TSNET スクリプト通信

TSNET

目次

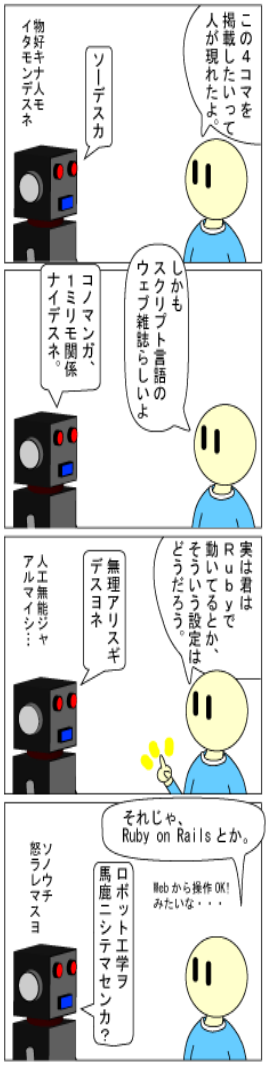
はじめに

「TSNET スクリプト通信」は今のところ、3 ヶ月に一回刊行するというペースで季刊を想定している。自分のことを考えると、3 ヶ月あれば、なんとか一つの問題について考えたことが形を成しはじめると思うからである。しかし、締め切りがなければなかなか書きそうもない。今回もＹささんと機械伯爵さんの投稿に後押ししてもらって書いた。書けそうだと思うことと実際書いてみることとは違う。なかなかそれなりに大変だった。今回は、Zed というエディタについて書いた。要望にいろいろと答えていただいた作者の久世さんに感謝する次第である。Zed というエディタがさらに大きく発展することを願っている。

Ｙささんの AWK スクリプトは例によってコマンドラインゲーム、その名は「鍵」。試してみるのが毎度の楽しみになってきた。整頓されたスクリプトはいつも清清しい。

機械伯爵さんは、「やｎでレ Python ～新約蛇神祭祀書～」の連載がはじまった。Python3000 に関するものであり、大作になる予感。楽しみである。Python を勉強したいならこれを読むべし。僕も読もう一っと。

表紙写真は、結局、jscripter 氏のといっても僕の新作になった。秋の宮島遠望である。



海鳥さんのロボ漫画も連載にさせていただくことにした。前号の転載への依頼に対して漫画を描いていただいたので、TSNET のページにも転載したが、記録としてここにまず転載させていただくことにして、前書きを終わる。

次号もお楽しみに。

2008 年 11 月 30 日
jscripter

鍵

written by Yさ

1. このゲームは

前作(VVF2=いわゆる格闘ゲーム風な攻防に数当て的要素をミックスしたもの)に引き続き、貧相な画面のゲーム第四弾。

今回は、0～9の数字を1回づつ使用した10桁の数列を手掛かりをヒントに高度な判断で推理するゲームです。

...って結局数当てなのですけど(^^;

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 3.1.6, 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 key.awk[ENTER]
~~~~~~~~~~~~~~~~~~~~~~
```

等で起動するとゲームスタートです。

ゲーム中に h か H または ? を入力すると、簡単なヘルプを表示します。

1)0～9の数字を1回づつ使用した10桁の数列を入力して下さい。
　当たり判定を行い、ヒントとして数列の一部と、一致した桁の数が表示されます。
2)全桁を当てるとラウンドクリアですが、8回以内に当てないとゲーム終了となります。
3)ラウンドクリア毎に問題のレベル(?)が最大5迄上がります。
　最終ラウンドまで当て続けて下さい。

ちなみにスクリプトの LAST_STAGENO の値で最終ステージを変更できます。デフォルトは5となっています。

※なお ROUND_LIMIT は目安ですので、気にしなければ10より大きくできます
　d(^^;

今回は、「どうしてもラウンドクリアできない」とお困りの方の声にお応えして、隠し機能をご用意いたしました。

スクリプトのコメントアウトしてある mode ="easy" の行を有効にするか、あるいは、

```
-v mode ="easy"
~~~~~~~~~~~~~~
```

のオプションを付けて起動すると、簡単なヘルプ表示の際に"特別ヒント"を一緒に表示するように変更できます。

ただし、"特別ヒント"にいわゆる全角文字を使用していますので、環境によってはうまく表示されないかもしれません。

..."特別ヒント"がいったい何を意味しているのか、といった事につきましてはスクリプトから読み取ってくださいませ(^^;

※蛇足ヒント：問題レベル4までの数列は、おおまかに、
　・単純に順番に並べたもの

・数字を2グループに分け、順番もしくは交互にミックスして並べたもの
・世の中にある数字が並んでできている"もの"を 3×3＋1 といった順にならべたもの
といった規則で作成されます。

4.開発環境など
ソースは、例によってかなり昔に会社の行き帰りの満員電車で立ったまま作成したC言語版を、今回 awk に移植したものです。
当時も相変わらずマシンは HP100LX でした v(^^;) が、HP も 200LX の生産中止にしちゃったし、液晶パネル蓋の止め具はバカになっているし、電池入れ蓋はセロテープで止められているし、液晶パネルのヒンジが突然もげて瞬間接着剤でかろうじて固定されていたりもするけれど、まだ元気よく動いていてくれていて、他のマシンには絶対到達できない便利さ＆タフさなのでした。しみじみ。

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TSNetへ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。

[key.awk]

```
##  鍵  written by Ｙさ

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

BEGIN{
##
## 定数定義
##
LAST_ROUND =5
# ROUND_LIMIT =10  ## 設定可能な最終ラウンド（目安）

  last_round=LAST_ROUND;  # default


# 答え格納用
# a[10];

# 答え(ヒント)表示用
  split("HTM,HMT,MHT,MTH,THM,TMH", showPattern, ",");
# showSw;
# hint;
# mode ="easy";  ## 特別ヒント表示（※この行を有効にする）
```

```
  # ゲームメイン
  doGame();
}


##
##  HELP 表示
##
function helpDisp(){
  print "";
  print "  0～9の数字を1回ずつ使用した10桁の数列を当ててください。";
  print "  ヒントとして数列の一部と、一致した桁の数が示されます。";

  if(mode=="easy") printf("¥n  ※特別ヒント：〔 %s 〕¥n", hint);
  print "";
}


##
## 答え設定
##
function ansSet(level,  i,p,sw,tmp){
  if(level==1) sw=(rnd(200)<150)?(rnd(8)+ 1):(rnd(8)+13);
  if(level==2) sw=(rnd(300)<250)?(rnd(7)+23):(rnd(8)+ 1);
  if(level==3) sw=(rnd(200)<100)?(rnd(2)+21):(rnd(4)+ 9);
  if(level==4) sw=(rnd(300)<200)?(rnd(8)+30):(rnd(8)+38);
  if(level>=5) sw=(rnd(1000)<10)?(rnd(45)+1):(100);

  if(sw<9){
    if(sw>=1){ for(i=0; i<=9; ++i) a[i]=i;       hint="S→L"; }     # seq
    if(sw>=2){ rev(0,9);                         hint="L←S"; }     # rev
    if(sw>=3){ rev(0,4);  rev(5,9);              hint="→LS→"; }    # L_up-S_up
    if(sw>=4){ rev(0,9);                         hint="←SL←"; }    # S_d-L_d
    if(sw>=5){ rev(5,9);                         hint="←S→L"; }    # S_d-up
    if(sw>=6){ rev(0,9);                         hint="L←S→"; }    # L_d-up
    if(sw>=7){ rev(0,4);  rev(5,9);              hint="→L←S"; }    # L_up-d
    if(sw>=8){ rev(0,9);                         hint="S→L←"; }    # S_up-d
    return; }
  if(sw<13){
    if(sw>=9) { for(i=0; i<=4; ++i){ a[i*2]=i; a[i*2+1]=9-i; }
                                                 hint="[0+9]→"; }  # 0add9
    if(sw==10){ rev(0,9);                        hint="←[0+9]"; }  # rev 0add9
    if(sw>=11){ for(i=0; i<=4; ++i){ a[i*2]=9-i; a[i*2+1]=i; }
                                                 hint="[9+0]→"; }  # 9add0
    if(sw==12){ rev(0,9);                        hint="←[9+0]"; }  # rev 9add0
    return; }
  if(sw<21){
    if(sw>=13){ for(i=0; i<=4; ++i){ a[i]=i*2+1; a[i+5]=i*2; }
```

```
                                              hint="0→E"; }     # odd-even
  if(sw>=14){ rev(0,9);                       hint="E←0"; }     # rev odd-even
  if(sw>=15){ for(i=0; i<=4; ++i){ a[i]=i*2; a[i+5]=i*2+1; }
                                              hint="E→0"; }     # even-odd
  if(sw>=16){ rev(0,9);                       hint="0←E"; }     # rev even-odd
  if(sw>=17){ rev(5,9);                       hint="←0→E"; }   # rev_odd-even
  if(sw>=18){ rev(0,9);                       hint="←E→0"; }   # rev_even-odd
  if(sw>=19){ rev(0,4);  rev(5,9);            hint="E→0←"; }   # even-rev_odd
  if(sw>=20){ rev(0,9);                       hint="0→E←"; }   # odd-rev_even
  return; }
if(sw<23){
  if(sw>=21){ for(i=0; i<=4; ++i){ a[i*2]=i*2+1; a[i*2+1]=i*2; }
                                              hint="[0+E]→"; }  # odd-even mix
  if(sw==22){ rev(0,9);                       hint="←[0+E]"; }  # rev odd-even mix
  return; }
if(sw<30){
  if(sw>=23){ a[0]=7; a[1]=4; a[2]=1; a[3]=0;
              a[4]=8; a[5]=5; a[6]=2;
              a[7]=9; a[8]=6; a[9]=3;         hint="c↓l→r"; }  # calc l-r
  if(sw==24){ rev(0,3); rev(4,6); rev(7,9);   hint="c↑l→r"; }  # calc rev l-r
  if(sw==25){ a[0]=9; a[1]=6; a[2]=3;
              a[3]=8; a[4]=5; a[5]=2;
              a[6]=7; a[7]=4; a[8]=1; a[9]=0; hint="c↓r→l"; }  # calc r-l
  if(sw>=26){ a[0]=7; a[1]=8; a[2]=9;
              a[3]=4; a[4]=5; a[5]=6;
              a[6]=1; a[7]=2; a[8]=3; a[9]=0; hint="c→u↓d"; }  # calc up-down
  if(sw==27){ rev(0,9);                       hint="c←u↑d"; }  # calc down-up
  if(sw>=28){ a[0]=1; a[1]=4; a[2]=7;
              a[3]=2; a[4]=5; a[5]=8; a[6]=0;
              a[7]=3; a[8]=6; a[9]=9;         hint="t↓l→r"; }  # tel l-r
  if(sw==29){ a[0]+=2; a[1]+=2; a[2]+=2;
              a[7]-=2; a[8]-=2; a[9]-=2;      hint="t↓r→l"; }  # tel r-l
  return; }
if(sw<38){
  if(sw>=30){ a[0]=1; a[1]=2; a[2]=3; a[3]=5; a[4]=7;
              a[5]=0; a[6]=4; a[7]=6; a[8]=8; a[9]=9;
                                              hint="∧→◇"; }   # open-close
  if(sw>=31){ rev(0,9);                       hint="◇←∧"; }   # rev open-close
  if(sw>=32){ rev(0,4);                       hint="◇→∧←"; } # up close-down open
  if(sw==33){ rev(0,9);                       hint="∧→◇←"; } # up open-down close
  if(sw>=34){ a[0]=0; a[1]=4; a[2]=6; a[3]=8; a[4]=9;
              a[5]=1; a[6]=2; a[7]=3; a[8]=5; a[9]=7;
                                              hint="◇→∧"; }   # close-open
  if(sw>=35){ rev(0,9);                       hint="∧←◇"; }   # rev close-open
  if(sw>=36){ rev(5,9);                       hint="∧←◇→"; } # down open-up close
  if(sw==37){ rev(0,9);                       hint="◇←∧→"; } # down close-up open
  return; }
if(sw<46){
  if(sw>=38){ a[0]=8; a[1]=3; a[2]=4;
```

```
                a[3]=1; a[4]=5; a[5]=9;
                a[6]=6; a[7]=7; a[8]=2; a[9]=0;  hint="囲-1";  }    # magic square #1
    if(sw==39){ rev(0,2); rev(3,5); rev(6,8);    hint="囲-1'"; }    # magic square #1'
    if(sw>=40){ a[0]=4; a[1]=9; a[2]=2;
                a[3]=3; a[4]=5; a[5]=7;
                a[6]=8; a[7]=1; a[8]=6; a[9]=0;  hint="囲-2";  }    # magic square #2
    if(sw==41){ rev(0,2); rev(3,5); rev(6,8);    hint="囲-2'"; }    # magic square #2'
    if(sw>=42){ a[0]=2; a[1]=7; a[2]=6;
                a[3]=9; a[4]=5; a[5]=1;
                a[6]=4; a[7]=3; a[8]=8; a[9]=0;  hint="囲-3";  }    # magic square #3
    if(sw==43){ rev(0,2); rev(3,5); rev(6,8);    hint="囲-3'"; }    # magic square #3'
    if(sw>=44){ a[0]=6; a[1]=1; a[2]=8;
                a[3]=7; a[4]=5; a[5]=3;
                a[6]=2; a[7]=9; a[8]=4; a[9]=0;  hint="囲-4";  }    # magic square #4
    if(sw==45){ rev(0,2); rev(3,5); rev(6,8);    hint="囲-4'"; }    # magic square #4'
    return; }

  # random
  for(i=0; i<10; ++i) a[i]=i;
  for(i=0; i<10; ++i){ tmp=a[(p=rnd(10))]; a[p]=a[i]; a[i]=tmp; }
  hint="？";
}
## 答え数列反転
function rev(st,ed,  i,tmp){
  for(i=0; i<=int((ed-st)/2); ++i){ tmp=a[st+i]; a[st+i]=a[ed-i]; a[ed-i]=tmp; }
}

##
## 答え(ヒント)表示
##
function ansPut(c,v,  i,t,l,s,tmp){
  tmp="";
  if(c==10){
    for(i=0; i<10; ++i) tmp=tmp sprintf("%d", a[i]);
  }else{
    if(c==1 || c==4 || c==7) showSw=rnd(6)+1;
    t=(c-1)%3; l=int((c-1)/3)+1;
    s=substr(showPattern[showSw],t+1,1);
    if(s=="H"){  # head
      for(i=0; i<l; ++i) tmp=tmp sprintf("%d", a[i]);
      tmp=tmp sprintf("%*s", 10-l, " ");
    }
    if(s=="T"){  # tail
      tmp=sprintf("%*s", 10-l, " ");
      for(i=0; i<l; ++i) tmp=tmp sprintf("%d", a[10-l+i]);
    }
    if(s=="M"){  # mid
      tmp=sprintf("%*s", 4-int(l/2), " ");
      for(i=0; i<l; ++i) tmp=tmp sprintf("%d", a[4-int(l/2)+i]);
```

```
      tmp=tmp sprintf("%*s", 6+int(l/2)-l, " ");
    }
  }
  printf("        %s", tmp);
  if(v>0)  printf(" (%02dhit%s)", v, ((v==1)?(""):("s")));
  if(v==0) printf(" (nohit)");
  if(c<10) print "";
}

##
## 答え合わせ
##
function ansChk(s,  i,r){
  r=0;
  for(i=0; i<10; ++i) if((substr(s,i+1,1)+0)==a[i]) ++r;
  return r;
}

##
## 0～9の数字全てを使用しているかチェック
##
function notAll(s,  i,tmp,p){
  # help ?
  p=toupper(substr(s, 1, 1)); if(p=="/")  p=toupper(substr(s, 2, 1));
  if(p=="H" || p=="?"){  #  help !
    helpDisp();
    return -1;
  }

  for(i=0; i<10; ++i) tmp[i]=0;
  for(i=0; i<10; ++i) ++tmp[(substr(s,i+1,1)+0)];
  for(i=0; i<10; ++i) if(tmp[i]!=1) return -1;
  return 0;
}


##
##  ゲームメインループ
##
function doGame(  v,c,s,h,b,round,level,t){
  srand();

  h=0;
  for(round=1; round<=last_round; ++round){
    s=0;  ansSet((level=(round<5)?round:5));
    printf("¥n*** Round%02d (Level%02d) ***¥n¥n",round,level);
    for(c=1; c<=8; ++c){
      do{ printf("[%2d] >> ", c);  t="";  getline t; }while(notAll(t));
      if((v=ansChk(t))==10) break; else{ ansPut(c,v); s+=v; }
```

```
    }
    print "";  ansPut(10,-1);
    printf("  %s¥n", ((v==10)?("Round clear"):("Over times")));
    print "";
    if(s>0)   printf("          Hit point %02d¥n", s);
    b=(round<5)?1:3;
    if(v==10) printf("        Bonus point %02d¥n", b*=10*(10+1-c));
              printf("              Total %02d¥n¥n", h+=(s+b));
    if(v!=10) break;
  }
  print "";
  if(round>last_round) print "Congratulations !!";
  else printf("Game Over  (Result:Round%02d)¥n",round);
}
```

やｎでレPython
～新約蛇神祭祀書～

NARRATOR #2: This man is about to die. In a few moments, now, he will be killed, for Arthur Jarrett is a convicted criminal who has been allowed to choose the manner of his own execution.

ナレーター：この男性はもうすぐ死にます。有罪宣告された犯罪者アーサー・ジャレット。彼は、自分自身の処刑方法を選ぶことを許されました。今現在、彼は数分のうちに殺されるでしょう。

『MONTY PYTHON'S THE MEANING OF LIFE』

０．マナブが学ぶ理由？

現実にも恋愛ゲームのようにフラグがあるのなら、自分はフラグを着実に立てつつある。
錦織真武（にしこりまなぶ）はそう確信しつつ、転けて床に這いつくばっている小さな少女に手を差し伸べた。
少女は顔半分を赤くしながら（どうやら床にぶつけたらしい）、涙目でマナブを見上げている。
これで、この少女に手を差し伸べるのは５回目だ。
転けて床にぶちまけたノートを集めるのを手伝ったり、転けて床にぶちまけたプリントを集めるのを手伝ったり、転けて倒した書棚の本を収納しなおすのを手伝ったり……
最初は偶然だった。
入学式直後、同じクラスになった女子が、床にお店を開いて往生しているのを、なんということはない親切心で助けたのがきっかけだった。
その時、至近距離ではじめて少女の顔を見た。
黒目がちの大きな目。
艶やかな長い黒髪。
透き通るような白い肌。
そして、その繊細な顔の造りの全てに、心奪われた。
以来、なんとなく彼女の近くにいるようにしていたら、次々と勝手にトラブルに遭うので、それを何度も手助けすることになったのだ。
最初のうちは、申し訳なさそうな顔でびくびくしていた少女も、この頃になるともう馴れたので、照れ笑いしながら「また助けられちゃった」と素直に手を伸ばしてマナブの手を取る。
体に見合った小さな手。
小さく華奢な体は、同じ高校生だとはどうしても思えない。
手を握るたびにドキドキしながらも、マナブは平生を装う。
「気にしなくていいよ。手は減るものじゃないし」
あまり気の利いた台詞じゃないな、と自らつっこみながら、その軽い体を引き上げる。
そして、床に散らばったノートや教科書（教室移動の途中だったのだ）を手早く集め、少女に手渡す。
「ありがと」
少女ははにかみながら、にっこりと笑ってその束を受け取る。

「あたし、どんくさいから、錦織くんに迷惑かけちゃうね」
「誰だって失敗はするよ。僕だっていつかは、困って羽生さんに助けてもらうことになるかもしれない」
「ネズミとライオンだね。うん、そのときは、いっしょーけんめー助けるね」
　無邪気に笑って答える姿に釣り込まれて、頬を緩めてしまう。
　もし最初のアクシデントが無ければ、このように話すことなど、多分無かっただろう。
　そもそも、マナブは中学までは女子とは別の世界に生きていた。
　コミックやラノベやゲームやアニメでの疑似恋愛はあっても、現実(リアル)の女子に心を動かされることはあまりなかった。
　そして少女……羽生一子(はぶいちこ)にしても、最初はクラスで一緒になって『えらくちっちゃい女の子がいるな』くらいの印象しか無かった。
　イチコは、飛び抜けて小さい体以外は、人の印象に残りにくい地味な少女だった。
　最初に手を貸した時、あまりにも驚いた様子だったので質すと、どうやら今までは、転けても誰も気づいてくれなかったらしい。
　だから、助けてもらうことも嬉しいが、気づいてもらったことはもっと嬉しい、とイチコは言う。
　家族以外の人の印象の中に自分がいることが、こんなに嬉しいことだと思わなかった、と。
　当然、友達といえる友達もなく、男女含めて同級生と用もないのに話すのは、マナブが初めてらしい。
　マナブも幸いというべきか、中学からの友人は全て別のクラスになってしまったので、最初の頃の放課中（この地方の言い方で、授業の間の休み時間のこと）は、もっぱらイチコと話すことが多かった。
　そうこうしているうちに、マナブにも男子の友人ができ、マナブといることで存在感が生まれたのか、イチコにも女子の友達ができて、二人だけで話す時間はやや少なくなったのだが、それでも機会やきっかけがあれば、今でもよく話す。
　話しているうちにわかったのだが、イチコは『理系』だった。
　学校の理数系教科が得意なだけでなく、殊コンピュータに関しては、自分でプログラムまで組むという女傑だった。
　マナブは、特に文系というわけでもないが、それでもコンピュータは普通にインターネットに繋いでウェブを覗いたり、ワープロや表計算ソフトを使ったりできるくらいでプログラミングなどとは全く縁が無い。
　モバイルパソコンに、無心にプログラムらしきものを打ち込んでいるイチコを見て、マナブはふと呟いた。
「羽生さんは、まるで呪文を操る魔法使いみたいだね」
　イチコはびっくりして、大きな目をぱちくりと瞬かせ、マナブを見た。
「魔法使い？」
「うん。だって不思議な言葉でコンピュータを操るからね」
　顔に朱がさす。
　色が白いので、それがはっきりとわかる。
「えっとね、天才的なプログラマのこと、ウィザードって言うんだよ」
「ウィザード……魔法使い？　そうなんだ。僕はハッカーって言うのかと思ってたけど」
「ウィザードはハッカーより上の尊称だよ。ハッカーは天才とかそういう意味より、システムのコードの密林(ジャングル)に分け入っていく、冒険者っていうイメージかな」
（ウィザード　ハッカー）
「魔法使いとか冒険者とか、コンピュータの世界ってファンタジーみたいだね」
「うん、自動プログラムはデーモンって言うし、反応しないプログラムの破片はゾンビって言うよ」
　思ったよりコンピュータってゲームっぽいんだな、と思う。
「羽生さんは、そんな魔法の世界を自由に操れるんだ、いいなぁ」
　その途端、イチコの瞳に『妖しい』光が宿った。

「錦織くんも、プログラム、覚えてみる？」
思いもかけない言葉に、マナブは苦笑する。
僕がプログラムだって？
無理に決まってる。
「無理無理。僕は羽生さんみたいに頭良くないよ。それに、覚えるの大変そうだし……」
ところが、イチコにしては珍しく食い下がる。
「プログラム自体は簡単だよ。プログラムで何かをしようと思うと難しいけど、コンピュータが得意な分野、例えば計算とかデータベース管理とかなら、意外なほど簡単に出来るんだよ」
「そ、そうなんだ。でも……」
「**あたし、教えるから**」
異様な迫力でずずぃっ、と迫ってくるイチコ。
久々に見たアップは、目が少しだけ怖かったけど、やっぱり綺麗で……
「う、うん。わかった。教えて貰おうかな」
そんな誘惑に、マナブが耐えられる筈などなかった。

翌日、それまで全く予想していなかったシチュエーションで、初めて自分の部屋に女の子が入ることになった。
イチコはその日の帰りに寄ると言ったのだが、当然「準備」の出来ていない部屋へ招き入れるわけにもいかないので、なんとか一日伸ばしてもらった。
その日の放課後、マナブはこっそりと近くの書店に寄り、プログラミングの本を数冊開いてみた。
全部、なんだか難しそうだった。
あらためて見ると、プログラミング言語というのは何種類もあった。
その中で、比較的冊数も多く、「日本人のつくった」言語があるということを知った。
本の書き始めなどを見る限り、初心者でも修得しやすい、と書いてあったので「羽生さんが教えてくれるのは、もしかしてこれかな」と勝手に思ってみたりした。
とりあえず、簡単な本を読んで予習でもしておこうかと思ったのだが、無理そうなので断念した。
家に帰り、久々の家の掃除と片づけ。
両親が共働きで、夜まで家を空けることが多いのは幸いだったが、そのために散らかし放題になっていた家を片づけのは一苦労だった。
それからお約束の「女の子に見られてはならないモノ」を封印し、押し入れの奥に仕舞いこむ。
その中には、イチコに似た雰囲気の少女が出てくる年齢制限のあるコミックもあり、思わず手が止まった。
邪気を封印し、窓を開けて空気を入れ替え、さっぱりとした気持ちで床につく。
しかし、青く若い力のうねりは、彼の眠りを何度も妨げた。
結果、熟睡することなく、翌朝を迎えることとなった。
そして、翌日。
学校の帰りに、イチコはマナブの家に着いてきた。
聞けば、家はマナブと同じ方向で、思っていた以上に近所だった為、これから帰りも一緒に帰れることも判明した。
ツいている、マナブは確信した。
すべてが自分の都合のいいように回っているような、そんな気分にさえなった。
そして、入念に用意した自分の部屋へとイチコを招く。
マナブの部屋には、ベッドと書き物机と本棚だけのシンプルな構成だった。
「パソコン、貸してくれる？」
「いいよ」
イチコは、普段マナブがネットを見たりゲームをしたりする時に使用しているパソコンの電源を入れた。

OSはWindows Xp Home Edition。
コンピュータに詳しいイチコに余計なものを見られないために、マナブは既に部屋と同じく「不適切な」ものはまとめて取り外し式のハードディスクに納めて隔離してある。
準備は万端の筈だった。
しかし、イチコはパソコンを一別しただけで、いたずらっ子のように笑った。
「錦織くん、ブラウザの履歴、処理してある？」
「え？」
「ブラウザで見るページは、一旦ハードディスクに保存するんだよ。だから、テンポラリフォルダとか見れば、錦織くんが昨日見たページとか、全部わかっちゃうんだよ？」
「ははは、は、み、見られてマズいものなんて……」
「じゃ、覗いちゃうよ？」
「……ごめんなさい、許してください」
くすくす笑うイチコ。
「ちゃんと消しておいてね。でないとあたし……」

———*軽く殺意覚えるかも*———

「え？」
聴き間違いかと思ったが、笑うイチコの目が全然笑っていない、どころか、据わっているのに気づく。
「ＣＧデータでもね、錦織くんが他の女の子を見てるの、**ヤなの**」
ある意味、男冥利につきる台詞を聞いているにも関わらず、マナブの心に湧きあがったのは、喜びではなく、**純粋な恐怖**だった。
「男の子だから、そういうの仕方ないけど、あたしの目の届く範囲ではやめてね。**どうなるか、わかんないよ**」
マナブは無言で首を何度も縦に振った。
「今日は許してあげる。錦織くんがパイソニスタになる記念の日だから」
「ぱいそにすた？」
ふと、イチコがUSBメモリからインストールしているプログラムを見る。
ウィンドウトップのタイトルバーに「Python」という文字があった。
「ぱいそん、って読むのかな？」
「うん。**世界一美しいプログラミング言語**だよ」
「へぇ、Rubyとかじゃないんだ」
なにげなく言った一言に、イチコは椅子を回していきなり振り向いた。
その顔は幽鬼の如く蒼白で、目が血走り、まるでマナブを怨み殺さんばかりに下から睨めつけていた。
「錦織くん……*ルビィストだったの？*」
「え？　な、なにそれ？」
「……Rubyが習いたいの？」
マナブは、ここで首を縦に振ってはいけないという本能に従った。
「ちがうよ。ただ、本屋でたくさん本が出てたし、簡単だって書いてあったし、日本人がつくったとかあったから、それかな、と思っただけ」
「そう」
夜叉のような表情は一応納まったものの、瞳の奥にくすぶる狂気の焔は、まだ消えていないようだった。
「Rubyはいい言語だよ。設計思想も統一してるし、比較的理解しやすい言語。でも、錦織くんのような**プログラムが未経験な完全な初心者には向かない**んだよ」
「僕にはそういうのわからないから、羽生さんに任せるよ」
「そう？」

見違えるように明るい笑顔を見せるイチコ。
その笑顔にときめきながらも、先ほどの**恐怖**が心の隅にしっかりと根を下ろしていた。
「じゃ、このPythonが一番。Pythonは教育用言語としての実績も高い言語なんだよ。それに理解しやすいから、プログラマ以外の人とプログラマが打ち合わせする時なんかにも使えるんだよ」
「じゃ、初心者に特化した言語なんだ？」
「ふふふ、そう思うでしょ？　でも、違うんだよ。錦織くんも、Googleは使ってるよね？」
「普通には。検索エンジンとしては一番有名だからね」
「そのGoogleのエンジンの一部は、かなり前からPythonで書かれてるんだよ」
「ホント？」
「ホントホント。今はPythonを作った人もGoogleに勤めてるし、Googleの基本技術の柱の一つになってる。つまり、実用でもバリバリに利用されてる言語なんだよ」
「へー凄いんだな」
イチコの目が、さらに輝きを増す。
「他にも、ILM(Industrial Light and Magic)ではＣＧアニメを制御するのにPythonを使ってるし、IBMみたいにハードウェアテストに使ってるとこもあるんだよ。高度な学術計算にも使われてるし、3Dグラフィックツールの Blenderみたいに、拡張機能として組み込んでるものもあるんだよ」
「……すごいね。それって、怖いもの無しじゃない？」
イチコは少し寂しそうに笑う。
「万能な言語も環境も無いよ。Pythonは優秀な言語で環境なんだけど、万能じゃないよ。ある特徴を取り入れる為には、ある特徴をあきらめなきゃならないようなトレードオフは常にあるんだよ。Pythonは広く愛されている言語だけど、全てのプログラマを満足させることはできないんだよ」
そして、吹っ切るように晴れ晴れとした表情を見せる。
「でも、これだけは言えるよ。最初に習うべき言語はPythonしかないって。他の言語へは、必要ならばいずれ移ることはできるよ。なんならRubyにでも移れる。でも、短期間にプログラムを書けるようになれて、しかも覚えた言語がずっと使える言語は、**Pythonしかない**んだよ」
イチコの熱っぽい弁論に、マナブもだんだんとハマってきた。
「うん、Pythonやるよ。そして早く、羽生さんみたいにプログラム、書けるようになるんだ」
「……でも、最初はあまり派手なこと出来ないから、それだけは覚悟してね？」
「ははは、はい。わかりました」
いつしかインストール終了のメッセージが出ていた。
「じゃ、さっそく始めよっか」
「宜しくお願いします先生」
「うん、がんばろうね」
マナブの頭の中には、もう先ほどのイチコの態度は、頭の中から抜けていた。

第１回　はじめてのPython

「じゃ、まずは対話モードで立ち上げて」
「どうやってやるの？」
「スタートメニューのPython 3.0の中の、Python(command line)をクリックするの」
　すると、真っ黒な画面に、なにやら文字がずらっと現れた。
「うぁっ、うぁっ、英語が出たっ、英語が出たっ！」
　騒ぎ立てるマナブ。
　インストール画面でも、さんざん見ていた筈だが、改めて自分の操作で出るとびっくりするらしい。
「錦織くんって、英語苦手だったっけ？」
「フツー。可もなく不可もなく」
「じゃ、メッセージ読めるんじゃない？」
　確かにそこには、バージョン情報とアップデート、それに『helpとかcopyrightとかcreditsとかlicenseとか打ち込んだら、もうちょっと情報が出るよ』程度の情報しか書いていない。
「別に読む必要無いけど」
　必要以上にびくついた照れ隠しに、反論するマナブ。
「必要無いの？　じゃ、なんで出るんだろう？」
「プログラミング言語は、バージョンによって微妙に仕様が変わるから、確認のためってトコかな？」
「だったら一行でいいのに」
　それ以上食い下がっても意味なさそうなので（これ以上イチコと喧嘩するつもりも毛頭無いので）とっとと鉾を収めた。
「まぁいいや。で、その下の『>>>』は？」
「プロンプト(prompt)。『入力を促すもの』って意味。Pythonの式や文が入力できますよっていう、しるしだよ」
「なるほど。で、まずは何をしよう？」
「そうだね、まずは雰囲気を掴んでもらおうかな。１＋１とか打ってみて」
「テンキーでいいの？」
「いいよ」
「じゃ、１＋１と」
「終わったらenterキーを押して」
「enter……っと」

```
>>> 1+1
2
>>>
```

「あ、答えが出た」
「まずは、電卓みたいなコトは出来るの、わかった？」
「フツーだね」
「フツーだよ。安心した？」
「うん……っていうか、拍子抜けっていうか」
「プログラミングって、もっと難しいものだと思った？」
「うん。だってこんなの、電卓でも出来るから」
「そうだね。じゃ、ちょっと電卓ではできないことしてみよっか。１＋２×３の答えっていくつ？」
「暗算？　７だよ。かけ算が先だからね」
「ご名答。でも、電卓でそう打ち込むとどうなるか知ってる？」
「多分、順番に計算しちゃうから、９になるかな？」
「じゃ、そのまま、打ち込んでみようか？」
「×の記号は？」

「×は＊(アスタリスク)、÷は／(スラッシュ)を使うんだよ」
「じゃ、１＋２＊３と……enterだよね？」
「よくできました」

```
>>> 1+2*3
7
>>>
```

「へぇ？　ちゃんと順番守って計算してくれるんだ」
「ちょっとしたことだけど、便利でしょ？」
「そうだね。もしかして、かっことか使える？」
「使えるよ。やってみて」

```
>>> (1+2)*3
9
>>>
```

「これって、関数電卓かポケコンみたい」
「……錦織くん、随分古いの知ってるんだね」
「昔親父が使ってたのを、さわらせてもらったことがあるんだ」
「じゃ、ポケコンのBASICみたいな使い方、してみよっか？」
「べーしっく？」
「昔、コンピュータには標準装備されてたプログラミング言語＋実行環境なんだって。あたしたちが生まれる前のことだから、あたしも話しか知らないしらないけど、本で見たことがあるから」
「そうなんだ」
「変数の使い方は、PythonでもBASICでもほとんど同じだよ。むしろPythonのほうが簡単かも」
「どうやってやるの？」
「まずは、aという変数に５という数字を入れてみようか。ａ＝５って打ち込むと、aという名前の変数に、５という数字が入るんだよ」

```
>>> a=5
>>>
```

「こんどは答えが出ないね？」
「よく気づいたね。Pythonでは、代入式は文だから、答えは無いんだよ」
「式？　文？」
「うん、詳しいことは、もう少し慣れてから話すよ」
「わかった」
「じゃ、もう一つ。ｂ＝７って打ち込んでみて」

```
>>> b=7
>>>
```

「これで、aは５として、bは７として使えるんだよ。計算してみて」
「えっと、こんな感じかな？」

```
>>> a+b
12
```

「あ、計算できた」
「もっといろいろ試してみて」
「じゃ、一通り……」

```
>>> a-b
-2
>>> a*b
35
>>> a/b
0.7142857142857143
```

「あちゃ、割り切れないかやっぱり。それじゃ、こうして……」

```
>>> b/a
1.3999999999999999
```

「あれ？　７÷５は1.4で割り切れるはずだけど？」
　イチコはちょっと微妙な顔をする。
「このへんがコンピュータの面倒なトコなんだけど、十進法では割り切れても、0.4をぴったりと二進法で表現することはできないんだよ。だから、二進法で計算できる範囲の近似値で計算しちゃうんだよ」
「……なんか電卓より不便じゃない？」
「そうだね。きちんと人間が見やすい形にするには、print関数を使うんだよ」
「ぷりんとかんすう？」
「printという名前の関数だよ」
「紙にプリントアウトするの？」
「昔、ディスプレイが無かった頃、タイプライターキーボードで紙にプリントアウトしながらコンピュータと対話していたんだよ」
「うぁ、すごい紙の無駄遣い」
　その頃はコンピュータも少なかったけどね、とイチコはこっそりほほ笑む。
「printっていう名前は、その頃の名残らしいよ。Pythonは新しい言語だけど、こういう名前は割とクラシックなものをそのまま使ってるんだよ。モダンっぽい言語だと、putsなんて名前になってるものもあるよ」
「putsとprintでは、あんまり変わらないなぁ」
「BASICでは、よく使うからっていう理由で、『P.』で省略できたんだって」
「あ、それは便利」
「Pythonでも、裏技を使えば省略できるけど、最初からは難しいからそのうち教えるよ。今はとりあえず、やってみて。printと打って、その後にかっこで囲って計算内容を打ち込むんだよ」
「かっこで囲むの？」
「古いPythonの書式では囲まなくても出来たんだけど、今はそうするんだよ」
「なんでわざわざ、面倒なルールに変えるんだろう？」
「**崇高な目的のため**だよ」
「ふへっ？」
　ふと、イチコの目に、またぞろ**怪しげな光**が宿っているのに気づく。

Benevolent Dictator For Life

「かっこで囲まない書式は、**『情け深き生命の絶対者』の若気の至り**だったんだよ。彼は**その過ちを後悔して、書式の改訂を命じられた**んだよ。でもそのおかげで、**様々な恩恵に預かれる**ようになったんだよ」
「……」

いろいろ突っ込みたかったが、なんだかイチコの様子が怖かったので、余計なことは言わない事にした。
「と、とにかく打ち込んでみるね」

```
>>> print(b/a)
1.4
```

「あ、綺麗に出た」
「本当は、計算結果を人間が見る時は、print関数に通すのが正式なんだよ。そもそも、この対話モードでは、確認のためにいちいち式の値を表示するようになっているんだけど、スクリプトを別のファイルに作って走らせると、print関数を使わない限り、値は表示されないんだよ」
「そうなんだ。このモードは特別なんだね」
「特別、とはいっても、Pythonの全機能が使えるから、あまり長すぎて一気に打ち込みきれないスクリプト以外は、大体何でも試せるんだよ」
「えっと……さっきから『スクリプト』って言ってるけど、何のこと？」
「あ、ごめん。スクリプト(script)ってプログラムと同じ意味だと思っていいよ。本当は微妙に違うんだけど、スクリプトっていう言葉が色んな意味で使われるから、厳密な定義は難しいんだよ。
Pythonでは、プログラムのことをスクリプトって言うこともあるって覚えておけばいいよ」
「ギョーカイヨーゴってヤツ？」
「そうそう」
少し嬉しかった。
「話を元に戻すけど、その対話モードでは、どうして最初から、人間の見える形にしておかないんだろう？」
「いろいろ理由はあるけど、つまるところはPythonの方針としか言いようがないんだよ。言語によっては、普通に小数点表示したり、中には分数表示したりするものもあるけど、どれがいいとは一概に言えないんだよ」
「それも、そのうち分かる？」
「そうだね」
「じゃ、続き、いこう」
「まず代入だけど、今のような計算式の結果も、代入できるんだよ。書き方は同じように、＝の右に値を入れる式、左に収納する変数を書くんだよ」
「こんな風かな？」

```
>>> c=10+15
>>>
```

「変数の式も使えるよ」
「じゃ、変数も」

```
>>> d=a*b
>>> e=a/2
```

「答えはプリントだね」

```
>>> print(c)
25
>>> print(d)
35
>>> print(e)
```

2.5

「printって何度も打つのが面倒なら、カンマでつなげば簡単だよ」

```
>>> print(c,d,e)
25 35 2.5
```

「あ、並んで出てきた」
「print関数には、もうちょっといろんな使い方があるけど、まずはこんなトコにしとくね。今日はとりあえず、どんな道具が使えるのか、簡単に見るだけ。今まで出てきたのは、整数と浮動小数点数、それに関数と変数。これは全部『オブジェクト』っていうの」
「浮動小数点数？」
「小数部分のある数のこと。小数表現には、浮動小数点数に対して固定小数点数っていうのもあるんだけど、Pythonではデフォルトでは採用してないの。もちろん、自分で後で作れば別だけど」
「ふーん？」
　今ひとつわからなかったが、流した。
「あと、変数はわかるけど、関数も一緒？」
「一緒だよ。関数の扱いが飛び抜けて自由なのは、Pythonの特徴の一つだよ。そして、代入子と四則演算子。代入子は『＝』、四則演算子は『＋－＊／』が今まで出てきたよ」
「うんうん」
　半分くらいは、わかった。
「Pythonは基本的にオブジェクトと代入子、演算子、それに特殊な予約語と区切り(delimiter)の『：』を使ってプログラムを書くんだよ。予約語と区切りはまたの機会にするけど、まずはオブジェクトを覚えて。オブジェクトが使えれば、色々なことができるんだよ」
「オブジェクトって『数』なの？」
「そういえば、～数って付くオブジェクトしか使ってなかったね。じゃ、～数って名前がつかない文字列を使うよ。英語でストリング(string)って言うんだよ」
「ストリングって糸のことじゃなかったっけ？」
「そう。文字の繋がった糸のようなイメージだからそう呼ばれるんだよ」
「じゃ、使ってみよう」
「まずは、Ｃ言語で定番の『ハローワールド』にしようか。print関数のかっこの中に、ダブルクォーテーション(double quotation)で囲って文字を打ってみて」
「ダブルクォーテーションって？」
「『”』の記号。クォーテーションマーク(quotation mark)が二つくっついた記号だよ」

```
>>> print("hello world")
hello world
```

「ダブルクォーテンションか、シングルクォーテーション『’』で囲うと、その間は文字列として使えるんだよ」
「ダブルとシングルの違いは？」
「扱いが違う言語もあるけど、Pythonでは、全く区別が無いよ」
「なぜ、二通り書き方があるんだろう？」
「例えば、文の中でクォーテーションマークが使いたくなったとき、ダブルクォーテーションで囲った文字列の中ではシングルクォーテーションが、シングルクォーテーションで囲った文字列の中では、シングルクォーテーションが使えるよ」
「あ、そっか。結構便利かも」
「もちろん両方使うために、エスケープ(escape)っていう特殊な書き方で同じ記号を入れることもできるけど、手軽さや見やすさは全然違うよ」

「日本語は使えないの？」
「普通に使えるよ。『こんにちわ』してみて」

```
>>> print("こんにちわ")
こんにちわ
>>>
```

「うん、普通だね」
「じゃ、hello っていう変数に、"こんにちわ"って代入してみて」
「え？」
「どうしたの？」
「変数の名前って、アルファベット１文字じゃないの？」
「256 文字までは認識するよ。そんな長い変数名、普通使わないけど」
「日本語の変数名も使えるの？」
「使えるけど、使わないほうがいいと思うよ」
「どうして？」
「あとで教えるけど、日本語の扱いって色々微妙なんだよ。文字列の中身として使う分には問題無いんだけど、変数名とかで使うのはお薦めできないよ」
「じゃ、やめとこ。えっと、hello に"こんにちわ"だね。イコールを使うんだよね？」
「そうそう」

```
>>> hello="こんにちわ"
>>>
```

「じゃ、hello の後に[0]ってつけて、print してみて」
「？？」

```
>>> print(hello[0])
こ
>>>
```

「『こ』？」
「『こんにちわ』の『こ』だよ。０から始まるナンバーが振られているんだよ」
「じゃ、１番は『ん』？」
「確かめてみて」

```
>>> print(hello[1])
ん
>>>
```

「あ、コレ面白い」

```
>>> print(hello[2])
に
>>> print(hello[3])
ち
>>> print(hello[4])
わ
>>> print(hello[5])
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

「わあっ、なんか文句言われたっ！」
「５文字だから０番から４番までしかないのに、５番を指定したから怒られたんだね」
「じゃぁ、最後の文字とか出したい時には、文字数を知ってないと駄目だね」
「文字数はlen関数で出るよ。でも、最後の文字を出すだけなら、－１番を指定すればいいよ」

```
>>> len(hello)
5
>>> print(hello[-1])
わ
>>>
```

「……－１って、もしかしてぐるっと回って最後、って意味」
「そうだね。後ろから２番目なら－２だから」
「真中何文字、とかも出来る？」
「出来るよ」

```
>>> print(hello[2:4])
にち
>>>
```

「……なんか、指定の数が妙なんだけど……『に』は２番で『ち』は３番だよね。なぜ４なの？」
「あはは、ちょっとややっこしいかもしんない」
　イチコはメモ帳を取り出し、下のような図を書いた。

[0]こ[1]ん[2]に[3]ち[4]わ[5]

「一文字だけの指定なら、番号の直ぐ後の文字。コロンを入れた書式、コレは『スライス(slice)』って言うんだけど、これは、番号と番号の間の文字を切り出すんだよ。[2]と[4]に挟まれてるのは『にち』だから、それが出たんだよ」
「あれ？　[5]がある？」
「[5]の後ろの文字は無いけど、[5]の前ならあるから」
「じゃ、『にちわ』と出すには……」

```
>>> print(hello[2:5])
にちわ
>>>
```

「……でいいのか、やっぱり」
「その場合、５は省略するのが普通だよ」

```
>>> print(hello[2:])
にちわ
>>>
```

「いろいろ出来るんだね」
「スライスにはもうひとつ『何文字飛ばし』っていう指定もできるんだけど、今回はやめとくよ」
「もう、お腹いっぱい」
「ふふふ、文字列操作はこんなもんじゃないよ。でも、そうだね、最初から飛ばしすぎはよくないね」
「Pythonで扱えるのは数字と文字、でいいの？」
「もう一つ、原始的(primitive)な生のバイナリデータ、つまりコンピュータのデータそのものも扱えるけど、扱いが難しいから、もう少しプログラムに慣れてから、だよ」
「終了は？」
「exit()かquit()を打ち込むか、WindowsならCtrl-zを入れてenterでもいいよ」
「じゃあ、exit()と」
　ウィンドウが消滅する。
「どうだった？」
「うん……覚えることが多そうな割には、出来ること少ないなって思った。結局、計算や文字操作してただけだもん」
　マナブの正直な感想に、イチコは苦笑する。
「そうだね。でも、ソフトウェアって結局、やってることは計算したりデータを変更したりしてるだけなんだけどね。今回はバイナリデータは扱わなかったけど、たとえば文字だけだって、自在に扱えれば、Webページの自動生成とかもできるよ」
「……あ、そっか。HTMLもテキストだっけ？」
「それに結局、バイナリデータも文字列と同じシーケンス(sequence)だから、特定のデータを見つけてそれを変更することができれば、バイナリデータもテキストデータと同じように使えるよ」
「シーケンスって？」
「順番に並んでいるもの、っていう意味。コンピュータのデータは、基本的には全て一列に並んでるの」
「意外に単純なんだね」
「そうだよ。コンピュータプログラムがやってることは、ホントは単純だよ。あまり単純すぎて、実際にやりたいことを単純な命令に分解するのが難しいだけ。それも、Pythonみたいな今のプログラミング言語は、ある程度は簡単にできるように準備してあるんだよ」
「じゃぁ、昔より、今のほうがプログラミングって簡単なの？」
「コンピュータの性能も上がって、出来ることは飛躍的に増えたから、それをフル活用するのは難しくなったけど、昔に苦労して何時間もかけて書いて何時間もかけて実行できるようにして、ようやく動いたプログラムが、今なら数分でコードを打ち込んで、一瞬で実行できるんだよ」
「じゃあ、もっとプログラムできる人が増えてもいいのに」
「昔は、プログラムを書かないとコンピュータが動かなかったんだよ。コンピュータを使うことはプログラムを書くこととイコールだった時代があったんだよ」
「僕の親父の時代がそうだったみたい」
「その頃は、コンピュータを使う人も使える人も少なかったって聞くよ。でも、コンピュータユーザは全員プログラマだったんだよ」
「……それはそれでイヤかも」
「だから、プログラムを書かなくても使えるようになって、ようやくコンピュータは一般的な道具になったんだよ。でも、プログラミング環境も進歩したのに、今度はプログラムを書く人が少なくなっちゃったんだよ」
「面倒なことは誰しもイヤ、と」
　イチコは強く首を振った。
「プログラミングは面倒じゃないよ。むしろ、コンピュータをフル活用するなら、プログラミングは不可欠だよ」
「一般には、必要無いよね」
「**……錦織くん、プログラム覚えるのイヤになった？**」
　リッチの冷たい眼差しで見つめられ、マナブは首が千切れんばかりに横に振る。

「ちがうちがう、い、一般論だよっ！」
「そう、なら安心した」
　にっこり。
「錦織くんまでプログラミングを否定するなら、**あたし、悲しくて刺しちゃうかも**」
「……」
「でもね、プログラミングが流行らないのは、錦織くんみたいに『難しい』って誤解してる人が多いからなんだよ」
　何事も無かったかのように、話を続けるイチコ。
「プログラミングには勿論技術は必要だけど、コンピュータを普通に使える人なら、プログラミングもできるんだよ。それに、新しくて使いやすいプログラミング言語……特にLL(軽量言語 Light weight Language)と呼ばれている言語群は、プログラムを書く手間に対して、絶大な効果を得られるの。PythonとかRubyとかPerlとかPHPとかJavaScriptとかTclとか……まだまだ沢山あるよ。あ、**錦織くんは当分はPythonを知ってるだけで十分だけど**」
　後は知らなくていい、と言わんばかりの口調だ。
「そ、そういえば、羽生さんが言ってたパイソニスタって、もしかして『Pythonを使う人』のこと？」
「御名答。パイソニスタ(Pythonista)は今、日本でも勢力を拡大してるんだよ。でも、その大半はまだ**『旧約』つまり古いタイプのPythonを使ってる人**なんだよ。我々**『真のパイソニスタ』**は、**偉大なるBDFLの真の意志に基づき、『新約』即ち真のPythonを広めなければならない**んだよ」
「我々って、もしかして僕も？」
「もちろん、そうだよ。**それとも、違うの？**」
「……ち、ちがわないよ」
　答えを**間違えればデッドエンド**、と、脳内コマンドウィンドウが開いていた。
　恋愛ゲームは恋愛ゲームでも**『病みゲー』**だったか、と今更ながら気づいたが、もう遅かった。

　その後、一時間ほど他愛のない話をして、イチコは帰って行った。
　プログラミングの話題に触れなければ、イチコはいたって普通の女の子だった。
　いや、マナブとしては、ゲームやアニメなど少しオタっぽい話にも十分ついてきてくれるあたり、多分『普通』と言われる女の子と話すより楽しいかもしれない。
　早くプログラミングの話も出来るように、一緒に頑張ろう、とイチコはやる気満々だ。
　二回目のレクチャーも、この分なら遠い未来ではないだろう。
　イチコが帰った後、マナブはネットで、Pythonのことを調べてみた。
　イチコの言う通り「Pythonは初心者が最初に学ぶべき言語」という主張も、結構見つかった。
　アメリカでは児童用の教育用言語として用いられているというレポートもあった。
　Pythonが、BBCのどぎついコメディー番組"Monty Python's Flying Circus"に由来する、というのはかなり意外だったが、だからといってふざけた仕様があるわけではない（というか、並み居るプログラミング言語の中でも、ずば抜けて『大人しい』仕様であること自体が冗句なのかもしれないが）
　しかし、色々調べていくうちに、プログラミング言語のユーザというのが、一種の『宗教』あるいは『信仰』であることに段々気づいてきた。
　当然といえば当然だ。
　言語が文化そのものであるように、プログラミング言語はプログラマの思考を規定する。
　そのため、言語の優劣の比較は、時には宗教戦争の様相を呈する。
　例えば、イチコが洩らした『ルビィスト(Rubyist)』というのは、Ruby言語ユーザのことらしい。
　パイソニスタとルビィストは（外野のマナブから見れば）近親憎悪に近い感情があるらしく、過去に何度か水掛け論争を巻き起こしていた。
　ちなみに、言語の作者同士は普通に友好的な関係（もちろん設計思想などの差異はあるが）にあるところが、宗教の教祖／指導者とは全然違うところだが、下っ端同士はどうも仲が悪いらしい。
　イチコの反応は、どうやら程度差はあれ、特定言語のシンパの特徴らしい。

（まぁいいか、僕はプログラミングの話なんか羽生さん以外とする気ないし）
彼女と一緒にパイソニスタになったところで、マナブには何のデメリットも無い。
そう思うと、気が楽になった。
（そういえば、そもそもPythonってどんな意味だっけ？）
マナブが知るPythonといえば、有名なコルトの回転式拳銃の名銃、コルトパイソンくらいだった。
（Pythonは、ニシキヘビか）
ふむふむと頷いて、ふと下をみると、もう少し意味があった。

python n.
1.ニシキヘビ; 《獲物を締め殺す》 大蛇.
2.【ギリシア神話】 **ピュートーン** 《Apollo 神が Delphi で退治した大蛇》.
3 **《人に取りついて予言をさせる》 心霊; 《心霊に取りつかれた》 予言者**.

2.や3.の、妙に禍々しいイメージは何なのだろう？
気になって周囲の語句を調べてみた。

Pythia
【ギリシア神話】 **ピューティアー** 《Delphi の Apollo 神の**巫子(みこ)**》.

pythoness
女神官[予言者], みこ, **いちこ**; 《Delphi の》 Apollo 神のみこ.

（い、いちこって……）
今度は国語辞典を引っ張っていちこ（市子、神巫、巫子）という言葉は、**心霊を降ろす霊媒**の意味があることを知る。
ピュートーン(Python)のピューティアー(いちこ)、あまりにも禍々しい符合に、寒気を覚える。

こうしてゲームは始まった。

【注釈】

- **Python**
  当然のように、ver. 3.0 以降を指す

- **新約**
  Python 3000(Py3k, Python 3.0以降)のこと

- **蛇神祭祀書**
  この話の目的は、初心者のためのプログラミング入門と、言語宗教の偏見の暴露である

- **錦織**
  ニシキヘビに引っ掛けてある、などと喋ると、これがホントの蛇足。ちなみにニシキヘビの仲間には本当に『蛇足』がある

- **恋愛ゲーム**
  柔らかい言い方でいうとギャルゲ。もう少し直接的にはエロゲとか言われるモノも含む

- **羽生**
  波布とか飯匙倩とか書く、いわゆる沖縄のハブに引っ掛けてある。名前の方の由来は、文章の最後の方参照

- **理系**
  プログラマは理系なのだろうか？

- **女傑**
  女性の豪傑。さも女性のプログラマが珍しいような言い方だが、これは事情を知らないマナブの視点からのお話であることに注意。こちら側の人なら、世界最初のプログラマが女性だった、くらいのトリビアは常識だろう

- **モバイルパソコン**
  ノートパソコンより小さいパソコンのことらしい。モバイルノートとも言う。でも、モバイルノートだとなんの意味だかわからない

- **プログラムらしきもの**
  実はPythonスクリプト。こちら側の人なら『コーディングしている』とでも言うだろう。ちなみに事情を知らない高校生が『コーディング(coding)』を訳すと『暗号化する』となるらしい

- **ウィザード**
  一般的には、インストーラなどでおなじみの『指示通りの手順で操作していくと出来てしまうタイプのプログラム』を指す。Visual C++には『スケルトンウィザード』なるものが『居た』が、アレはリッチ(Lich)なのだろうか？

- **ハッカー**
  最近『犯罪者』という意味以外でハッカーという言葉が一般にも使われるようになってきたのは嬉しい限りだが、本来の意味でも非合法的な匂いのする言葉であるのは否めない

- **自動プログラムはデーモン**
  正確には常駐型監視プログラム。デーモン(daemon)は本来悪魔ではなく、daimon(守護神)の意

味らしい。一般の人には、maildaemon が有名（帷子悪魔……ではなくメール監視デーモン：宛先不明の手紙を突っ返す）

- **反応しないプログラムの破片はゾンビ**
  ゾンビプロセスのこと。Windows 2000 で OpenOffice.org を使うと、よくアンデッドになるので、タスクマネージャーで dispel している

- **「日本人のつくった」言語**
  アレです

- **年齢制限のあるコミック**
  マナブくん、君、年齢制限にひっかかってるのでは？

- **ツいている**
  漢字で書くと「憑いている」と書く（本当）

- **Windows Xp Home Edition**
  あまりにも広まりすぎて、Microsoft がサポート打ち切りを延長せざるをえなかった OS。前もそんなようなこと、あったような……。ちなみに次世代の Windows Vista は今でも嫌われている（プレインストールされているマシンですら動作が遅いってナニ？）

- **テンポラリフォルダとか見れば**
  見ません、普通は

- **世界一美しいプログラミング言語**
  イチコの主観です。念のため（以降の彼女の台詞も、かなり偏見がある）

- **ルビィスト(Rubyist)**
  るびまのバックナンバーを見てたら、どうやら命名はご本人らしい

- **Google のエンジンの一部は**
  本当は Yahoo のエンジンも Python で書かれてるんだけど……

- **万能な言語も環境も無いよ**
  あったらあったで、多分面白くない

- **その下の『>>>』は？**
  会話でどうやって『>>>』と言ったかは謎

- **ちゃんと順番守って計算してくれるんだ**
  でも、なぜ乗除算が先なのかの理由は、小学校では教えてくれない（中学校でも、項の説明を理解し逃すと、もうわからない）

- **関数電卓かポケコン**
  関数電卓はまだ現役だが、ポケコンは現在、工業学校の生徒の教育専用になっているらしい（sharp とかでまだ密かに売ってます）

- **プログラミング言語＋実行環境**

シェル、というより OS に近い存在だった

- **0.4 をぴったりと二進法で表現することはできない**
  小数点第三位くらいの、固定小数点数クラスくらいは、作っておくと便利かもしれない（私は有理数クラスは自作したが……）

- **昔、ディスプレイが無かった頃**
  流石にこの頃は、私(機械)も噂でしかコンピュータを知らなかった。実際、パンチカードとディスプレイの中間があることは、最初知らなかった。

- **かっこで囲むの？**
  関数ですから（もう予約語じゃないです）

- **崇高な目的**
  多分

- **情け深き生命の絶対者**
  と、BDFL(Benevolent Dictator For Life)を訳してみたが、神のことらしい。つまり Guido のこと。

- **対話モード**
  Python のシェル(Tcl 風)、Python の REPL(Lisp 風)、Python の irb(Ruby 風)

- **スクリプトっていう言葉が色んな意味で使われるから**
  少なくとも Monty Python's Flying Circus のスケッチ(sketch)の脚本(script)のことではない筈

- **分数表示したりする**
  Scheme とか Smalltalk とか（ちょっと羨ましい）

- **代入子**
  Python では代入は文扱いなので、『代入演算子』ではなく代入子。なお＋＝のような『演算代入子』とでも言うものはあるが

- **Ｃ言語で定番の『ハローワールド』**
  Ｋ＆Ｒ本の最初の例。Ｃ言語は、こんな短いモノですら、インクルードしたりコンパイルしたり大変だ。Java はもっと大変だ（でも嫌いじゃないけど）

- **クォーテーションマーク(quotation mark)**
  って何だっけ？　と、学生時代に悩んだ記号。引用符。ちなみに普通にクォーテーションマークといえば『“”』のこと。『‘’』で引用している文書を、浅学な私はコンピュータ以外では見たことが無い（多分あるのだろうけど）『’』は、普通一個で使って『アポストロフィ(apostrophe)』として使う筈だと思ったのだが。ちなみに Python お得意のトリプルクォートは二重引用符として実際に使用されるが、『“‘’”』と組み合わせて使うのが一般的だとか（イギリスでは逆に『‘“”’』も使うらしい）どちらにせよ、日本語記号では開き／閉じがあるのに、なぜ ASCII では『″』と『'』しかないのか？（バッククォート『`』はあるけど『｀』と『‘』は別グリフだし……）あ、そうそう、序でに、Py3k ではバッククォート評価も廃止された

■ **エスケープ(escape)**
エスケープシーケンシー(escape sequency)の略。本来は『順序から外れる』という意味で、文字列から外れてレイアウト(改行、タブ)や機能(ビープ音など)を鳴らすような特殊表記のことを指すが、もともと特殊な用途（ここで言えば文字列の開始と終了の記号）の記号がエスケープされると、そのまま文字として扱われる。なお、エスケープ記号としては、日本語コードで￥マークに文字化けするバックスラッシュ『＼』が有名だが、%記号や$記号や&記号を使う場合もある

■ **日本語の変数名**
Python の基本コードは UTF-8 でエンコードされたユニコードなので、日本語変数名を使ってもかまわないのだが、文字コードの問題はややこしいので、やらないほうが無難。それに全角文字が多用されるプログラムは、B-TRON のマイクロスクリプトみたいで、美しくない（私見）

■ **hello の後に[0]ってつけて**
以降は、ユニコード主体に完全に以降した Py3k のデモンストレーションも兼ねている。日本語を多用するプログラマにとっては有難いが、既に旧約で大量のコードを書いているプログラマにとっては悪夢に等しい。もちろん『u"文字列"』という書き方も廃止された。

■ **指定の数が妙**
ICON ばりにスライスが使えるため、文字区切りがかえって気になる。一度覚えてしまえば簡単なのだが、適当に覚えると後々まで祟る。

■ **もう、お腹いっぱい**
最初から色々と詰め込んではいけません（『気楽な入門編』に複素数が出てくると、多分文系の人は引くのでは？ >> Python チュートリアル）

■ **exit()か quit()**
Python 2.5 より装備。sys をインポートしなくても、エスケープ文字を入力しなくても終了できる

■ **HTML もテキスト**
こんなに Web 文化が普及した今でも、知らない人が多い

■ **ホントは単純**
ハードウェアコントロールの部分については、あまり単純だとは言えない

■ **リッチの冷たい眼差し**
『伝説のオウガバトル』参照。リッチなのになぜか神聖直接攻撃？

■ **我々『真のパイソニスタ』**
Gudio 師の意向に沿うユーザ、という意味なのだろうが、『真』とかいうと大抵カドが立つ（もちろんイチコの偏見）

■ **偉大なる BDFL の真の意志に基づき**
ちなみに、Guido 師は「初心者は本が揃ってるから、2.x 系から覚えた方がいいんじゃない？」とか、どっかで書いていた気がする……

■ **『病みゲー』**
『闇ゲー』がひっかけてあるのかも。ヤンデレなヒロインが跳梁跋扈する、大体流血沙汰で終わるゲーム。ヒロインが絶望的に死んでいく『鬱ゲー』や、ヒロインを寝取られる『ネトラレゲー』とか、世も末なゲームが存在したりする。『僕は……いやだ(by 破嵐万丈)』

■ **コルトパイソン(COLT PYTHON .357 MAGNUM)**
1955 に発売され、現在も製造されるロングランの名銃。その独特のフォルムには、日本人のファンも多い。コルトのダブルアクションの回転式拳銃のシリーズは、戦後初のコブラ(COBRA)モデルを筆頭に、パイソン、ダイヤモンドバック(DIAMONDBACK=ガラガラヘビ)やキングコブラ(KING COBRA)、そして.44MAG のアナコンダ(ANACONDA)など、蛇の名前がつくものが多い。

■ **ニシキヘビ(Python)**
ニシキヘビ科の蛇の総称。大型の蛇が多いが、基本的に気性は穏やかで、ボールパイソン（ボールのように丸まる性質がある）のように、ペットとして人気の高いものも多い。毒牙ではなく、獲物を絞め殺すコンストリクター(constrictor)タイプの蛇だが、小さな獲物は当然丸呑み。

■ **（イチコの言葉が変なの）**
仕様です。念のため。

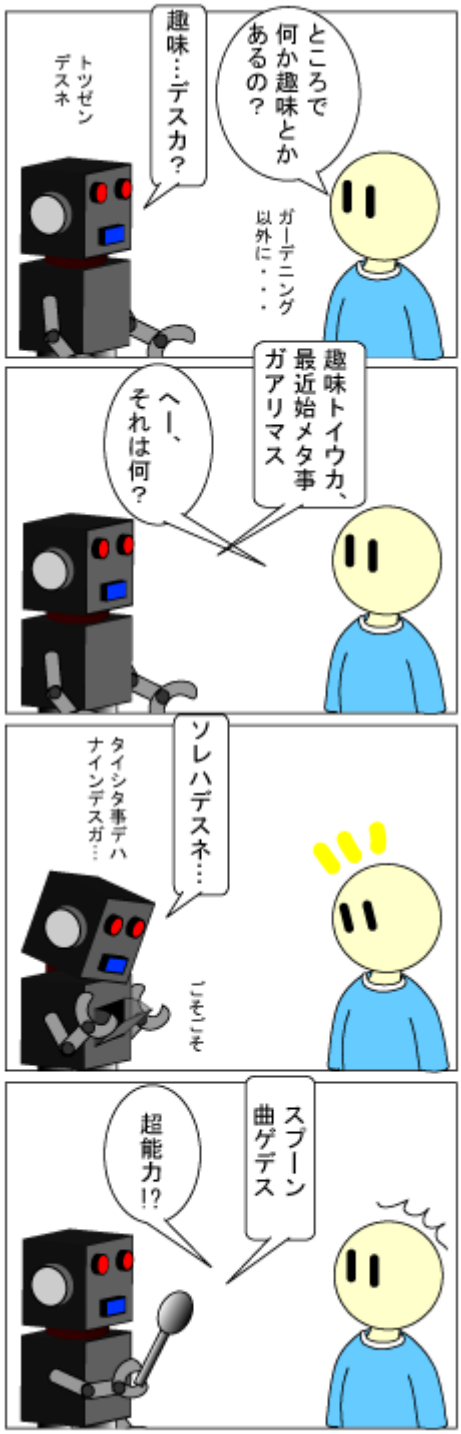
＜参考／引用＞

リーダーズ英和辞典(研究社)
Python チュートリアル(O'REILLY)
コルトのすべて(国際出版株式会社)
伝説のオウガバトル公式ガイドブック（アスキー出版局）

よしおさんとロボ太

海鳥作

「超科学」

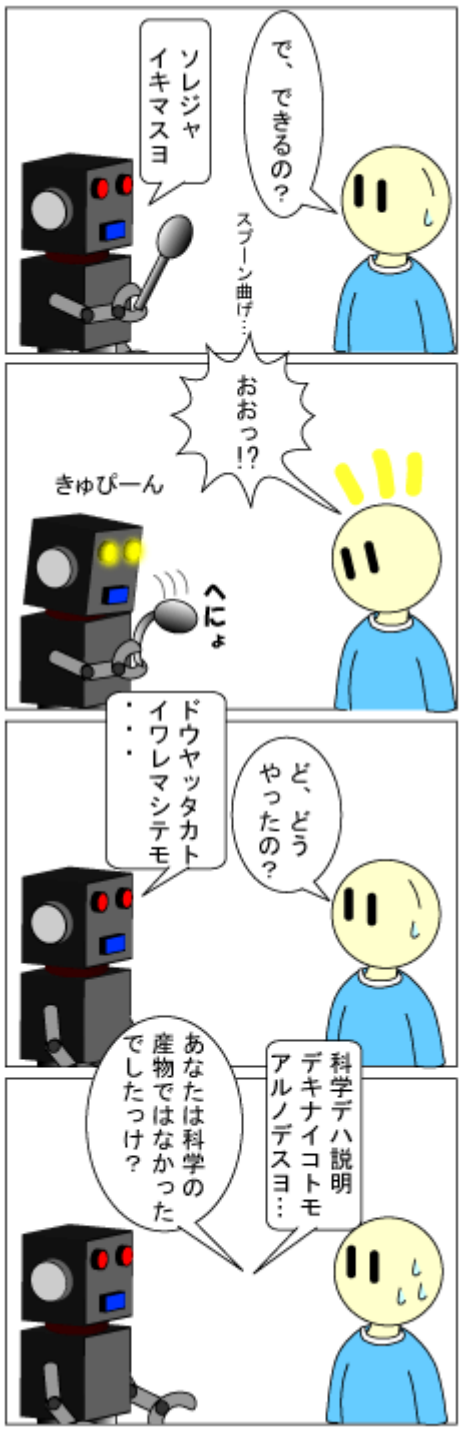


４〇元ポケット？

よしおさんとロボ太

海鳥作

「続・超科学」



ロボ諭す

Zed 入門

jscripter

I. 「エディタ」というテキスト処理のプラットフォーム

テキスト処理のプラットフォームとして最もポピュラーなのはコマンドラインです。Windows ではコマンドプロンプトか、DOS 時代の名残で DOS 窓などと呼ばれています。コマンドラインでは、例えば Perl なら-e スイッチを使って直接スクリプトを動かすことができます。

テキスト処理の他の重要なプラットフォームとして CGI とエディタがあります。

CGI は HTML のフォームからの入力を処理して、サーバー上に何かを記録したり、入力に応じた情報を HTML に変換してブラウザに送り返したりすることができます。HTML がテキストであることはご存知の通りです。

エディタはコマンドラインや CGI のように標準化されていないので、どのような方法でテキスト処理の自動化が可能になるかはエディタの作者の設計方針に依存します。エディタはテキストを取り扱うのでテキスト処理ツールであることは当然なのですが、すべてのエディタが自動的なあるいはプログラマブルなテキスト処理に対応しているとは限りません。

エディタとしては DOS では VZ エディタが Windows 用としては秀丸エディタが有名です。いずれも専用のマクロでテキスト処理を自動化しています。しかし、スクリプタが日常的に使用しているスクリプト言語をエディタ上でも使いたいと考えることは自然です。それに最初に答えてくれたのは著者にとっては乗松さんの Dana でした。Dana は素晴らしいエディタですが、UTF-8N の編集に対応しなかったために(UTF-8 は編集可能)、著者は UTF-8N 編集をきっかけとして寺尾さんの TeraPad を使い始め、結果として外部ツールとして標準入出力ツールを作っていただきました。このツールによって Perl をテキストの編集や編集中のテキストを保存して FTP でサイトに送り出したりすることを自動化しました(「実践実用 Perl」、毎日コミュニケーションズ、2004 年)。これらの機能は Dana でも実現できます。

著者は新しいエディタを見るといつも触ってみたくなります。どのような機能が実装されているのだろうか、試したくなるのです。これまでに見たことのない夢を実現するエディタを常に探しています。

II. Zed というエディタ

本稿で紹介する久世氏の Zed エディタは Perl などに編集中の選択範囲のテキストを渡し、スクリプトで加工した内容に置換出力することができます。その他にも興味深い特徴や機能があります。久しぶりに「これは使ってみたい」と思わせるエディタでした。

ここで、Zed の紹介を兼ねて「日曜プログラマのひとりごと」に書いた Zed に関するいくつかの記事を転載しておきます。

9/15/2008 (Mon.)

[Editor] 新星現る

Zed というエディタに出会った。α 版ということなのだが、デザインの良さに惹かれた。晴れときどき更新中: 自作ツール にある。

スクリプト言語の lua が編集に使えるということと正規表現エンジンとして鬼車を使っていることが、スクリプタを惹き付ける大きな要因でもある。

矩形選択ができること、アウトライン表示の実装、lua などのスクリプト言語を編集用に実装してい

るのを見るとDanaを思い出すが、テキストのタイプ別の設定、編集中文書の切替えの「文換え」や編集用のシェルの実装などの新機軸を持ち込み、魅力的である。

- Lua - Wikipedia
- Lua 5.1 リファレンスマニュアル
- DMonkey

更新: 2008-09-15T10:37:14+09:00

10/25/2008 (Sat.)

[Editor] Zed 0.34 リリース

Zed 0.34 リリース：晴れときどき更新中（掲示板）：@nifty レンタル掲示板：@nifty ネタ。とうとうスクリプトを自在に編集に使えるようになった。

いろいろと要望事項が溢れてきてしまう^^;)サブウインドウが編集のコマンドライン操作用となっているのが、スクリプタにとっては大変興味深い。

更新: 2008-10-25T12:05:08+09:00

11/3/2008 (Mon.)

[Editor] 日記編集の現場 - 思考の道具としてのエディタ

パターンマッチングマシン (2008/09/06)に堀田善衛、司馬遼太郎、宮崎駿の対談集「時代の風音」(朝日文庫、1997年、原著: 1992年)の書籍購入記録だけは載せていて、宮崎駿監督、堀田善衛を語る (2008/10/29)で再度取り上げようとしたのだが・・・

ある関連する話題を取り上げる場合、元の記事に戻って更新する場合と新たに記事を書き起こす場合がある。それはそれでよいとして、今回は記事が完成 しないで、時が過ぎ去り、月が替わってしまった。最近の記事は数日掛けて更新していく場合が多くなっているが、次第に考えていることを短時間では表現しき れないと感じるようになってきた。関連情報量の増大と思考・作業時間の不足がある。様々な類似したパターンが渦巻いている。

日記は時系列データだが、思考は過去と現在を自在に行き来する。物理的に時系列であることが思索の邪魔をする場合もある。ハイパーリンクを張ること によって記事の関連付けが簡単にできるのがWeb日記の大きなメリットなのだが、現在の「日曜プログラマのひとりごと」における思考と編集の過程では、修 正や追加をする場合、元記事のあるファイルや記事の場所を探すのは煩雑な作業となる。

エディタ上の編集作業でハイパーリンクやタブが使えると便利だと思う。「ファイル」を「開く」ダイアローグのインターフェースを使うのは大変非効率 だからだ。思考のプロセスが中断されてしまう。最近、エディタは、編集するテキストファイルに対してWebブラウザのような機能を持つべきと感じるように なった。制限的ではあってもデスクトップのファイルシステム上でハイパーリンクを実現することは可能だろう。Zedのようなスクリプトを動作させることが できるエディタはそのような可能性を秘めているようにも感じている。

更新: 2008-11-03T20:54:18+09:00

11/5/2008 (Wed.)

[Editor] Zed 0.35 登場

待ちに待った外部標準入出力ツール対応 Zed の改良版。久世さんありがとう。

いろいろと改善をしていただいたみたいなので、一つずつ試していこう。まずは選択範囲が広い場合の処理でエラーが出ないことの確認。変更点は次の通り。

今回の変更は、マクロ関係がメインです。

- Script や、DOS コマンドなどの標準入出力をエディタの選択領域にした場合、選択領域が広いとエラーになる不具合修正
- 正規表現による検索で"¥¥"の指定ができなかった不具合を修正
- Script の特別ルールを強化(Cmd, #!, Para, ShortCut)
- カレントフォルダに存在する Script も実行可能に変更

まずはここまで。以下、FTP 転送テスト。

・・・・・

・・・・・

・・・・・

・・・・・

うまく FTP スクリプトで編集ファイルを転送できるようになった。スクリプトを実行する時のディレクトリは自動的に編集ファイルのあるディレクトリとなるのを利用する。これで、ほぼ自在に使える^^)

更新: 2008-11-06T00:19:39+09:00

Zed は僕が使ってきたエディタの中では Dana に最も似ています。Dana は VB に似た編集用スクリプト言語 DanaScript を持っているのに対して、Zed は Lua などを編集用スクリプトとして使用できる点。もう一つは、アウトライン表示用ウインドウやファイル管理用ウインドウを装備していること。実はアウトライン表示用ウインドウの動作はまだ完全ではありませんし、ファイル管理ウインドウは未実装です。しかし、ユーザフレンドリーな高機能なインターフェースを実装しようという意思は Dana を思い起こさせます。

Dana や TeraPad は標準入出力機能を持つテキスト処理ツールなどを特別なインターフェースを通して外部コマンドとして登録が可能です。Zed はスクリプトそのものに動作指示用のコマンドを書き込んでおくという独自の方法を用います。ルールを知っていれば、スクリプトを書いて特定のディレクトリに格納するだけで、別途、登録作業をする必要がありません。起動するインタープリタも Unix と同様にスクリプトの先頭行の#!の後に指定することができます。クールですね。これらのスクリプトは自動的に編集のプルダウンメニューに登録されます。

次に Zed の入手先とサポート掲示板を載せておきます。

入手先: Kuze's soft Down Load room

サポート: 晴れときどき更新中 (掲示板) : @nifty レンタル掲示板 : @nifty

III. Zed 用のスクリプトの書き方

まず、久世さんの「Script の特別ルール」を Ver. 0.35 の Script 仕様.txt から引用します。

Script の先頭の 10 行に書き記載をすることで、特殊な効果が得られます。
Title = ○○○　・・・ メニューにおける Script のタイトルを○○○に設定
Input　・・・ エディタ部で現在選択している領域を、Script へ標準入力として与えます。
　ただし、現在 Lua, Dmonkey は、標準入力に対応できていません
Input = ○○○　・・・ エディタ部で現在選択している領域を、○○○というファイルに保存後、Script を実行します。
Output　・・・ Script の標準出力結果で、エディタの現在選択している領域を書き換えます。
Output = ○○○　・・・ Script 実行後、○○○というファイルの中身で、エディタの現在選択している領域を書き換えます。
Cmd = ○○○　・・・ Script を実行するためのコマンド名を指定します。
#!○○○　・・・ Script を実行するためのコマンド名を指定します。
　フォルダの区切りは、'¥'以外に、'/'も使えます。
　ドライブ名 'C:'は、省略可能です。
　本処理は、'/'を全て'¥'と解釈するので注意が必要です。
Para = ○○○　・・・ Script 実行時に与えるパラメータを指定します。
　パラメータには、下記マクロ文字が使用できます。
　　$P：カレントディレクトリ
　　$D：カレントドライブ
　　$F：現在編集中のファイル（フルパス）
　　$f：現在編集中のファイル（ファイル名のみ）
　　$C：現在編集中のファイル（拡張子を含む,$f と同じ）
　　$X：現在編集中のファイル（拡張子を含まない）
　　$$：'$'自身
ShortCut = ○○○　・・・ マクロ実行のためのショートカットキーを指定します。

・例 1. sort.bat

```
@ECHO OFF
rem Title = 選択領域ソート実行
rem Input
rem Output
rem ShortCut = Shift+Ctrl+S
sort
```

上記、sort.bat の場合、エディタの現在選択している領域が、DOS の sort コマンドの
入力として実行され、その出力結果で、エディタの現在選択している領域を書き換えます。
具体的な動作で書くと、

```
  3
  2
  1
```

と書かれた部分を選択後、サブウィンドウ or メニューから、sort を実行すると

```
  1
```

```
  2
  3
```

となります。
また、コマンドの実行は、"Shift+Ctrl+S"でも可能です。

・例2. abc.pl

```
#!/Perl/bin/MSWin32-x86-object/jperl.exe -w
# Para = -K $F
…
```

上記、abc.plの場合、以下のように実行されます。
C:¥Perl¥bin¥MSWin32-x86-object¥jperl.exe -w abc.pl -K 現在編集中のファイル

次に実例を示します。編集中のスクリプトをFTPで転送するPerlスクリプトです。スクリプトの特別ルールはコメント行に記述します。下記の例は、jperlを起動しています。#!の次に「C:」などのようにドライブ名を指定することができます。具体的には、「#!C:/Perl/bin/MSWin32-x86-object/jperl.exe」のように記述します。ドライブ名がなければ、Cドライブをデフォルトとします。

[ftp2site.pl]

```
#!/Perl/bin/MSWin32-x86-object/jperl.exe
# Para = $f
# Title = ftp2nifty転送

use strict;
use Net::FTP;
my $ftpsite = 'your.ftp.site';# your.ftp.siteにftpサイト名を記載してください。
my $home = '/yourdirectry';# ディレクトリを指定してください。
my $ftp = Net::FTP->new($ftpsite);
print "${ftpsite}に接続しました。¥n";
$ftp->pasv();
$ftp->login('ftpaccount','password');# ftpのアカウントとパスワードを記載してください。
print "loginしました。¥n";
$ftp->cwd($home);
print "${home}ディレクトリに移動しました。¥n";
$ftp->ascii();
print "asciiモードにします。¥n";
$ftp->put($ARGV[0]) or die "put failed ", $ftp->message;
print "$ARGV[0]を転送しました。¥n";
$ftp->quit();
print "FTPを終了しました。¥n";
sleep(1);
```

スクリプトは、Zedのインストールディレクトリにあるmacro¥exec¥menu¥edit のディレクトリに保

存します。今のところ、文書編集後、保存してから実行する必要があります。実行する時は、「編集」のプルダウンメニューに表示される「ftp2nifty 転送」をクリックします。メニューのタイトルは「スクリプトの特別ルール」の「title」に記述しておきます。置換処理をしない場合の標準出力は、コマンドラインのサブウインドウに出力されます(図 1)。

編集しているテキストの選択範囲を処理し処理結果で置換する場合の「スクリプトの特別ルール」は、例えば次のように書きます。

```
#!/Perl/bin/MSWin32-x86-object/jperl.exe
# Title = ddl 編集
# Input
# Output
```

IV. Zed への期待

Zed は現在 α 版、編集できる文字コードも SJIS だけです。しかし、文字コードの対応は今後拡大される予定であり、jperl などのスクリプトを編集に自在に使えるだけでなく、アウトライン、ファイル管理、コマンドラインなどのインターフェースをフルに活用できるようになれば、テキスト処理ツールとして今後大変期待できると思います。

エディタは知的生産のためのツールとして今後大きく発展する可能性のあるアプリケーションです。なぜなら「編集」こそが、知的生産そのものであるからです。Web が登場して、編集するテキストが HTML や XML/RDF のようなテキストとなったことによって、その可能性が飛躍的に増大したと考えています。

Zed に関心を持たれた方は是非使ってみてください。

(2008 年 11 月 30 日)

図 1. Zed のスクリプト実行画面と「編集」メニュー

編集後記

さて、ようやく編集も終わりに辿りついた。投稿していただいたみなさんの御蔭である。今回から、編集にはOpenOffice.org 3.0を使用させていただいている。PDFに変換して、リンクが生きていれば動作確認終了となる。

ようやく、3号に到達。遂に第3号というわけだ。次号、第4号は来年、2009年2月刊行予定である。予定通り出せるかどうかだが、お正月を間に挟むから、ネタはなんとか仕込めるかもしれないね。そういう意味では11月号がもっともつらい月かもしれない。

スクリプティング言語の世界では、PythonはPython3000を、PerlはPerl 6/Parrotを完成させつつあり、Rubyでは1.9の開発が進んでいる。僕はあまり追いかけていないけど、Tcl/Tkも着実に進んでいるのではないかと思う。現在8.5がリリースされたところ。今後はTcl9.0を目指すだろう。GUIはMVC(Model-View-Controller)のViewを司るだけにスクリプタにとって目は離せないのではないか。もっともGUIはWebブラウザに主役を奪われつつあるのかもしれない。アプリケーションはWebブラウザを包含しつつ、デスクトップのリソースを融合する方向に進むはずだと思う。そして、編集機能が加わる必要があるのだけど。

個人的にはWemoやデスクトップCGIのプロジェクトがあるんだけど、もう少し何かおもしろいアイデアが欲しい。取りあえずやりたいことはやろうと思えば何でもできちゃうわけなんだが、もっと他の領域で興味が出てきてしまうので、作業時間がない。少しずつでも、おもしろいことを探してやりたいと思っている。やりたいことをやるのが原則。無理はせずに、意味のあること、やりたいことを生み出していくように進める。

「TSNETスクリプト通信」は執筆者がほぼ固定してきたのだが、新しい参加者も歓迎である。自分の興味のある範囲、できる範囲で参加していただければと思う。

TSNETにも書いたが、ニュースのページを作りたいと思うので、来年の2月前半ぐらいに400字を目安にこんなことがあるよという感じでよいので、TSNETに書き込んでいただいてもよいし、私宛にメールで送っていただいても良い。そうそう、忘れていたけど、表紙ページの写真・イラストなども募集中です。

それでは、Have a fun.

2008年11月30日
jscripter

ＴＳＮＥＴスクリプト通信

2008年11月30日　　1.3.003版発行

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 mf-nokuchi2pho dot ne dot jp
機械伯爵　kikwai at livedoor dot com
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 3.0 Writer

発行所(一次配布所)

TSNETWiki ： 「TSNETスクリプト通信」のページを参照のこと
