日
ISSN: 1884-2798 出版地: 広島市 創刊: 2008年5月7日