```
--> /* ［数値処理］→［自動的に数値で出力］*/
    if numer#false then numer:false else numer:true;
    /*浮動小数点の桁数=7に指定 */
    fpprec : 7;

--> /* 浮動小数点表示桁数を7に指定*/
    fpprintprec:7;

--> /* 1.2.1 */
    x:4;

--> x /*x を出力する*/;

--> y : -3.2 /* y に -3.2 を代入する */;

--> x+y  /* x と y の加算を行う */;

--> x^10 /* xの累乗を計算する */;

--> x^100;

--> x^1000;

--> /* %i=虚数単位 */
    1+2* %i;
    (1+2* %i) * (2+%i);
    expand(%) /* 直前の結果を展開
    expand(式)で式を展開，%で直前の結果を引用*/;

--> y : "character" /* y に文字列 characterを代入，文字列は" "で囲む*/;
    y;

--> z : "pasted" /* z に文字列pastedを代入 */;
    concat(y, z) /* concat(引数1, 引数2,...) 引数を連結する */;
    concat(y, " ", z); past;

--> /* 問解答例*/
    /* 1 */  (3 * 4 + 3.5)/3.3;
    /* 2  */ (1 + 3 * %i) * (3 + 3 * %i); expand(%);
    /* 3 */ concat(My,  " ", name, " ", is, " ", Taro);

--> kill(all);
```

```
--> /* 1.2.2 ベクトル */
    /* リスト[]で行ベクトルを生成する．*/
    x: [0,1,2,3,4]; x;

--> y : ["tokyo", "kyoto"]; y;

--> x[2] /* ベクトル(リスト)xの 2 番目の要素を取り出す */;

--> y[1] /* ベクトルyの 1 番目の要素を取り出す*/;

--> makelist(x[i], i, [1,3,5]) /* xの 1,3,5 番目の要素をまとめて取り出す */;

--> x : [1,2,3]; y:[3,4,1];
    x *y  /* 要素ごとの積 */;

--> x.y /* 内積 */;

--> x: makelist(0+0.5*i ,i, 0, 6) /* 0 から 3 まで 0.5 刻みの数列からなるリストを作る　*/;

--> x: makelist(0+3/4*i ,i, 0, 4) /* 0 から 3 まで長さ(要素数) 5 の数列からなるリストを作る*/

--> z :makelist(i,i, 1, 10) /*1 から 10 まで 1 刻みの数列からなるリストを作る */;

--> z : makelist(1,8) /* 1 を 8 回繰り返す */;

--> y : makelist([1,2,3], 3) /* [1,2,3] を 3 回繰り返す */;

--> length(x) /* リスト x の長さ（要素数）を返す*/;

--> y : makelist(i, i,2 , 5);

--> z : append(x, y) /* リスト x の末尾にリストyを追加*/;

--> reverse(z) /* ベクトルを反転する */;

--> /* 1.2.3 行列 */
    /* 行列は，同じ長さのリストを列ベクトルとして，
    matrixで作成する．*/
    y : matrix([1,3,5,7], [2, 4, 6, 8]);
    y : matrix([1,5], [2, 6], [3, 7]);

--> /* 行列は，列ベクトルを addcolを用いて結合しても生成できる*/
    /* transpose ベクトル，行列の転置を返す */
    /* transpose(リスト) 列ベクトルを生成する */
```

```
--> x :  makelist(i ,i, 1, 4); y: makelist(i ,i,5, 8);
    addcol(transpose(x),transpose(y));

--> /* addrowは行列に行ベクトルを追加する */
    addrow(transpose(transpose(x)), transpose(transpose(y)));

--> /* 注　addrow(x,y) はエラー :·リストに行ベクトルは追加できない
    tranposeで転置を行うと，行列をみなされる*/
    addrow(x,y);

--> addcol(matrix([1,3], [2,4]), matrix([5,7],[6,8])) /* 行列同士も結合できる*/;
    addrow(matrix([1,3], [2,4]), matrix([5,7],[6,8]));

--> entermatrix(3,3)
    /* entermatrix(m,n) 対話的に要素を読み、 m行 n列行列を返す */;

--> ident(3) /* 単位行列の生成 */;

--> x:matrix([1,5,9,3], [3,7,11,15]);

--> x[2,3]  /*2 行 3 列を取り出す*/;

--> row(x, 2) /*2 行目を取り出す*/;

--> col(x,3) /* 3 列目を取り出す */;

--> rank(x) /* 階数を返す */;

--> mat : matrix([1,0.2,0.3], [0.2,1,0.5] , [0.3,0.5,1]);
    x: [1,0.4,0.3];
    mat * x /* 各要素ごとの掛け算が出力される. */;
    mat.x    /* 積　*/;
    xyz : transpose(matrix(x, [2,0.5,1.3], [1,5,0.2]));
    mat.xyz;

--> killa(all);

--> /* 1.2.4 リスト */

--> equity:[[1,2,3], [100,300,240], ["muji"]]
    /* 三つの要素からなるリストequityを生成*/;

--> equity[1]; equity[2]; equity[3];
```

```
--> month:1; price:2; brand:3 /* リストの項目に名前をつける */;
    equity[month] /* equity から month を取り出す */;
    equity[price] /* equity から price を取り出す */;
    equity[brand] /* equity から price を取り出す */;
    equity[price][3] /* equity の price の 3 番目の要素を取り出す */;

--> equity[month] /* 取り出した要素が数値ならば,ベクトルとして抽出される */;
    equity[month]*3 ;
    equity[month] * [3,2,3] ;
    equity[month] . [3,2,3] ;
    equity[month][3]*[3,2,3] ;
    killa(all)

--> /* 1.3 関数 */
    /* 関数を定義するには,
    関数名(引数):=式
    式が複数の場合, (式1, 式2,...,式n)とする */
    /* 半径 r の円周を計算する関数 */
    enshu(r):=2 *%pi*r;

--> enshu(1);

--> enshu([1,3,2]) /* 引数にベクトルを指定することもできる */;

--> enshu(matrix([3,3], [1,2])) /*引数に行列を指定することもできる*/;

--> enshu (3+3* %i) /* 複素数に対しても演算が定義される */; expand(%);

--> moji(x):=concat(x, " ", "goo");

--> moji ("ha");

--> moji (3);

--> menseki(r):=%pi * r^2;

--> menseki(3);

--> is_odd (n) :=(
     if mod(n, 2)=1 /* mod(x,y)は xをyで割った余りを返す */
      then print(TRUE)
        else
            print(FALSE)
    );
```

```
--> is_odd(4); is_odd(5);

--> /* 1.4.1 if 文 */
    /* if  条件式 then 式1
       if  条件式 then 式1 else 式2
       if  条件式 then 式1 else if  条件式 then 式2
       if  条件式 then 式1 else if  条件式 then 式2 else 式3 */

--> x:3;

--> if x=3 then x:4;  x;

--> if x=3 then x:4 else x:2;  x;

--> if x=3 then x:4 else if x=2 then x:2 else x :1; x;

--> /* 1.4.2 繰り返し処理( do 文 ) */
    /*
      for 変数 : 初期値 step 増分 thru 終了値 do (処理1, ..., 処理n)
      ただし，step=1 は省略可
      処理が複数の場合は，(処理1, ..., 処理n)とする
      while 条件 do 処理  ( 条件成立 => 処理 )
      unless 条件 do 処理 ( 条件不成立 => 処理 )
       for分とwhile, unlessを組み合わせて使うこともできる
      /for 変数 : 初期値 step 増分 while 条件 do 処理  ( 条件成立 => 処理 )
      for  変数 : 初期値 step 増分 unless 条件 do 処理 ( 条件不成立 => 処理 )
    /*

--> acc(n):=(
     tmp :1,
     for i:1 thru n do tmp : tmp*i,
     print (tmp)
    );

--> acc(3); acc(0);

--> acc(n):=(
     tmp :1,
     i :1,
     while i <=n  do (tmp : tmp*i, i:i+1),
     print(tmp)
    );

--> acc(3); acc(0);
```

```
--> acc(n):=(
     tmp :1, i:1,
     unless i > n  do (tmp : tmp*i, i:i+1),
     print(tmp)
    );

--> acc(3); acc(0);

--> /* 問　解答例 */
    sum_1(n):=(
     tmp:0,
     for i:1 thru n do tmp : tmp + i,
     print(tmp)
    );
    sum_2(n):=(
     tmp:0,
     for i:1 while i<=n do tmp : tmp + i,
     print(tmp)
    );
    sum_1(10); sum_2(10);

--> kill(all);

--> /* 1.5 グラフィクス */
    /* グラフの作成は，描画パッケージdrawを
    読み込んだあと，2次元であれば，draw2d
    3次元であれば，draw3dを用いる.
    drawはgnuplotへのインターフェイスを提供する */

-->  /* (x,y)平面上の座標点グラフ */
    /* データ準備 */
    /* 標準正規乱数を１０個発生させる */
    /* ramdom_normal(m,s,x)
       平均m, 標準偏差s正規乱数をx個返す
       ただし，事前にdistribパッケージ(所定の確率分布を収録)を読み込む */
    load(distrib);
    X: random_normal(0,1,10);
    Y: random_normal(0,1,10);

--> /* 結果を(x,y)座標データリストにする */
    data_iX:makelist([i, X[i]], i, 1, 10);
    data_XY:makelist([X[i], Y[i]], i, 1, 10);
```

```
--> load(draw);
    /*(x,y)座標点の描画 */
    /* draw2d( points((x,y)座標点リスト), オプション) */
    draw2d(points(data_iX),
    xlabel="Index", ylabel="X", /* x軸ラベル, y軸ラベル指定*/
    title="Std. Normal Random Numbers");

--> draw2d(points(data_XY),
    xlabel="X", ylabel="Y",
    title="Bivariate Std. Normal Random Numbers"
    );

--> kill(all)
    /* 関数グラフ */
    /* 陽関数f(x)を区間x in [a,b]の範囲で描くには,
    draw2d(explicit(f(x), x, a, b), オプション) */;

--> draw2d(
    explicit( 2*x^2 -5*x, x, 0, 5),
    xlabel="x", ylabel="y",
    title="y=2x^2 -5x");

--> draw2d(
    explicit( log(x), x, 0, 10),
    xlabel="x", ylabel="log(x)",
    title="y=log(x)",
    yrange=[-3,3], /* y軸範囲指定 */
    xaxis=true /* x軸描画*/);

--> /* グラフの重ねあわせ */
    /* draw2d内に描画コマンドを重ねる */
    draw2d(
    key="x^3", /* 凡例 */
    explicit( x^3, x, -2, 2),
    key="x^2",
    color=red, /* 色指定 */
    explicit(x^2, x, -2, 2),
    xlabel="x", ylabel="y",
    title="Superpostion of Figures",
    xaxis=true, yaxis=true /* x軸y軸描画 */);
```

```
--> /* ヒストグラム */
    /* 記述統計パッケージ descriptiveを読み込んだ後,
     histogram(データ, オプション)
    あるいは，draw2d内で
    histogram_description(データ,オプション) */

--> /* データ */
    load(distrib);
    X : random_normal(0,1,10);
    /* ヒストグラム */
    load(descriptive);
    histogram(
    X,
    nclasses =[-4,4,8]
    /* ヒストグラムの境界を-4と4に、階級数を8に設定
    境界値を省略すると，[最小値，最大値]を階級数で分割
    デフォルト階級数=10 */,
    xlabel="X", ylabel="Frequency",
    title="Histogram of X",
    ytics=1 /* y軸目盛間隔=1 */
    );

--> /* ヒストグラム draw2d*/
    load(descriptive);
    load(draw);
    draw2d(
    histogram_description( X, nclasses =[-4,4,8] ),
    xlabel="X", ylabel="Frequency",
    title="Histogram of X",
    ytics=1 /* y軸目盛間隔=1 */
    );

--> kill(all);
```

```
--> /* 3次元グラフ */
    /* 独立な2次元標準正規分布密度関数 */
    load(distrib);
    gauss3d(x,y):=pdf_normal(x,0,1) * pdf_normal(y,0,1)
    /* pdf_normal(x,m,s) 平均m，標準偏差sの正規分布密度関数 */;
    load(draw);
    draw3d(
     explicit(gauss3d(x,y), x, -4,4, y, -4,4),
    /* dimensions=[幅, 高さ]
    デフォルト値[600, 500] */
    surface_hide =true /* 隠れた部分をプロットしない */,
    /* enhanced3d=true, 色付けオプション */
    /* view=[x軸回り垂直回転角度, z軸りの水平回転角度]
    [0, 360]の範囲で指定 デフォルト値[60, 30]*/
    xlabel="x", ylabel="y", zlabel="z",
    title="z=1/(2pi)e^{-(x^2+y^2)/2}"
    );

--> /* グラフのファイルへの書き出し */
    draw3d(
     file_name ="fig" /* ファイル名 拡張子無*/,
     terminal = eps_color /* ファイル形式 eps, pdfならば pdfなど*/ ,
     explicit(gauss3d(x,y), x, -4,4, y, -4,4),
     dimensions=25* [21, 29.7] /* A4縦1/4 単位は0.1mm*/,
     surface_hide =true,
     xlabel="x", ylabel="y", zlabel="z",
     title="z=1/(2pi)e^{-(x^2+y^2)/2}"
    );

--> /* データの書き出し，読み込み　*/

--> /* 書き出し用行列データ*/
    mat : matrix([1.0, 0.2,  0.3], [0.2, 1.0, 0.5], [0.3, 0.5, 1.0]);
    /* write_data(X, "ファイル名")
       write_data(X, "ファイル名", separator_flag)
      separator_flagは，comma, csv, pipe, semicolon, space */
    write_data(mat, "matrix.txt");
    write_data(mat, "matrix.csv", csv);
```

```
--> /* 行列データの読み込み */
    read_matrix("matrix.txt");
    read_matrix("matrix.csv");
    /* データのリストでの読み込み */
    read_list("matrix.txt");
    read_list("matrix.csv") /* ファイル名の後に, separator_flagを付けても良い */
```