# TSNETスクリプト通信

TSC編集委員会
ISSN 1884-2798

# 目次



---

表紙写真: 羊蹄山の雲海
撮影: ムムリク
日時: 1985年4月
場所: ニセコ
メモ: TSNETスクリプト通信第7号の十勝岳に続く第二弾(jscripter)

# 巻頭言

jscripter

新年の第15号が無事出る。

Yささんの「イマジン」は夢の中で生まれたトランプ・ゲーム。お楽しみください。

でびさんからは初投稿をいただいた。Excelのマクロである。数取器をExcelで実現している。クリックし過ぎて、間違えた場合は引き算ができて戻れるのが、普通の数取器と違うところ。

ムムリクさんの「MeCabでルビ振り」。高度なテキスト処理をしようと思えば、テキストの形態素解析をする必要が出てくる。そこで、MeCabが登場する。TSNETスクリプト通信では初めて取り上げられる。勉強になりそう。

機械伯爵さんの「Pythonの文法」草稿の続編、第6回。文字列操作に関連したメソッドの解説である。

新年は、TSNETで質問して勉強した。PCのIPアドレスを取得する方法についてである。デスクトップCGIにおいては、DHCPによるIPアドレスをCGIで取得できると自動化に便利なのである。

そのうち、でびさんからも、サブディレクトリのファイルをサブディレクトリ名をファイル名に付加・リネームして、一つ上のディレクトリに移動するバッチについて質問された。これも記事としてまとめた。

さて、編集者は、引き続きよたよたとしていて、みなさんの投稿を受け止めるのに精一杯となってしまった。「覚醒と進化 試論 II」はDesktopWebフレームワークを作る出発点としてはささやかだが、小さな一歩が大きく発展すると信じている。

以上。

（投稿: 2012年 2月 5日）

# イマジン

written by Yさ

## 1. このゲームは

※タイトルは平和を願う歌の邦題を彷彿させますが、内容は全く関係ありません...
また、仕事や恋に自分らしさを模索する女性を描いた漫画とも関係ありません。

コンピュータと対戦する形式のゲームで、互いに何枚かのカードを伏せて配置し、推理して先に全てを当てた方が勝ちとなるトランプゲームです。

※「貧相な画面のゲーム」なので分かりにくいのですが、
双方が'黒'のカードを伏せて配置し、'赤'のカードで推理します。
(;^^ゞ

## 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 4.0.0, 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 imagine.awk[Enter]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

等で起動するとゲームスタートです。

1)最初にゲームのレベル('場'のサイズ)を選択してください。通常プレイか練習プレイのどちらかを3,5を入力して選びます。

・通常プレイ ： 5
・練習プレイ ： 3

なお、

```
-v FIELDSIZE=5
~~~~~~~~~~~~~~~
```

といった感じでオプションを付けて起動しても、レベルを決定できます。

2)<セットアップ>

まず最初にコンピュータに当てさせる自分の'黒'(スペード)のカードを配置します。
盤面に'場'と'手札'をガイド付きで表示されます。どこに何のカードを配置するかを'場'と'手札'(または逆順に)2文字で入力して[Enter]で決定してください。

入力する文字：

'場'： a～e（※練習プレイ時は a～c)
'手札'： 1～9,0,j,q,k（※なお"10"を画面の都合で'0'で表現しています^.^;)

また、','(カンマ)や空白で区切って、複数を一度に入力することもできます。

例) a1,2b[Enter]

全ての配置が終わったら、'z'を入力して「セットアップ」を終了してください。

3)<対戦>

「セットアップ」と同様に盤面に'場'と'手札'がガイド付きで表示されます。

相手が伏せて配置した'黒'(クラブ)のカードのどの場所に何の数字(1～9,0,j,q,k)があるかを推理して、自分の'赤'(ハート)の手札を、文字で入力して[Enter]で配置してください。

(入力する文字は<セットアップ>と同じです。)
全ての推理配置が終わったら、'z'を入力して終了してください。

続けて、コンピュータが人間側の配置を推理して'赤'(ダイヤ)のカードを配置した後、双方の当り外れの判定が行われます。

a.当り = 置いた'赤'のカードと伏せた'黒'のカードの位置・数字が一致
b.外れ = 置いた'赤'のカードの数字が、伏せた'黒'のカードの中に無い
c.位置違い = 置いたカードの数字が、伏せたカードの中にあるが位置が違う

'当り'の場合、相手が伏せて置いた'黒'のカードが取り除かれます。
'外れ'の場合、自分が推理して'場'に置いた'赤'のカードが無くなります。
'位置違い'の場合、'場'に置いた'赤'のカードはそのまま残ります。

双方がカードの推理を繰り返します。

どちらかの伏せて置かれた'黒'のカード全てが全て取り除かれると、勝敗を表示して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。

今回のトランプゲームは、一応、元ネタ無しのオリジナルです(^^; 実はなんと夢の中で誰かと遊んでいて、目覚めた時まで覚えていたというものなのですw

(...といってもオリジナルが存在するかどうかをちゃんと調査した訳ではないので、実際は違うのかもしれませんが)

## 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. imagine.awk

以下に本記事のスクリプト、imagine.awkを収録します。

---

```
##  imagine   written by Ｙさ

BEGIN{
  srand();

  # '場'のサイズ決定
  gm=FIELDSIZE;  ## 5=normal, 3=training
  if(gm!="5" && gm!="3"){
    print " <LEVEL>"
    print "  training   - 3"
    print "  normal     - 5"
    printf("¥n Which?(3 or 5) >");
```

```
    do{ gm="";  getline gm;  gm=tolower(gm); }while(gm!="5" && gm!="3");
  }
  FIELDSIZE=gm+0;
  print "¥n";

  ## 初期設定

  # カード
  split("S,D,C", StrSuit, ",");  StrSuit[0]="H";
  split("2,3,4,5,6,7,8,9,0,J,Q,K", StrRank, ",");  StrRank[0]="A";
  # Cards[key, 52];  ## カード本体
  for(s=0; s<4; ++s){
    for(r=0; r<13; ++r){
      p=s*13 +r;
      Cards["suit", p] = s;  # マーク
      Cards["rank", p] = r;  # 数字
    }
  }
  # ガイド
  split("a,b,c,d,e", NtoA, ",");
  for(i=1; i<=9; ++i) Guide[i]=i "";
  Guide[10]="0"; Guide[11]="j"; Guide[12]="q"; Guide[13]="k";

  YOU=0; COM=1;
  RED=0; BLACK=1;
  # Field[who, kind, FIELDSIZE];  ## 場札 0="A"～12="K",-1=空き
  #   who(0=人,1=COM)
  #   kind(RED=攻, BLACK=守)

  # Hands[who, suit, rank]  ## 手札 0=未使用,1-5=指定済,9=当り,-1=取られ済
  #   who(0=人,1=COM)
  #   suit(攻,守  (人:0="H",1="S" / COM:2="D",3="C"))
  #   rank(0="A"～12="K")

  ##  ゲームメイン
  do{
    setUp();
    do{ you(RED);  com(RED);  display();  judge(); }while( ! allOpen());
    result();
  }while(decision("Continue")=="y");
  exit;
}
```

```
function setUp(  who,n,suit,rank)
{
  HIDE=1;  ## COMのBLACK='守'を伏せて表示

  # '場'をクリア
  for(who=0; who<=1; ++who)
    for(n=1; n<=FIELDSIZE; ++n)
      Field[who,RED,n]=Field[who,BLACK,n]=-1;
  # '手札'をクリア
  for(who=0; who<=1; ++who)
    for(suit=0; suit<4; ++suit)
      for(rank=0; rank<13; ++rank) Hands[who,suit,rank]=0;

  # COM用  ## 候補リスト
  for(rank=0; rank<13; ++rank) TryField[rank,0]=0;

  you(BLACK);  com(BLACK);
  print "¥n";
}

## setUp用表示
function setUpDisplay(  i,n)
{
  print "¥n----";
  dispSpace(13-FIELDSIZE);  dispFieldGuide();
  dispSpace(13-FIELDSIZE);  dispField(YOU, BLACK);  print "  <YOU¥n";

  dispHands(YOU, BLACK);  dispGuide(BLACK);
}

## ゲーム用表示
function display(  i,n,v)
{
  print "¥n====";
  dispHands(COM, RED);  print "";

  dispFieldGuideSkip();  dispFieldSkip(COM);
  dispSpace(13-FIELDSIZE*2);  dispField(COM, RED);  print "  <COM";

  dispField(YOU, RED);
  dispSpace(13-FIELDSIZE*2);  dispFieldSkip(YOU);   print "  <YOU¥n";
```

```
  dispHands(YOU, RED);  dispGuide(RED);
}

# 手札 カード表示
function dispHands(who,kind,  i)
{
  for(i=0; i<13; ++i) printf("%4s", cardString(who, kind, i));
  print "";
}
# 手札 ガイド表示
function dispGuide(kind,  i,v)
{
  for(i=0; i<13; ++i){
    v=getHands(YOU, kind, i);
    printf("%4s", (v<0 || v==9)?(""):sprintf(" %s) ", Guide[i+1]));
  }
  print "";
}
# 場札 カード表示
function dispField(who,kind,  n)
{
  for(n=1; n<=FIELDSIZE; ++n)
    printf("%4s", cardStringBy(n, who, kind));
}
function dispFieldSkip(who,  n)
{
  for(n=1; n<=FIELDSIZE; ++n)
    printf("%4s", (Field[who,BLACK,n]==-1)?(""):cardStringBy(n, who, BLACK));
}
# 場札 ガイド表示
function dispFieldGuide(  n)
{
  for(n=1; n<=FIELDSIZE; ++n) printf(" %s) ", NtoA[n]);
  print "";
}
function dispFieldGuideSkip(  n)
{
  for(n=1; n<=FIELDSIZE; ++n)
    printf("%4s", (Field[COM,BLACK,n]==-1)?(""):sprintf(" %s) ", NtoA[n]));
  print "";
}
```

```
# カードの表示
function cardString(who,kind,rank,  pos,v)
{
  if(rank<0) return "[  ]";
  if((v=getHands(who,kind,rank))==-1) return "";
  if(HIDE==1 && who==COM && kind==BLACK) return "[##]";

  pos=cnvSuit(who,kind)*13 +rank;
  return sprintf((v==9)?"*%s%s*":"[%s%s]",
      StrSuit[Cards["suit", pos]],
      StrRank[Cards["rank", pos]]);
}
function cardStringBy(pos, who,kind){
  return cardString(who,kind,Field[who,kind,pos]);
}
function cnvSuit(who,kind){ return who*2+kind; }
function dispSpace(num,  i){ for(i=0; i<num; ++i) printf("%4s", ""); }


## 当り判定
function judge()
{
  compare(YOU,COM);
  compare(COM,YOU);
}
function compare(who,another,  n,r,p, list,cnt)
{
  delete list;  cnt=0;
  for(n=1; n<=FIELDSIZE; ++n){
    r=Field[another,BLACK,n];
    if(r!=-1 && r==Field[who,RED,n]){  #RED->BLACK 当り (BLACK 取られる)
      setHands(who, RED, r, 9);
      remove(another, BLACK, r, n);
    }else
      list[cnt++]=r;
  }
  for(n=1; n<=FIELDSIZE; ++n){
    r=Field[who,RED,n];
    if(r!=-1 && getHands(who,RED,r)!=9 && find(list,r)<0)  #RED 取られる
      remove(who, RED, r, n);
  }
}
```

```
# 勝敗結果表示
function result(  y,c, s)
{
  print "¥n¥n ****** GAME OVER ******";

  HIDE=0;  ## COMのBLACK='守'をオープン表示
  display();

  c=noOpen(YOU,BLACK);  y=noOpen(COM,BLACK);
  if(y==0 && c>0)  s=" YOU WIN!";
  if(y>0 && c==0)  s=" YOU LOSE";
  if(y==0 && c==0) s=" ...DRAW ";
  printf("¥n%s (SCORE:%05d0)¥n¥n", s, ((FIELDSIZE-y)*100+c));
}

function allOpen()
{
  if(noOpen(YOU,BLACK)==0 || noOpen(COM,BLACK)==0) return 1;
  return 0;
}
function noOpen(who,kind,  n,q)
{
  q=0;
  for(n=1; n<=FIELDSIZE; ++n) if(Field[who,kind,n]!=-1) ++q;  # 残り有り
  return q;
}


## 場に出す(COM)
function com(kind,  n,r,v,handList,priority,cnt,i,j,p)
{
  if(kind==RED) printf("¥n>>>> COM");
  # '場'に残った手札をローテーション
  delete handList;
  cnt=0;
  for(r=0; r<13; ++r){
    n=getHands(COM,kind,r);
    if(1<=n && n<=FIELDSIZE) handList[cnt++]=r;
  }
  if(cnt>0){
    for(p=0; p<cnt; ++p){
```

```
      r=handList[p];  v=getHands(COM,kind,r);
      # 未作成なら候補リストを作る
      if(TryField[r,0]==0)
        for(n=1; n<=FIELDSIZE; ++n) TryField[r,++TryField[r,0]]=n;
      # 置いていた場所と'当り'を候補リストから取り除く
      for(i=1; i<=TryField[r,0]; ){
        n=TryField[r,i];
        if(n==v || (Field[COM,kind,n]!=-1 && getHandsBy(n,COM,kind)==9)){
          if(n==v) Field[COM,kind,n]=-1;
          for(j=i; j<TryField[r,0]; ++j) TryField[r,j]=TryField[r,j+1];
          --TryField[r,0];
        }else
          ++i;
      }
      priority[p]=p;
    }
    # 候補リストから１つ取り出して、'場'に置く
    sort(priority,cnt);
    for(p=0; p<cnt; ++p){
      r=handList[priority[p]];
      for(i=1; i<=TryField[r,0]; ++i){
        n=TryField[r,i];
        if(Field[COM,kind,n]==-1){ placement(COM,kind,r,n);  break; }
      }
    }
  }
  # '場'に手札から出す
  delete handList;
  cnt=0;
  for(r=0; r<13; ++r) if(getHands(COM,kind,r)==0) handList[cnt++]=r;
  rndRange(cnt);  # 作業用に rndList[] を使う
  p=0;  r=-1;
  for(n=1; n<=FIELDSIZE; ++n){
    if(r==-1) r=handList[rndList[p++]];
    if(Field[COM,kind,n]==-1){
      placement(COM,kind,r,n);
      r=-1;
    }
  }
}

function swap(tbl,p1,p2,    t){ t=tbl[p1];  tbl[p1]=tbl[p2];  tbl[p2]=t; }
```

```
function sort(tbl,cnt,  i,j)
{
  for(i=0; i<cnt-1; ++i)
    for(j=0; j<cnt; ++j)
      if(TryField[tbl[i],0]<TryField[tbl[j],0]) swap(tbl,i,j);
}
function rndRange(size,   i)
{
  # 作業用に rndList[] を使う
  delete rndList;
  for(i=0; i<size; ++i) rndList[i]=i;
  for(i=0; i<size; ++i) swap(rndList, rnd(size), i);
}
function rnd(N){ return int(N * rand()); }  ## 乱数


## 場に出す(YOU)
function you(kind)
{
  printf("¥n>>>> YOU");
  while(choice(kind))
   ;
}
function choice(kind,  z,x,y,w,what,where)
{
  do{
    (kind==RED)?display():setUpDisplay();
    printf("¥n Which?(1a-ke/z) >");  z=input();
    if(length(z)>1){ gsub(/,/," ",z);  split(z, x);
      for(y in x){
        if(length(x[y])>=2){
                what=substr(x[y],1,1);
          if(isInGuide(what)){          where=substr(x[y],2,1); }
          else{ what=substr(x[y],2,1);  where=substr(x[y],1,1); }
          put(kind,what,where);
        }
      }
    }
  }while(z!="z");
  if(noOpen(YOU,kind)==FIELDSIZE){
    if(decision("Ready")=="y"){ print "";  return 0; }  # 入力終了
  }else{
```

```
    printf("¥n ...Not Yet");  pushEnter();
  }
  print "";
  return 1;  # 入力続行
}
function isInGuide(what,  p)
{
  if(what2Rank(what)>=0) return 1;
  return 0;
}
function put(kind,what,where,  rank,pos,v)
{
  if((rank=what2Rank(what))<0) return;
  if((pos=where2Pos(where))<0) return;
  if(Field[YOU,kind,pos]!=-1 && getHandsBy(pos,YOU,kind)==9) return;
  v=getHands(YOU,kind,rank);
  if(0<=v && v<=FIELDSIZE){  # 使える'手札'なら...
    if(Field[YOU,kind,pos]!=-1)    # '場'が空いていなければ空けて...
      reset(YOU,kind,pos);
    if(v!=0) Field[YOU,kind,v]=-1; # 使用済みなら一旦戻して...
    placement(YOU,kind,rank,pos);  # '手札'を'場'に置く
  }
}
function find(list,val,  p)
{
  for(p in list) if(list[p]==val) return p;
  return -1;
}
function what2Rank(what,  p)
{
  if((p=find(Guide,what))>=0) return p-1;
  return -1;
}
function where2Pos(where,  n)
{
  if((n=find(NtoA,where))>=0) return n;
  return -1;
}

function remove(who,kind,rank,pos)
{
  setHands(who, kind, rank, -1);
```

```
  Field[who,kind,pos]=-1;
}
function reset(who,kind,pos)
{
  setHandsBy(pos, who, kind, 0);
  Field[who,kind,pos]=-1;
}
function placement(who,kind,rank,pos)
{
  setHands(who, kind, rank, pos);
  Field[who,kind,pos]=rank;
}

function getHands(who,kind,rank){
  return Hands[who,cnvSuit(who,kind),rank];
}
function getHandsBy(pos, who,kind){
  return getHands(who,kind,Field[who,kind,pos]);
}
function setHands(who,kind,rank,v){
  Hands[who,cnvSuit(who,kind),rank]=v;
}
function setHandsBy(pos, who,kind,v){
  setHands(who,kind,Field[who,kind,pos],v);
}

function input(  tmp){ tmp="";  getline tmp;  return tolower(tmp); }
function yorn(  yn){ do{ yn=input(); }while(yn!="y" && yn!="n");  return yn; }
function decision(msg){ printf(" %s?(y/n) >", msg);  return yorn(); }
function pushEnter(  ){ printf(" <PUSH ENTER>");  input(); }
```

(投稿: 2012年 1月31日)

# Excel±クリック数取器(カウンター) Ver1.0

Written By でび

## ■数取器って何だ？

「数取機（かずとりき）」という名称そのものが一般的には知られていませんね。
「日本野鳥の会の人が持っているアレ」とか「交通量調査で使っているカチカチカウンター」って何ていうの？、という質問もネット上では散見されます。

白状しますが、私も知りませんでした。
商品名は「数取機」って言うらしいですよ。
文具屋さんでチャンとしたものが1000円くらい。
最近は、100円均一ショップでも「それなりに使える」ものが売られているそうです。

で、フリーソフトとして配布されている「数取機」は「カウンター」などの名称が付けられていることが多いようです。

## ■カウンターソフト

この種のカウンターソフトを10種類くらい試してみました。
私個人の感性に過ぎませんが、それらの中では、くま778氏作、「カウンタ Ver.1.01」（http://www7b.biglobe.ne.jp/~kuma778/pgm/counter/counter.html）が最も良さそうでした。
観点は、「わかりやすい」＋「多機能」＋「現代的プログラミング言語で開発されている」ということです。

何よりも、「見出し語」を設定できて、CSVファイルで保存、読み込みが出来る点が優れていますし、流石に2011年の暮れの段階で、昔のソフトで動作環境として要求されていたVB6のランタイムを入れるのは、今更どうかなぁ・・・という気がしたわけです。
くま778氏作の「カウンタ Ver.1.01」は「.Net Framework 2.0」環境が必要なのですが、Windows VistaやWindows7には、元々.Net環境が入っていますからね。

## ■数取機全般で困ること

文具屋さんで売っている数取機や、カウンターソフトで、ちょいと困ったのが、2点。

【一つ目】
「間違って増やしちゃった時に、引き算をしたい！」ということ。
どうやら、事務機器として販売されているホンモノの「数取機」にも引き算ボタンは無いようです。

「慣れちゃえば、間違えないでしょ？」ってことなんでしょうか。

【二つ目】
「項目数は10個程度では全然足りない！」ということ。
駅前でのトラフィック調査をやっているパイプ椅子に座っているお兄さんたちは、複数の「数取機」を画板のようなものに固定して使っていますが、それでも10個くらいですよね。
だから、普通は10個もあれば充分なのでしょう。
キーボードに割り当てるとしたらテンキーで10個とか、[F1]～[F12]までで12個とかがUI的にもシンプルですしね。

でも、10個や12個では、AKBの総選挙では項目数が足りません。
テニミュの全キャスト（2ndシーズンを含む）からナンバーワンのイケメンを投票で決めるときにも困ります。

あと、30名を超える会員互選方式の選挙の場合も、困ります。
たとえば、マンション管理組合の委員とか、中学校の学級委員とかですね。
会員数約2,000名で、非立候補制の会員互選方式選挙の開票作業を、手作業で「正ちゃんマーク」でやったことがありますが、集計が大変でした。

その他、計量言語学の分野で、印刷媒体（新聞や雑誌など）で、特定の用語・用字の使用度数を調べたい場合に、便利かもしれません。
…というか、そのような頻度集計作業をする必要が出たので、仕方なく作った。

**■本マクロの良い点：**

○見出し語設定ができる。
○マウスクリックで、「加算」だけでなく「減算」もできる。
○見出し数の限度は、104万8575個まで設定可能（Excel12007以降）。
○合計も（セル中に自分で式を入れておけば）自動で出せる。
○データ入力が終わった時点で、カウント数の▽で「数値フィルタ」を選択すると、色々な絞り込み閲覧ができる。

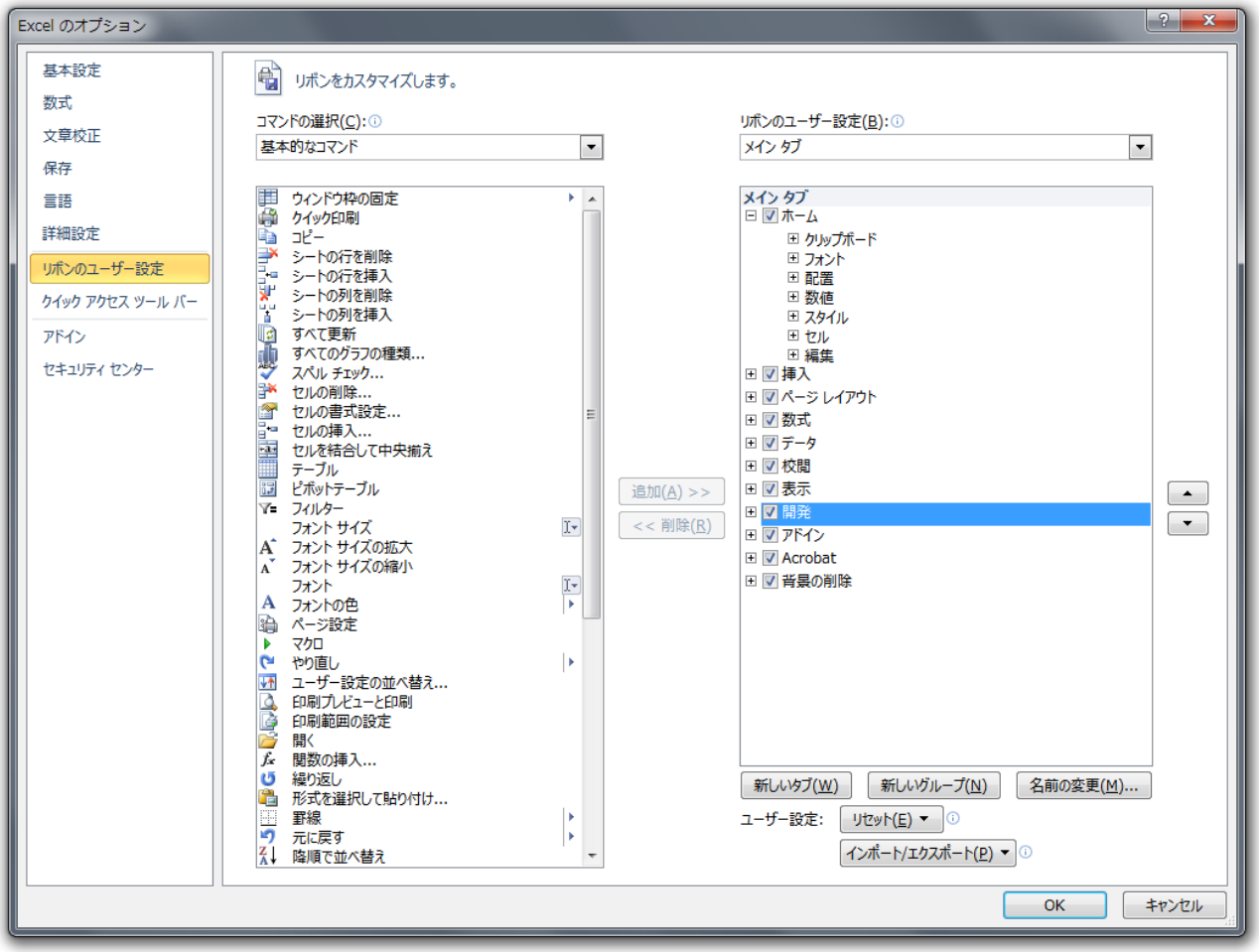
**■本マクロの悪い点：**

○マウスクリックが早すぎると、該当セルにカーソルが入ってしまう。
少し、「溜め」が必要なんですが、コレ、結構、苛つきます。
○キーボード入力での加算用ボタン、減算用ボタンは用意していない。
テンキーから入力とかの機能は実装していません。
候補数が106個を超えたら割り当てできないし。
Nキーは何番目の候補に相当するなんて、覚えきれないでしょ？

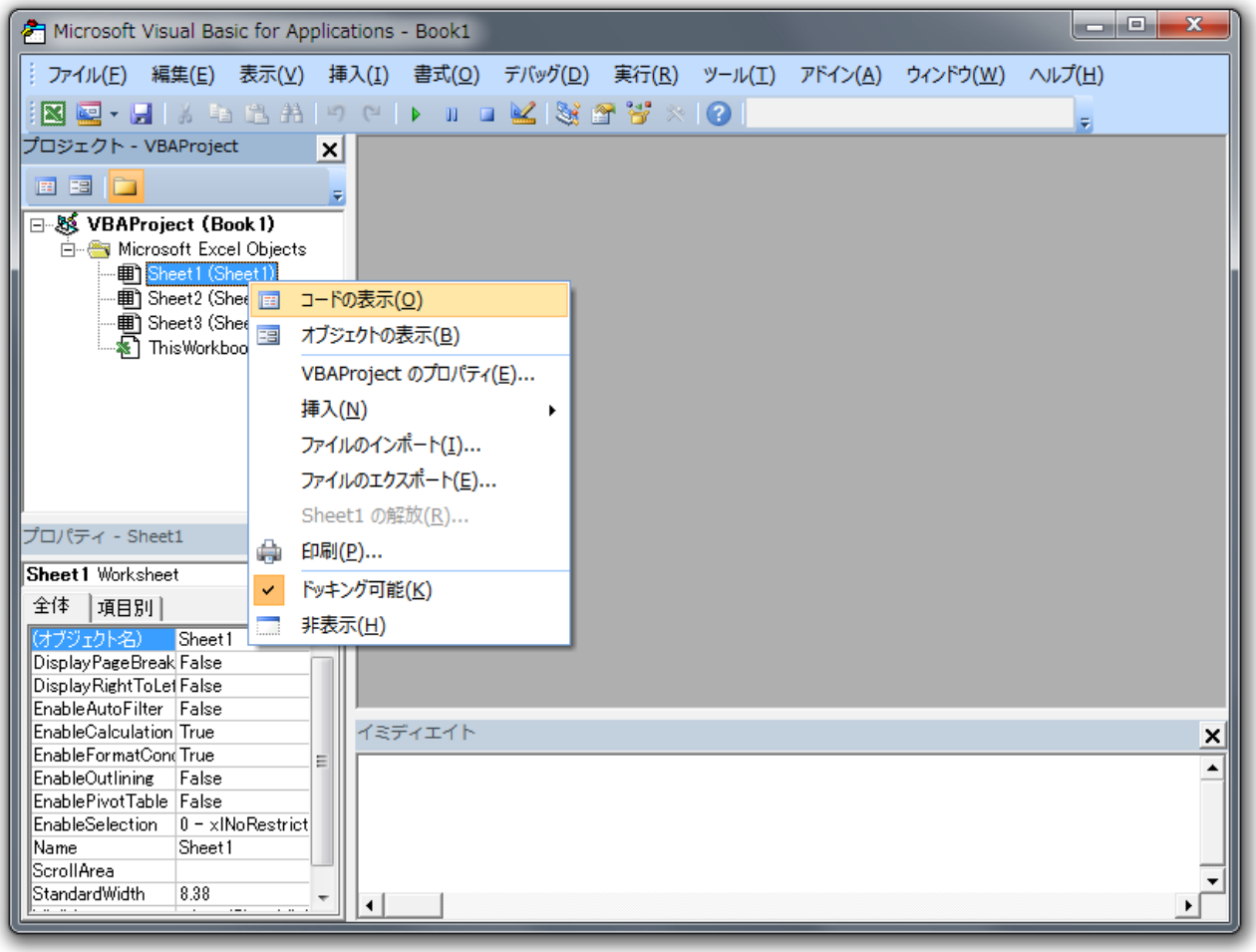
だいたい、こんな感じです。

## ■組み込み作業

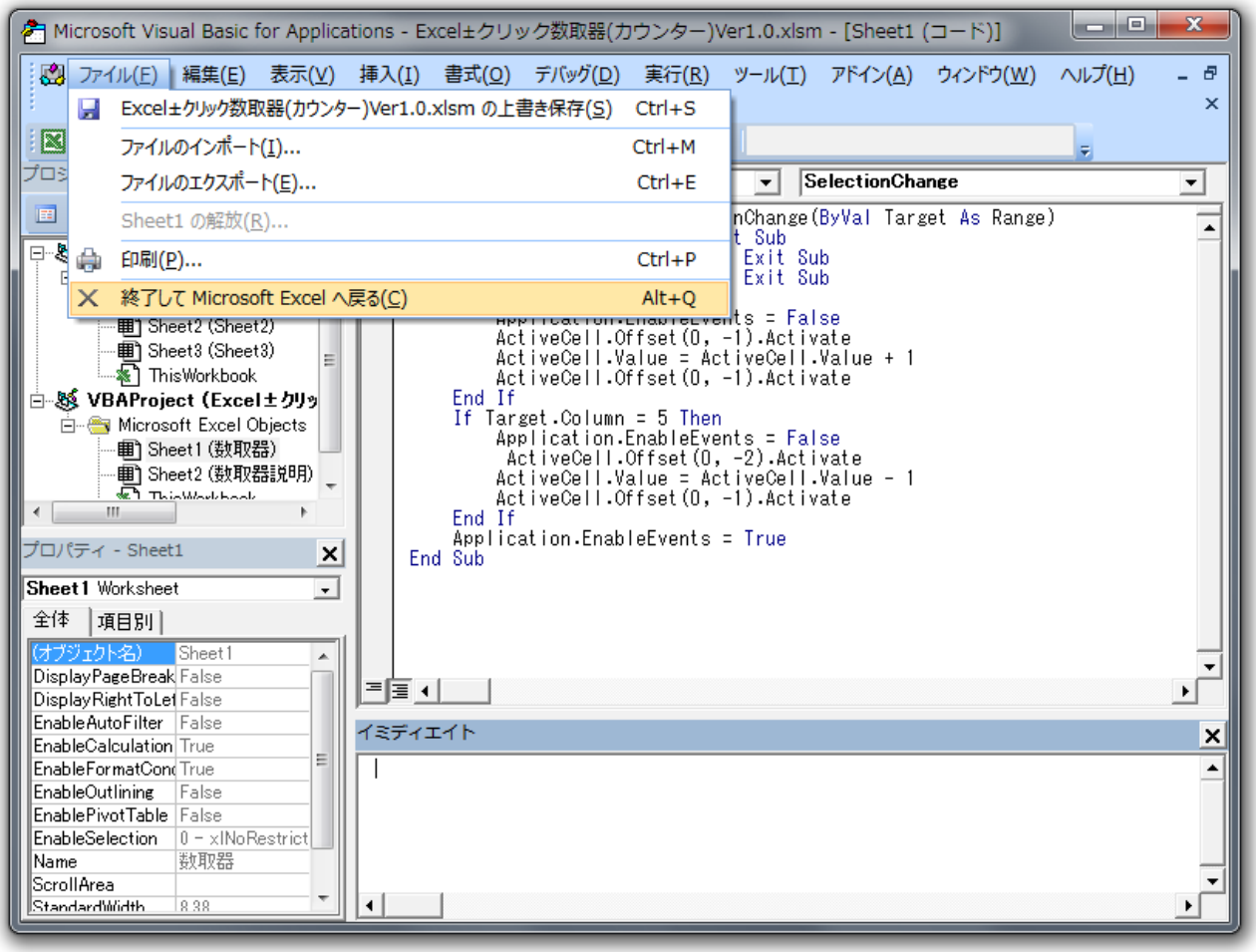
●[開発]タブの表示（Excel2010 の場合）
1.[ファイル]→画面左端の下の方に小さくある[オプション]から、「Excel オプション」ウィンドウを開く。
2.「Excel オプション」ウィンドウの画面左端で[リボンのユーザー設定]を選択。
3.画面右側に出た「メインタブ」で「開発」にチェックを入れて、OK で閉じる。




●マクロの貼り付け
0.普通のマクロのように、標準モジュールではありません。
1.[開発]タブを開き、一番左 にある[Visual Basic]ボタンを押す。
2.なんだかExcel とは様子が異なる画面が表示される。これが VBA Editor です。
3.VBA Editor 画面の左端にある「Sheet1(Sheet1)」の真上で[【右】クリック]して[コードの表示]。




4. そこに、下記コードを貼り付ける（行番号は入れないこと）。
5. [ファイル]→[終了して Microsoft Excel へ戻る]で VBA Editor を終了する。




●Excel の Sheet1 に見出し等の書き込みをする
1 行目は A 列～E 列まで、以下のような感じ。
No., カウント対象, カウント数, ＋, －

2 行目の A 列は 1 として、下方向にオートフィルで数値を増やす。
2 行目の D 列と E 列は下方向にオートフィルで＋やーをコピー。

●ファイルの保存
Excel2007 以降では、マクロ入りファイルは、拡張子が xlsm となっている
「Excel マクロ有効ブック」形式で保存することになっています。
初回の保存時に気をつけましょう。

■コード

```
01    Private Sub Worksheet_SelectionChange(ByVal Target As Range)
02        If Target.Row = 1 Then Exit Sub
03        If Target.Column >= 6 Then Exit Sub
04        If Target.Column <= 3 Then Exit Sub
```

```
05        If Target.Column = 4 Then
06            Application.EnableEvents = False
07            ActiveCell.Offset(0, -1).Activate
08            ActiveCell.Value = ActiveCell.Value + 1
09            ActiveCell.Offset(0, -1).Activate
10        End If
11        If Target.Column = 5 Then
12            Application.EnableEvents = False
13             ActiveCell.Offset(0, -2).Activate
14            ActiveCell.Value = ActiveCell.Value - 1
15            ActiveCell.Offset(0, -1).Activate
16        End If
17        Application.EnableEvents = True
18    End Sub
```

■コード解説

01行目
Worksheet_SelectionChangeは、ワークシート上でイベントが発生した時に呼び出される特殊なメソッドです。
それを、Private Subにして、マクロの実行ボタンなどから選択起動するのではなく、ワークシートを開いたときに自動で動くようにしてあります。

02行目
シート中の1行目は、「見出し」なので、マウスによるセル選択が1行目のセルだった場合は、なにもしない処理にしてあります。
この規制を上の方に入れておくことによって、以降の記述をスキップして、そのイベントについては無視して終了します。
以下でもそうですが、Row（行）やColumn（列）で指定する数字は、0から起算するのではなく、1から起算するのがExcel VBAでの流儀のようです。

03行目
シート中のF列目を含めて右側セルへの操作の場合は、なにもしない処理にしてあります。
この規制を上の方に入れておくことによって、以降の記述をスキップして、そのイベントについては無視して終了します。

04行目
シート中のC列目を含めて左側のセル（すなわちA～C列）への操作の場合は、なにもしない処理にしてあります。
この規制を上の方に入れておくことによって、ry)

05行目～10行目
　選択したセルがD列（＋）だった場合、アクティブセルを同じ行の1つ左のセル、C列（カウント数）に移し（07行目）、既に入っている数値に1を加算し（08行目）、加算終了後に、アクティブセルをもう1つ左のB列（カウント対象）に移し（09行目）ます。
　現在位置からのOffset指定はRow（行）,Column（列）の順に書きます。
　06行目は、07～09行目のセル遷移や加算操作そのものが、イベントとして認識されてしまうのを防ぐために、一時的にイベント認識を抑制する指定です。
　09行目で、加算終了後にアクティブセルを他のセルに移動させるのは、D列のセルが選択されたままだと、もう一度加算を行いたい場合に、2回目のイベント発生を取得できないため、敢えて移動させています。
　この操作はプログラム上の都合なので、ユーザには無駄な挙動にも思えますが、「選択セルを定位置へ戻す」という風に受け止めて頂くより仕方ありません。

11行目～16行目
　選択したセルがE列（－）だった場合の処理です。
　Column指定の数字が-2になっているのは、現在位置がD列ではなくE列だからですね？
　あとは、05行目～10行目と見比べれば、何をやっているのかは分かると思います。

17行目
　06行目と12行目の指定を解除しています。

## ■感想

　ユーザーフォームで一個ずつボタンを作るのは、面倒くさかったのです。
　ちょっと動作がトロいので、クリックするタイミングによって引っかかりを感じるのが難点ですが、セル選択をイベントとして取得することで、思惑通りのものが、一応、できました。よかったよかった。

## ■ライセンス



（投稿：2012年 2月 2日）

# MeCab でルビ振り

ムムリク

テキスト読み上げ機能などで日本語を読ませるとき、どうしても漢字や熟語によっては期待したような読み方をしてくれないことがあります。それならばきちんとしたルビを振っておくのが一番なのかと思うのですが、手作業ですべてのルビを振るのは大変です（読み上げ機能がルビに対応しているかどうかという問題もあるかもしれませんが）。

そこで、形態素解析の MeCab （めかぶ）を使ってルビ振りを行わせてみることにしました。MeCab と Ruby を使って処理してみます。

## MeCab のインストール

Windows 用にはバイナリーパッケージが用意されているので、そちらをダウンロードします（http://mecab.sourceforge.net/#download）。2012 年 1 月現在での最新バージョンは 0.991 でした。インストールするとデスクトップに「 MeCab 」アイコンができるので実行するとコンソールが開きます。ここに日本語文章を入力してリターンキーを押すと解析結果が表示されます。きちんと動いていることを確かめておきましょう。

スタートメニューにある MeCab のフォルダには辞書ファイルの文字コードを変換するメニューもあるので、必要に応じて手軽に変換できます。

## MeCab-Ruby バインディング

さまざまなスクリプト言語などで使用するためのバインディングは別に用意されています（ダウンロードページから mecab-ruby-x.xxx.tar.gz をダウンロード：x.xxx はバージョン番号）。しかし、Windows ですぐに使える状態にはなっていないため、自分で make する必要があります。また、説明されている手順通りに行ってもうまくいかなかったり、コンパイラを用意するという必要もあり、Windows ではやや使いにくい印象もあるかもしれません。

ウェブを検索しているとそのための対策としていろいろ行われているようです。libmecab.dll ファイルを直接扱う方法や、IO::popen を利用する方法などが紹介されています。（http://www.freia.jp/taka/blog/758/ が詳しいです）

情報を参考に試してみたところ一応 make して使えているようなので、参考までにその手順を示します。（この手順で問題がないのかどうかは未確認です）

なお、事前にフリーで使えるマイクロソフトの Visual C++ 2010 Express （http://www.microsoft.com/japan/msdn/vstudio/express/）などをインストールしてコンパイルできる環境を用意しておきます。

1. mecab-ruby バインディングのソースファイルをダウンロードする。

2. アーカイブを展開して extconf.rb を編集する。

```
----- extconf.rb 編集前 ----
require 'mkmf'

mecab_config = with_config('mecab-config', 'mecab-config')
use_mecab_config = enable_config('mecab-config')

`mecab-config --libs-only-l`.chomp.split.each { | lib |
  have_library(lib)
}

$CFLAGS += ' ' + `#{mecab_config} --cflags`.chomp

have_header('mecab.h') && create_makefile('MeCab')
----------------------------

----- extconf.rb 編集後 ----
require 'mkmf'

#mecab_config = with_config('mecab-config', 'mecab-config')
#use_mecab_config = enable_config('mecab-config')

#`mecab-config --libs-only-l`.chomp.split.each { | lib |
#  have_library(lib)
#}

#$CFLAGS += ' ' + `#{mecab_config} --cflags`.chomp
$CFLAGS += ' -IC:/PROGRA~2/MeCab/sdk'
$LOCAL_LIBS += ' libmecab.lib'
$LIBPATH << 'C:/PROGRA~2/MeCab/sdk'

have_header('mecab.h') && create_makefile('MeCab')
----------------------------
```

MeCab を標準のインストール先にそのまま入れた場合、C:¥Program Files (x86)¥MeCab 以下にインストールされますが、半角スペースを含んだファイル名をパスとして指定する

とスペースのところでそれぞれが分断されてしまい、正しく認識されません。そのため、PROGRA~2 のように短縮名を指定する必要があります。
# でコメントアウトした元もとの部分は面倒であれば削除してもかまいません。

### 3. Makefile を作る

Ruby にパスの通ったコンソールを開き、mecab-ruby-0.991 のフォルダに移動します。
```
>ruby extconf.rb
```

として Makefile を作ります。

### 4. Makefile を編集して make する

Makefile の中にある Ruby のパスなども多くの場合 C:¥Program Files (x86) 以下にあります。そこで、この部分も先ほどの extconf.rb の場合と同様に修正をします。
編集としてはこれだけなので Visual Studio 2010 などのコンソールを開き、Makefile のあるフォルダに移動してから nmake を実行してみると「ruby.h のビルド方法が指定されていません」と表示してエラーになります。そこで、Makefile の最下行にある $(OBJS): の行をコメントアウトしておきます。

```
#$(OBJS): {.;$(VPATH)}$(hdrdir)/ruby.h {.;$(VPATH)}$(hdrdir)/ruby/defines.h $(arch_hdrdir)/ruby/config.h
```

再度 nmake するとしばらくして MeCab.so ファイルが作成されます。現状では問題なく使えているようですが、なんらかの問題が内在しているのかどうかはわからないので、そのつもりでご使用ください。

## Ruby から使う

実際に Ruby から使ってみます。ここでは 1.9.3-p0 を使っています。1.8.7 系は近い将来メンテナンスが終了するので、できるだけ 1.9 系を使うことをおすすめします。
手軽に MeCab を使うのであれば、実行するプログラムと同じフォルダに先ほど作成した MeCab.so と C:¥Program Files (x86)¥MeCab¥bin にある libmecab.dll もコピーしておきます。（本来的には Ruby のインストールフォルダの /lib や /bin 以下に置くべきですが)
mecab-ruby バインディングに添付されている test.rb をコピーして試してみます。test.rb は UTF-8 で書かれているので、スタートメニューの MeCab フォルダから「Recompile UTF-8 Dictionary 」を選択して、辞書の文字コードを UTF-8 に変更しておきます。
また、require 'MeCab' を require './MeCab' と修正しておきます。

```
>ruby test.rb
```

実行するとコンソールには白い四角が多数表示され、文字がきちんと表示されません。
Ruby は 1.9 系から文字エンコードについてそれまでと少し変更があり、文字列はあくまでも文字列であるとして処理されているそうです。文字列は自分の文字コードを知っています。一方で MeCab から返される文字列は 1.8 のころと同じような扱いであるバイト列として返してくるようです。
Ruby からは UTF-8 で送られた文字列が、MeCab で処理されて戻ってくるときには ASCII-8BIT で返ってきます。このため Ruby ではそのまま ASCII-8BIT として処理してしまうので文字列が正しく表示されません。
そこで、強制的に文字エンコードを変更する force_encoding() を使えば正しく表示されるようになります。
test.rb の先頭のほうにある

```
puts tagger.parse(sentence)
```

を、

```
t = tagger.parse(sentence)
puts t.force_encoding("UTF-8")
```

として実行してみます。

```
--------------------------
>ruby test.rb
0.991
太郎　　名詞,固有名詞,人名,名,*,*,太郎,タロウ,タロー
は　　助詞,係助詞,*,*,*,*,は,ハ,ワ
この　　連体詞,*,*,*,*,*,この,コノ,コノ
本　　名詞,一般,*,*,*,*,本,ホン,ホン
を　　助詞,格助詞,一般,*,*,*,を,ヲ,ヲ
二　　名詞,数,*,*,*,*,二,ニ,ニ
郎　　名詞,一般,*,*,*,*,郎,ロウ,ロー
を　　助詞,格助詞,一般,*,*,*,を,ヲ,ヲ
見　　動詞,自立,*,*,一段,連用形,見る,ミ,ミ
た　　助動詞,*,*,*,特殊・タ,基本形,た,タ,タ
女性　　名詞,一般,*,*,*,*,女性,ジョセイ,ジョセイ
に　　助詞,格助詞,一般,*,*,*,に,ニ,ニ
渡し　　動詞,自立,*,*,五段・サ行,連用形,渡す,ワタシ,ワタシ
た　　助動詞,*,*,*,特殊・タ,基本形,た,タ,タ
。　　記号,句点,*,*,*,*,。,。,。
```

```
EOS
----------------------------
```

今度は正しく表示されました。続く他の部分についても同様に force_encoding("UTF-8") を使うことで正しく表示されます。

しかし、いちいち force_encoding するのも大変です。本来はバインディングのほうで修正ができるのが望ましいのでしょうが、ひとまず補完的な措置を取ることにします。

次のようなファイルを用意します。

```
------- mecabex.rb ---------
# encoding: UTF-8
require './MeCab'
module MeCab
   class Tagger
      def parseEx(arg)
         $enc = arg.encoding if arg.instance_of?(String)
         result = self.parse(arg)
         result.force_encoding($enc) if arg.instance_of?(String)
      end
      def parseToNodeEx(arg)
         $enc = arg.encoding if arg.instance_of?(String)
         result = self.parseToNode(arg)
      end
   end
end
----------------------------
```

test.rb に require './mecabex' を追加し、tagger.parse() となっている箇所を tagger.parseEx() と修正します。ここでは MeCab::Tagger の parse と parseToNode だけ補完メソッドを用意しましたが、必要に応じて他の部分にも用意するとよいかもしれません。

## MeCab を使ってルビを振る

ルビを振るためには Tagger::parseToNode を使用したほうが便利です。解析して分解されたそれぞれを順に処理できます。

ただ、Node で取得できる情報の feature は多数の情報のまとまりであるため、品詞や読み方などだけを取り出すことができません。そこで、さきほどの mecabex.rb に追加することにします。class Node ... end までを追加します。

```
------- mecabex.rb ---------
```

```
module MeCab

   class Node
      def surfaceEx
         self.surface.force_encoding($enc)
      end
      def featureEx
         fex = self.feature.force_encoding($enc)
         fexa = fex.split(/,/)
         @pos = fexa[0]
         @pos1 = fexa[1]
         @pos2 = fexa[2]
         @pos3 = fexa[3]
         @dec1 = fexa[4]
         @dec2 = fexa[5]
         @root = fexa[6]
         @reading = fexa[7]
         @sound = fexa[8]
         fex
      end
      attr_reader :pos, :pos1, :pos2, :pos3, :dec1, :dec2, :root, :reading,
:sound
   end

end
----------------------------
```

surface は分割された文字そのもの、feature では品詞分類や原型、読み、発音などを示します。（いささかスマートさに欠けているのが難ですが）

ついで、次のようなサンプルを用意します。

```
------- rubysample.rb ------
 1:# encoding: utf-8
 2:
 3:require './mecabex'
 4:
 5:str = "太郎はこの本を二郎を見た女性に渡した。"
 6:KANA = Regexp.new("[ぁ-ﾞｧ-ヶー]")
 7:
 8:model = MeCab::Model.new("")
 9:tagger = model.createTagger()
```

```
10:n = tagger.parseToNodeEx(str)
11:n = n.next
12:result = ""
13:while n.next do
14:    ns = n.surfaceEx
15:    n.featureEx
16:    if /¥A#{KANA}+¥z/ =~ ns or "記号" == n.pos
17:       str2 = ns
18:    else
19:       str2 = "#{ns}《#{n.reading}》"
20:    end
21:    result = result + str2
22:    n = n.next
23:end
24:puts result
----------------------------
```

11: で一度 node を進めているのは、node でははじまりと終りを示す特別の node が必ずはいるため、それをあらかじめスキップするために進めています。13: では次の node がある限り繰り返すようにしているため、終りの node では次がないので、そこで終了します。( 11: をなくし、13: の n.next を n と変更して実行すると違いがよくわかります)

14: 15: での主な目的は文字エンコードの処理です。同時に 15: では細かな情報への分割とアクセスを準備します。

16: では「ひらがな・カタカナ」だけであるか、品詞として「記号」である場合にはルビ振りはせず ( 17: ) 、それ以外、つまり漢字を含んでいるであろう時はルビを振る ( 19: ) というようにしています。ルビの表記は青空文庫で使われる形式にしています。

実行すると次のような結果が表示されます。

```
----------------------------
太郎《タロウ》はこの本《ホン》を二《ニ》郎《ロウ》を見《ミ》た女性《ジョセイ》に渡し《ワタシ》た。
----------------------------
```

一応それらしくなりました。

ただ、「ジロウ」であって欲しいはずの「二郎」が「ニロウ」となっていますし、「ニ」「ロウ」と分かれてしまっているのも、気になります。さらに、「渡し《ワタシ》」と送り仮名まで一緒になってしまっているのも嬉しくない感じがします。

読み方については辞書登録を充実させることで、ある程度対処することができるかもしれません。送り仮名などが一緒になる問題は漢字部分とかな部分を分割するように処理を修正することで対処できるでしょう。

また、ここでは文章をプログラム内に用意しましたが、テキストファイルから読み込むようにしたほうが、実際的には便利です。

完全に正しい読み方のルビを自動的に振るには限界がありそうではありますが、いずれにしても最終的に人の手によって確認する作業は必要でしょうから、ある程度の精度でルビが振れることで作業は軽減されるのではないでしょうか。

よりきちんと処理しようと思えば、もう少しプログラムを修正する必要はあるでしょうが、比較的簡単に扱えるということがわかりました。

（投稿：2012年 2月 3日）

# Pythonの文法

**TSNETスクリプト通信版　草稿　第６回**

**2012.2.3 機械伯爵**

### 6-2-2-2-3-2-1-2 アルファベット用のメソッド

初期のPythonでは関数でしたが、とにかく最初から存在した伝統的なメソッドです。

**●英文の文字列全体を変換**
大文字／小文字の変換ですが、ASCII以外のUnicode文字もちゃんと変換してくれるようです

**■全てを小文字に**
**書式：S0.lower() ⇒ S1**
**動作：文字列中のアルファベットを全て小文字に**

**■全てを大文字に**
**書式：S0.upper() ⇒ S1**
**動作：文字列中のアルファベットを全て大文字に**

```
>>> s0 = "aBcDe ａＢｃＤｅ"
>>> s0.lower()
'abcde ａｂｃｄｅ'
>>> s0.upper()
'ABCDE ＡＢＣＤＥ'
```

**●最初を大文字、あとは小文字に強制変換**
次は、英文書式の定番である、頭文字のみ大文字に変換するメソッドです。勿論、マルチバイト文字にもばっちし対応。'title'と'capitalize'の使い分けは「語頭」と「文頭」です。

**■語頭を大文字に、あとは小文字に強制変換**
**書式：S0.title() ⇒ S1**
**動作：それぞれの単語に対して全て頭大体小変換を行う**

**■語頭を大文字に、あとは小文字に強制変換**
**書式：S0.capitalize() ⇒ S1**
**動作：文頭のみ大文字、あとは小文字に強制変換を行う**

実行コードを見て、その違いを確かめてみて下さい。

```
>>> "aBcDeFg".title()
'Abcdefg'
>>> "aBcDeFg".capitalize()
'Abcdefg'
>>> "abc abc abc".title()        # 'title'は語頭
'Abc Abc Abc'
>>> "abc abc abc".capitalize()   # 'capitalize'は文頭
'Abc abc abc'
```

```
>>> "abc ａｂｃ abc".title()          # マルチバイトでも完全アルファベット扱い
'Abc ａｂｃ abc'
>>> "ａｂｃ abc".capitalize()
'Ａｂｃ abc'
>>> "abc ａｂｃ".capitalize()
'Abc ａｂｃ'
>>> "ａｂｃ abc".title()
'Ａｂｃ Abc'
>>> "abc ａｂｃ".title()
'Abc Ａｂｃ'
>>> "abc あいう abc".title()          # アルファベット以外は区切り扱い
'Abc あいう Abc'
>>> "abc ａｂｃ abc".title()          # アルファベットは連続とみなす
'Abc ａｂｃ abc'
>>> "あいう abc".capitalize()         # 文頭でないので反応しない
'あいう abc'
```

**●大文字を小文字に、小文字を大文字に**

コレが何に使えるのか、正直わかりませんが、とりあえずあります。

**■大文字を小文字に、小文字を大文字に変換する。**

**書式：S0.swapcase() ⇒ S1**

**動作：文字列中の大文字を小文字に、小文字を大文字に変換する**

```
>>> "aBc Abc ａＢｃ ＡｂＣ".swapcase()
'AbC aBC ＡｂＣ ａＢｃ'
```

### 6-2-2-2-3-2-1-3 文字種判別用メソッド

文字種を判別するメソッド群ですが、どうも help の説明やマニュアルの説明と食い違いがあるように思えます（help やマニュアルでは「最低一文字でも"there is at least one character in S"」と書いてありますが、実際には指定された文字種以外の文字が混入している場合は全て'False'が返ります(ver.3.2.1 現在)

全てのメソッドで、Unicode 文字も同様に扱われます。

**■アルファベットのみ**

**書式：S0.isalpha() ⇒ bool**

**動作：文字列"S0"の文字が全てアルファベットの場合は'True'が返る。それ以外は'False'が返る**

**■数字のみ（３つのメソッドの相違については不明）**

**書式１：S0.isdecimal() ⇒ bool**

**書式２：S0.isdigit() ⇒ bool**

**書式３：S0.isnumelic() ⇒ bool**

**動作：文字列"S0"の文字が全て数字の場合は'True'が返る。それ以外は'False'が返る**

**■アルファベットか数字**

**書式：S0.isalnum() ⇒ bool**

**動作：文字列"S0"の文字が全てアルファベットか数字の場合は'True'が返る。それ以外は'False'が返る**

**■識別子として使用可能な文字列か**

**書式：S0.isidentifier() ⇒ bool**

**動作：文字列"S0"が識別子として使用可能ならば'True'を返す（ただし予約語として利用されている文字列であっても'True'を返すので、形式のみのチェックらしい）**

**■小文字かどうか**

**書式：S0.islower() ⇒ bool**

**動作：文字列"S0"に含まれているアルファベット文字が全て小文字なら'True'を返す**

**■大文字かどうか**

**書式：S0.isupper() ⇒ bool**

**動作：文字列"S0"に含まれているアルファベット文字が全て大文字なら'True'を返す**

**■表示可能かどうか**

**書式：S0.isprintable() ⇒ bool**

**動作：文字列"S0"にエスケープ文字などの表示不可能な文字が含まれているいなければ'True'を返す**

**■語頭が大文字でその他が小文字（タイトルスタイル）かどうか**

**書式：S0.istitle() ⇒ bool**

**動作：文字列"S0"に含まれるアルファベットのみで構成される語の語頭が大文字で、それ以外が小文字であるならば'True'を返す。数字などのアルファベット以外の文字は、このスタイルの単語の前後につく場合には'True'、単語の中に挟まれている場合は'False'を返す**

**■空白文字であるかどうか**

**書式：S0.isspace() ⇒ bool**

**動作：文字列"S0"が全てスペース、タブ、改行などの空白文字で構成されている場合は'True'、そうでなければ'False'を返す**

**※出力テストに関しては、あまりにも退屈なので省略します**

### 6-2-2-2-3-2-1-4 検索／置換用メソッド

検索、置換等に関するメソッド群です。正規表現に関しては、're'モジュールを使用しますが、簡単なものに関しては文字列のメソッドとして実装されています。

**●文頭、文末のフォーマットのテスト**

文字列が指定した文字列（文字、でも勿論可）で始まったり終わっているかを調べるメソッドです(名前はおそらく、"START of String WITH"と"END of String WITH"と思われる)

**■指定した文字列から始まっているか**

**書式：S0.startswith("s0"[,i1[,i2]]) ⇒ bool**

**動作：文字列"S0"が、引数である文字列"s0"から始まっているならば'True'を返す。"i1"整数オプションはチェックを始める位置を、"i2"整数オプションはチェックを修了する位置を表す**

**■指定した文字で終わっているか**

**書式：S0.endswith("s0"[,i1[,i2]]) ⇒ bool**

**動作：文字列"S0"が、引数である文字列"s0"で終わっているならば'True'を返す。"i1"整数オプションはチェックを始める位置を、"i2"整数オプションはチェックを修了する位置を表す**

```
>>> "abc def ghi".startswith("abc")
True
>>> "abc def ghi".startswith("abc",1)
False
>>> "abc def ghi".startswith("def")
False
>>> "abc def ghi".startswith("def",4)
True
>>> "abc def ghi".startswith("def",4,5)# 範囲が狭い
False
>>> "abc def ghi".endswith("ghi")
True
>>> "abc def ghi".endswith("def",0,7)
True
>>> "abc def ghi".endswith("def",6,7)  # 範囲が狭い
False
```

※正直、'startswith'メソッドの'end'オプション(="i1")と、'endswith'メソッドの'starts'オプション(="i2")は意味が無いようですが、多分フォーマットを揃えてあるのだと思われます

## ●検索してヒット位置を返す

### ■左から検索

書式：S0.find(k0) ⇒ int

動作：文字列"S0"を左から検索し、"k0"が最初に見つかった位置の先頭文字の位置を整数値で返す。見つからなかった場合には、'int'型の'-1'を返す

### ■右から検索

書式：S0.rfind(k0) ⇒ int

動作：文字列"S0"を右から検索し、"k0"が最初に見つかった位置の先頭文字の位置を整数値で返す。見つからなかった場合には、'int'型の'-1'を返す

### ■右から検索（２）

書式：S0.rindex(k0) ⇒ int

動作：文字列"S0"を右から検索し、"k0"が最初に見つかった位置の先頭文字の位置を整数値で返す。見つからなかった場合には、順序列コレクション共有メソッド'index'と同じく、'ValueError'を返す

```
>>> "abc def ghi def abc".find("def")
4
>>> "abc def ghi def abc".rfind("def")
12
>>> "abc def ghi def abc".find("xxx")
-1
>>> "abc def ghi def abc".rfind("xxx")
-1
>>> "abc def ghi def abc".index("def")
4
>>> "abc def ghi def abc".rindex("def")
12
>>> "abc def ghi def abc".index("xxx")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
>>> "abc def ghi def abc".rindex("xxx")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

**※'find','rfind'と'index','rindex'メソッドの違いは、ターゲットの文字列が見つからなかった場合に、'-1'を返す('find'系)か、'ValueError'を返す('index'系)かの違いです。処理を中断させたくないなら'find'系、例外をキャッチしたいのなら'index'系、と使い分けられます**

**●文字列の置換**

簡易置換メソッドです。正規表現を使うまでもない大雑把な文字列置換は、コレで行えます

**■文字列の置換**

**書式：S0.replace(rs0, rs1) ⇒ str**

**動作：文字列"S0"の中に含まれる全ての"rs0"を"rs1"に置換した文字列を返すする。"rs0"に合致する文字列が見つからずとも、元の文字列をそのまま返し、エラーなどは一切出さない**

```
>>> "abc def ghi def abc".replace("def","xxx")
'abc xxx ghi xxx abc'
>>> "abc def ghi def abc".replace("xxx","def")       # 別にエラーは出ないが置換もされない
'abc def ghi def abc'
>>> "abc def ghi ｄｅｆ abc".replace("def","xxx")
'abc xxx ghi ｄｅｆ abc'
```

### 6-2-2-2-3-2-1-5 連結／分割／分離用メソッド

文字列の連結、分割／分離のためのメソッド群です。

**●連結メソッド**

Python の文字列連結メソッド'join'は、他言語のユーザから「気持ちが悪い」と評判（苦笑）のメソッドです。セパレータ文字列のメソッドとして文字列のタプルやリストなどのコレクションを引数に取る方法は、確かに奇妙です。文字を操作する関数を廃した結果なのですが、その賛否はともかく、使い方を紹介します。

**■文字列を連結**

**書式：SEP.join(C) ⇒ str**

**動作：コレクション"C"のアイテム（辞書'dict'の場合はキー）に含まれる文字列を"SEP"で連結して返す。なお、コレクションは文字列をアイテムとするリスト、タプルの他、辞書や集合('set')などでも可能。また'str'自身も「文字列のコレクション」なので使用可能**

```
>>> ":".join(("abc","def","ghi"))       # 普通の連結その１
'abc:def:ghi'
>>> ":".join(["abc","def","ghi"])       # 普通の連結その２
'abc:def:ghi'
>>> ":".join({"abc","def","ghi"})       # 集合の連結（順不同）
'abc:ghi:def'
>>> ":".join({"abc":100,"def":200,"ghi":300}) # 辞書のキーの連結（順不同）
'abc:ghi:def'
>>> ":".join({100:"abc",200:"def",300:"ghi"}) # キーが文字列以外なら、当然エラー
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: sequence item 0: expected str instance, int found
>>> ":".join({100:"abc",200:"def",300:"ghi"}.values())      # 値を連結するなら…
'def:ghi:abc'
>>> ":".join("abcdefghi")         # 文字列は……れ、連結？
'a:b:c:d:e:f:g:h:i'
```

```
>>> "".join(("abc","def","ghi")) # セパレータは空文字列
'abcdefghi'
```

**※見てのとおり、文字列そのものに'join'を適用すると「一文字一文字がアイテムのシーケンス」として扱うので、結果的には文字を"SEP"で分離してしまう結果になります。なお、"SEP"を空文字列''とすれば、セパレータ無しに普通に連結できます**

**●分割メソッド**

文字列を、特定のセパレータ文字列で分割します。文字列を検索してヒットする全てのセパレータの箇所で分割します。

**■文字列を分割する**

**書式１：S0.split(SEP) ⇒ L0**
**書式２：S0.rsplit(SEP) ⇒ L0**
**動作：セパレータ文字列"SEP"を検索して、そこで文字列を分割したリスト"L0"を返す。'split'はセパレータを左から検索し、'rsplit'は右から検索する。なお、"SEP"を省略した場合は、全ての空白文字（スペース、タブ、改行）をセパレータとみなす**

**■文字列を改行で分割する**

**書式：S0.splitlines() ⇒ L0**
**動作：複数行ある文字列を、改行記号で区切って分割したリストを返す**

```
>>> "abc:def:ghi".split(":")
['abc', 'def', 'ghi']
>>> "abc:def:ghi".rsplit(":")           # 一文字では違いは無し
['abc', 'def', 'ghi']
>>> "abc:@:@:def:@:@:ghi".split(":@:")
['abc', '@:def', '@:ghi']
>>> "abc:@:@:def:@:@:ghi".rsplit(":@:")# 複数文字セパレータで違いが出る
['abc:@', 'def:@', 'ghi']
>>> "abc\ndef\nghi".splitlines() # 改行で分割
['abc', 'def', 'ghi']
>>> "abc\ndef\nghi".split()             # 一見、似ているが…
['abc', 'def', 'ghi']
>>> "abc def\tghi\njkl".split()         # ホワイトスペース全てで分割
['abc', 'def', 'ghi', 'jkl']
>>> "abc def\tghi\njkl".splitlines()    # 改行だけ
['abc def\tghi', 'jkl']
>>> "abc def\tghi\njkl".split("\n")
['abc def\tghi', 'jkl']
```

**※'split'と'rsplit'は、セパレータが一文字の場合は違いがありませんが、複数文字がセパレータになった場合には、上記のように検索順序によって結果が異なる場合があります。'splitlines()'は'split("¥n")'の省略形と考えられます**

**●分離メソッド**

分離メソッドは、分割メソッドに似ていますが、文字を分けるのは一回だけです。また、分割メソッドではセパレータ文字列は除かれますが、分離メソッドではセパレータ文字列がその位置に残ります

**■文字列を分離する**

**書式１：S0.partition(SEP) ⇒ L0**
**書式２：S0.rpartition(SEP) ⇒ L0**

**動作：文字列"S0"をセパレータ文字列"SEP"の位置で分離する。リスト'L0'は["SEPの前の文字列","SEP","SEPの後の文字列"]となる。'partition'はセパレータ文字列を左から検索し、'rpartition'は右から検索する**

```
>>> "a:b:c:d:e".partition(":")
('a', ':', 'b:c:d:e')
>>> "a:b:c:d:e".rpartition(":")
('a:b:c:d', ':', 'e')
```

### 6-2-2-2-3-2-1-6 整形用メソッド

余分な文字を削ったり、文字列の位置を調整したりするメソッド群です

#### ●左右の文字を削除

#### ■左右にある特定の文字を消去

**書式：S0.strip(CL) ⇒ S1**
**動作：文字列"CL"に含まれる文字が、文字列"S0"の左右にあった場合、それを除去した文字列"S1"を返す。"CL"は文字通り「文字の並び」として捉えられ、並び(シーケンス)ではなく集まり(セット)とみなされる。つまり"CL"が"#$@"とあれば、"#$@"という一塊の文字列ではなく"#"と"$"と"@"というセットとして考えられる**

```
>>> "*abcdef***".strip("*")              # 個数関係なし
'abcdef'
>>> "*abcdef***".strip("**")             # ばらばらに判断
'abcdef'
>>> "*abcdef***".strip("*$")             # 「並び」ではない
'abcdef'
>>> "$*abcdef$***".strip("*$")           # 「集まり」とみなされる
'abcdef'
>>> "$*@@abcdef$*@**".strip("*$@")
'abcdef'
>>> "***abc*def***".strip("*")           # 両端でなければ無視
'abc*def'
```

**※以下に挙げる'lstrip','rstrip'は、'strip'の「左端のみバージョン」と「右端のみバージョン」です**

#### ■左にある特定の文字を消去

**書式：S0.lstrip(CL) ⇒ S1**
**動作："CL"に指定された文字が"S0"の左端にあれば除去**

#### ■右にある特定の文字を消去

**書式：S0.rstrip(CL) ⇒ S1**
**動作："CL"に指定された文字が"S0"の右端にあれば除去**

```
>>> "***abc*def***".strip("*")
'abc*def'
>>> "***abc*def***".lstrip("*")          # 左端のみ除去
'abc*def***'
>>> "***abc*def***".rstrip("*")          # 右端のみ除去
'***abc*def'
```

## ●位置調整

文字幅を指定し、そのすき間を空白か指定のパッディング文字で埋めます。

### ■中央位置に調整

**書式：S0.center(i0[, c1]) ⇒ S1**

**動作：文字列"S0"を、整数"i0"で指定された文字幅の中央に位置させる。文字幅が"S0"より小さい場合は無視される。通常、空白文字でパッディングされるが、オプションの"c1"で指定された場合は、その文字が使用される。"c1"は、一文字のみの文字列とする。奇数偶数の関係でちょうど真ん中に位置できない場合は、以下のように処理される**

**文字数が偶数で文字幅が奇数⇒右寄り（右のパッディング文字が一つ少ない）**
**文字数が奇数で文字幅が偶数⇒左寄り（左のパッディング文字が一つ少ない）**
**おそらく、次のような計算をしているものと思われる……**
**右のパッディング文字数＝文字幅÷２（切捨て）－文字数÷２（切捨て）**
**左のパッディング文字数＝文字幅－（文字数＋右のパッディング文字数）**

### ■左端位置に調整

**書式：S0.ljust(l[, C]) ⇒ S1**

**動作：文字列"S0"を、整数"l"で指定された文字幅の左端に位置させる。文字幅が"S0"より小さい場合は無視される。通常、空白文字でパッディングされるが、オプションの"C"で指定された場合は、その文字が使用される。"C"は、一文字のみの文字列とする**

### ■左端位置に調整

**書式：S0.rjust(i0[, c1]) ⇒ S1**

**動作：文字列"S0"を、整数"i0"で指定された文字幅の左端に位置させる。文字幅が"S0"より小さい場合は無視される。通常、空白文字でパッディングされるが、オプションの"c1"で指定された場合は、その文字が使用される。"c1"は、一文字のみの文字列とする**

**※なお、"c1"にUnicode文字も指定できるが、あくまで「文字数的に中央」という処理となるので、ASCII文字とUnicode文字の幅が異なる日本語の環境では、変な結果になります。打ち出し画面参照**

```
>>> "abc".center(7)
'  abc  '
>>> "abc".center(7,"*")
'**abc**'
>>> "abc".center(8,"*")    # 左寄り
'**abc***'
>>> "abcd".center(7,"*")   # 右より
'**abcd*'
>>> "abc".center(7,"＊")   # 文字幅はあくまで文字数指定
'＊＊abc＊＊'
>>> "ａｂｃ".center(7,"*")
'**ａｂｃ**'
>>> "abc".ljust(7)
'abc    '
>>> "abc".ljust(7,"*")
'abc****'
>>> "abc".rjust(7)
'    abc'
>>> "abc".rjust(7,"*")
'****abc'
```

### ■数字の前を"0"で埋める

**書式：N0.zfill(W) ⇒ N1**

**動作：数値文字列"N0"の前を、整数値"W"で指定した桁数になるまで"0"で埋める。ただし"N0"の桁数が"W"で示した桁数以上なら、"N0"はそのまま返される。なお、"N0"は数値文字列でなくても一向に構わないが、意味は無い。「N0.rjust(W,"0")」と意味的には同じ**

```
>>> "123".zfill(10)
'0000000123'
>>> "123".zfill(2)
'123'
```

**■タブの大きさを調整する**

**書式：S0.expandtabs([TS]) ⇒ S1**

**動作：文字列"S0"の中のタブの大きさを"TS"に指定されたセル数に指定して、ホワイトスペースに変換して"S1"に返す。"TS"を省略すると８スペースタブとしてみなされる**

**※タブとしてちゃんと機能していることを示すために、'print'出力してみました**

```
>>> tab08="abc\tdef\tghi\nabcx\tdefx\tghi".expandtabs()
>>> tab10="abc\tdef\tghi\nabcx\tdefx\tghi".expandtabs(10)
>>> print(tab08)
abc     def     ghi
abcx    defx    ghi
>>> print(tab10)
abc       def       ghi
abcx      defx      ghi
>>> tab08
'abc     def     ghi\nabcx    defx    ghi'
>>> tab10
'abc       def       ghi\nabcx      defx      ghi'
```

### 6-2-2-2-3-2-1-7 変換用メソッド

様々な文字列の変換を行うメソッドです。

**■文字列をバイトコードに変換**

**書式：S.encode(encode="utf-8", errors="strict") ⇒ bytes**

**動作：文字列"S"をバイトコードに変換する。デフォルトはUTF-8だが、例えば"sjis"とすればSHIFT-JISコードに変換する。ただし、変換できなければエラーとなる。エラーの動作のオプションは以下のとおり**

**"strict"……'UnicodeEncodeError'を上げる(デフォルト)**

**"ignore"……無視する(「b''」を出力)**

**"replace"……不明文字"?"に置き換える**

**"xmlcharrefreplace"……XMLで用いられるキャラクターコードに変換する**

```
>>> "abc".encode()
b'abc'
>>> "あいうえお".encode()
b'\xe3\x81\x82\xe3\x81\x84\xe3\x81\x86\xe3\x81\x88\xe3\x81\x8a'
>>> "あいうえお".encode("sjis")
b'\x82\xa0\x82\xa2\x82\xa4\x82\xa6\x82\xa8'
>>> "abc".encode("ascii")
b'abc'
>>> "あいうえお".encode("ascii")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-4: ordin
al not in range(128)
>>> "あいうえお".encode("ascii","ignore")
b''
>>> "あいうえお".encode("ascii","replace")
b'?????'
>>> "あいうえお".encode("ascii","xmlcharrefreplace")
```

```
b'&#12354;&#12356;&#12358;&#12360;&#12362;'
```

※'str.encode'は、'bytes.decode'に対応しているので、「B.decode(encode)」の書式で逆変換をかけられます

```
>>> x = "あいうえお".encode('sjis')
>>> print(x.decode("sjis"))
あいうえお
```

## ●変換テーブルを作成して変換

### ■変換テーブルの作成

書式：str.maketrans(x[, y[, z]]) ⇒ dict

動作：キーが整数（文字コード番号）、値が変換文字（あるいは文字コード）の辞書を作成する。１引数のみの場合と、２引数、３引数の動作が異なる。１引数のみの場合は、キーが文字番号として使用できる整数か文字１文字、値はそれに加え、'None'オブジェクト（変換後省かれる）と、複数文字列が指定できる。２引数、３引数の場合は、第１引数と第２引数は「文字数が同じ」文字列である必要がある。第１引数に指定した文字が、同じ位置の第２引数に指定した文字に変換される。なお、第３引数に指定した文字は'None'に置き換えられる（つまり変換後省かれる）なお、静的メソッド（static method)なので、'str'オブジェクトの位置に任意の文字列を置いても使用できるが、その文字列は文字列クラス'str'の呼び出しに使用されるだけで、何の意味も無い（静的メソッドについては定義／変則参照メソッド／静的メソッドを参照）

### ■変換テーブルに従って文字を変換

書式：S0.translate(TD) -> S1

動作："S0"の文字列を、変換テーブル(辞書＝dictオブジェクト)に従って変換した文字列"S1"として返す。"TD"はキーが整数(文字コード番号）、値が文字、文字コード番号(整数)及び'None'オブジェクトの辞書である。フォーマットさえ正しければ、特にstr.maketranseを使用しなくても良い。'None'オブジェクトがマップされている文字は削除される

```
>>> for k in "abcxyz":print(k+"="+str(ord(k)))
...
a=97
b=98
c=99
x=120
y=121
z=122
>>> str.maketrans({"a":"x"})                    # １引数
{97: 'x'}
>>> str.maketrans("abc","xyz")                  # ２引数
{97: 120, 98: 121, 99: 122}
>>> str.maketrans("abc","xyz","d")              # ３引数
{97: 120, 98: 121, 99: 122, 100: None}
>>> TD = str.maketrans("abc","xyz","d")
>>> "aaabbbcccddd".translate(TD)                # テーブルを利用
'xxxyyyzzz'
>>> "aaabbbcccddd".translate({'a':'x'})         # テーブルを手打ち
'aaabbbcccddd'
>>> "aaabbbcccddd".translate({97:'x'})
'xxxbbbcccddd'
>>> "aaabbbcccddd".translate({97:120})
'xxxbbbcccddd'
```

### 6-2-2-2-3-2-1-8 '%'演算子を用いた古い形式の文字列の整形

現在の新しい Python では、文字列は format メソッドで整形することが前提となっていますが、以前から用いられていた"%"演算子を用いた手軽な文字列整形方法も一応今でもサポートされています。

これは、C 言語の'printf'関数で使用するようなスタイルの文字列の整形方法で、'%'が頭につく代替文字を後につづく単体オブジェクトや、リストや辞書の値と入れ替えるようなスタイルです。

本書では詳しく触れませんが、古い Python の本やサイト、あるいは C 言語の本などを見ていただければ大体わかります。

【Python のマニュアルに載っているコードサンプルの転記】

```
>>> print('%(language)s has %(number)03d quote types.' %
...       {'language': "Python", "number": 2})
Python has 002 quote types.
```

[by "The Python Standard Library"
"4.6.2. Old String Formatting Operations"]

# バッチの技術 - ファイルのリネームと移動

jscripter(まとめ)

## １．はじめに

TSNET でのでびさんの表題の課題についての検討結果を簡単にまとめる。でびさん、Y ささん、KIMURA さん、aMI さんのバッチや説明を検討してまとめている。ありがとうございました。

## ２．課題

あるディレクトリでバッチを実行して、ディレクトリ内のサブディレクトリにあるファイルのファイル名の先頭に、サブディレクトリ名を_を挟んで付加した名前にリネームし、サブディレクトリからバッチを実行したディレクトリに移動させる。

## ３．バッチ1(Y ささんのバッチから)

わかってみれば、なんということはない。cmd.exe を実行して、バッチコマンドの仕様を確認して、下記のバッチの記述と対比させてみよう。例えば、「for /?」を実行してみる。ちなみに「%%~nxf」 は 「ファイル名と拡張子」に展開 されることがわかる。

```
-----^
for /d %%d in (*) do for %%f in ("%%d¥*.*") do ren "%%f" "%%d_%%~nxf"
for /r %%f in (*) do move "%%f" .
-----$
```

## ４．バッチ２(aMI さんのバッチから)

バッチ１では入れ子になっている for～in～do の構造を、call でサブルーチンを呼ぶように分離している。「call /?」で仕様を確認してみること。

```
-----^
for /d %%d in ( * ) do call :loop1 "%%d"
exit /b

:loop1
pushd %1
for %%f in ( * ) do call :loop2 %1 "%%f"
popd
exit /b

:loop2
```

```
ren %2 %1_%2
move %1_%2 ..
exit /b
-----$
```

## ５．半角空白を含むディレクトリ名やファイル名への対応

上記のバッチはいずれも、ディレクトリ名やファイル名、パスが文字列として展開される変数出力部分をダブルクォート「""」で括っている。半角空白をディレクトリ名やファイル名が含んでいる場合には必須である。代入される変数入力部分にはこの操作は不要である。

## ６．cmd.exe のコマンド拡張機能

「setlocal /?」を調べると、最後の方に次のように記載されている。

```
-----^
VERIFY OTHER 2>nul
SETLOCAL ENABLEEXTENSIONS
IF ERRORLEVEL 1 echo 拡張機能を有効にできません

この方法が使えるのは、古いバージョンの CMD.EXE では、SETLOCAL は
ERRORLEVEL の値を設定しないためです。VERIFY コマンドに誤った引数を
指定すると、ERRORLEVEL の値は 0 以外の値に初期化されます。
-----$
```

あなたの環境で前記のバッチがうまく動かない場合には、これを利用して確かめる必要があるかもしれない。

コマンド拡張機能はデフォルトで有効になっている。レジストリの
HKEY_LOCAL_MACHINE→ Software→ Microsoft→ Command Processor
のキーの値「EnableExtensions」が 1 であれば有効であることが確認できる。デフォルトは 1 である。

前記バッチでコマンド拡張機能が有効である必要があるコマンドは、次のとおりである。

(1) コマンド拡張機能を有効にすると、call のターゲットとしてラベルが使えるようになる。詳しくは「call /?」の出力を参照のこと。

```
call :ラベル 引数
```

(2) 以下の for コマンドも拡張機能を有効にする必要がある。詳しくは「for /?」を参照

のこと。

-----^
for /d %変数 in (セット) do コマンド [コマンドパラメータ]

セットがワイルドカードを含む場合は、ファイル名ではなくディレクトリ名
の一致を指定します。

for /r [[ドライブ:]パス] %変数 in (セット) do コマンド [コマンドパラメータ]

[ドライブ:]パスから始めて、ツリーの各ディレクトリで for 文を実行し
ます。/r の後にディレクトリが指定されていない場合は、現在の
ディレクトリが使用されます。セットが単一のピリオド (.) である場合は、
ディレクトリ ツリーの列挙だけを行います。
-----$

以上。

(投稿: 2012年 2月 5日)

# 覚醒と進化 試論 II フレームワーク

jscripter

## １．DesktopWebフレームワーク

さて、問題は何か。最近、GoogleがDesktopから撤退して、デスクトップは真空地帯となっている。僕は、Google Desktopが出たときに、これはヤバイと考えたものだ。しかし、デスクトップの検索ツールから大きく発展することはなかったので、我らの仕事は今のところ安泰である。

現在では、デスクトップやWebの情報整理ツールはevernoteなどのクラウド・アプリに委ねられている。似たようなツールは溢れているが、今のところ満足ができるような状況には至っていないように思う。単なる収集整理、スクラップ、検索のためのツールでは知的生産のためには足りない。自らが産み出す情報と合わせて、編集できることが必要だろう。

自分が今やっていることをWebアプリケーションにまとめることができるだろうか。そんなことを長い間、漠と考えてきたが、もう少し具体的にしたい。それをDesktopWebフレームワークと呼ぶことにした。実体が今のところないが、少しずつ作っていこう。試論Iで述べた様々なシステムをモジュール化して、統合したフレームワークとして発展させることが目的である。

## ２．小さな一歩^^;）デスクトップCGIが動作するPCのIPアドレスを得る

もちろん、モジュールを書いたことはあるけど、フレームワークを作るところまで考えたことはなかった。それでも、まずはモジュールづくりの復習から。

デスクトップCGIは、主にインターネットに接続する個人の小さなネットワーク上で動作することを想定している。そのような場合、デスクトップCGIが動作するサーバーのIPアドレスはDHCP(Dynamic Host Configuration Protocol)で得られることが多いだろう。

この場合、サーバーに他のPCやタブレットなどからアクセスしたい場合もある。が、デスクトップCGIに埋め込まれる別の機能を持つデスクトップCGIのURLのホスト名はlocalhostでは動かない。IPアドレスで直に記述しておく必要がある。しかしながら、これを変数として取り扱えるようにしておかないと環境が変化した場合にスクリプトの該当箇所をすべて書きなおす必要が出てくる。これは避けたい。

変数にしておくのはよいが、CGI実行時の値を取得するモジュールを準備する必要がある。Zazelさんに相談に乗ってもらって、いろいろと検討していて思いついたのが、pingをコンピュータ名を指定して実行する方法である。

```
ping -n 1 $computername
```

と実行する。-nオプションはpingする回数を指定する。コンピュータ名は環境変数COMPUTERNAMEから取得する。この方法はWindows PCだけしか使えないと思われるので、その他のマシンで動作させる場合には他の方法も検討する必要があるだろう。

さて、モジュールping2ip.pmは次のようである。

```
package DesktopWeb::ping2ip;
require Exporter;

our @ISA = qw(Exporter);
our @EXPORT = qw(ping2ip);
our $VERSION = 0.01;

sub ping2ip{
	my ($computername) = @_;
	unless($computername){
		$computername = $ENV{'COMPUTERNAME'};
	}
	my $ip = "can't detect.¥n";
	my @pingstr = `ping -n 1 $computername`;
	foreach (@pingstr){
		if(/(¥d{1,3}¥.¥d{1,3}¥.¥d{1,3}¥.¥d{1,3})/){
			$ip = $1;
			last;
		}
	}
	return $ip;
}
1;
```

モジュールを使ってみよう。スクリプト名をping2ip.plとしよう。

```
use DesktopWeb::ping2ip;

print ping2ip;
```

実行する。

```
C:¥>perl ping2ip.pl
192.168.0.15
```

うまく動く。

## ３．次の一歩

次は、デスクトップの情報について考えてみたい。情報の種類、発生の仕方、格納場所などによって、インターフェース表現を考える必要がある。それをモジュール群としてまとめ直す。

現在のブラウザ上のデスクトップはHTMLのFRAMEをインターフェースとして使っている。この方法はiPadなどのモバイルデバイスには向かない。少し研究して試行錯誤する必要があるだろう。

（投稿: 2012年 2月 6日）

# 編集後記

jscripter

予想外にみなさんの投稿をいただいて、今号は大変うれしかった。

ハッカーが世の中を動かす時代という。僕らもWebで何か新しいサービスを生み出せればと考えたりしている。客観的に見て、個人の能力は限りなく増大しつつある。それは、コンピュータとインターネットがもたらしたものである。

DesktopWebフレームワークがそれを現実のものとするようにしていきたい。と、ここで小さな声で宣言しておこう。

以上。

(投稿: 2012年 2月 5日)

# ＴＳＮＥＴスクリプト通信

ISSN 1884-2798　出版地: 広島市

2012年 2月 6日　4.3.001版
2012年 2月 7日　4.3.002版
2012年 2月 9日刊行　4.3.003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)


Ｙさ　saw[at]s7[dot]wh[dot]qit[dot]ne[dot]jp
でび　davi-1984[at]nifty[dot]com
ムムリク　qublilabo[at]gmail[dot]com
機械伯爵　kikwai[at]livedoor[dot]com
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4巻第3号(通算第15号)
発行: TSC編集委員会 発行日: 2012年 2月 9日
ISSN: 1884-2798 出版地: 広島市 創刊: 2008年5月7日