

# 1 基礎

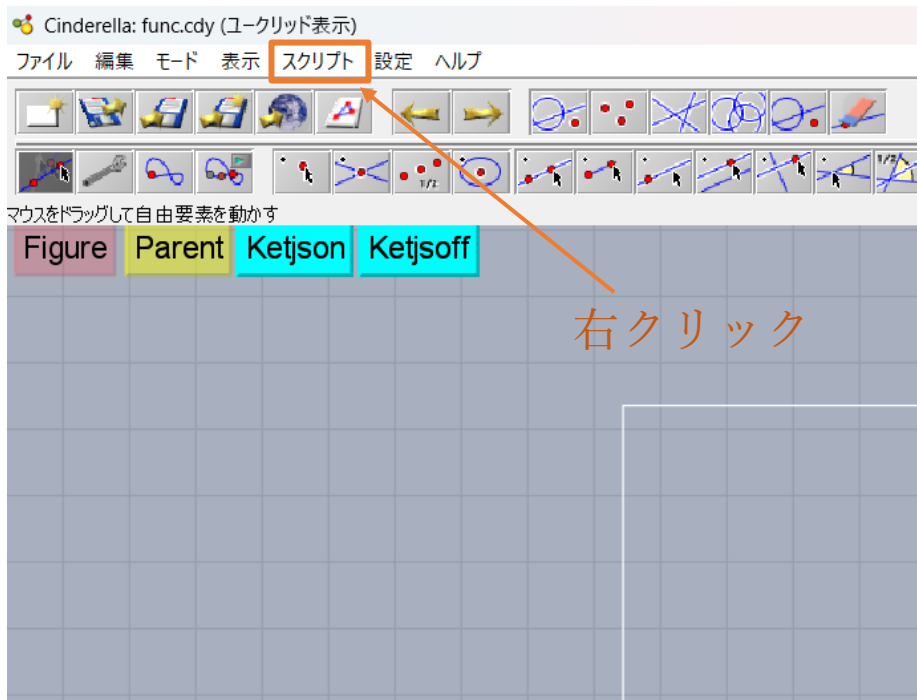


図 1-1 スクリプトの場所

Cinderella では、上の「スクリプト」からプログラムを書くことができます。  
スクリプトを右クリックすると、以下のようなウィンドウが表示されます。

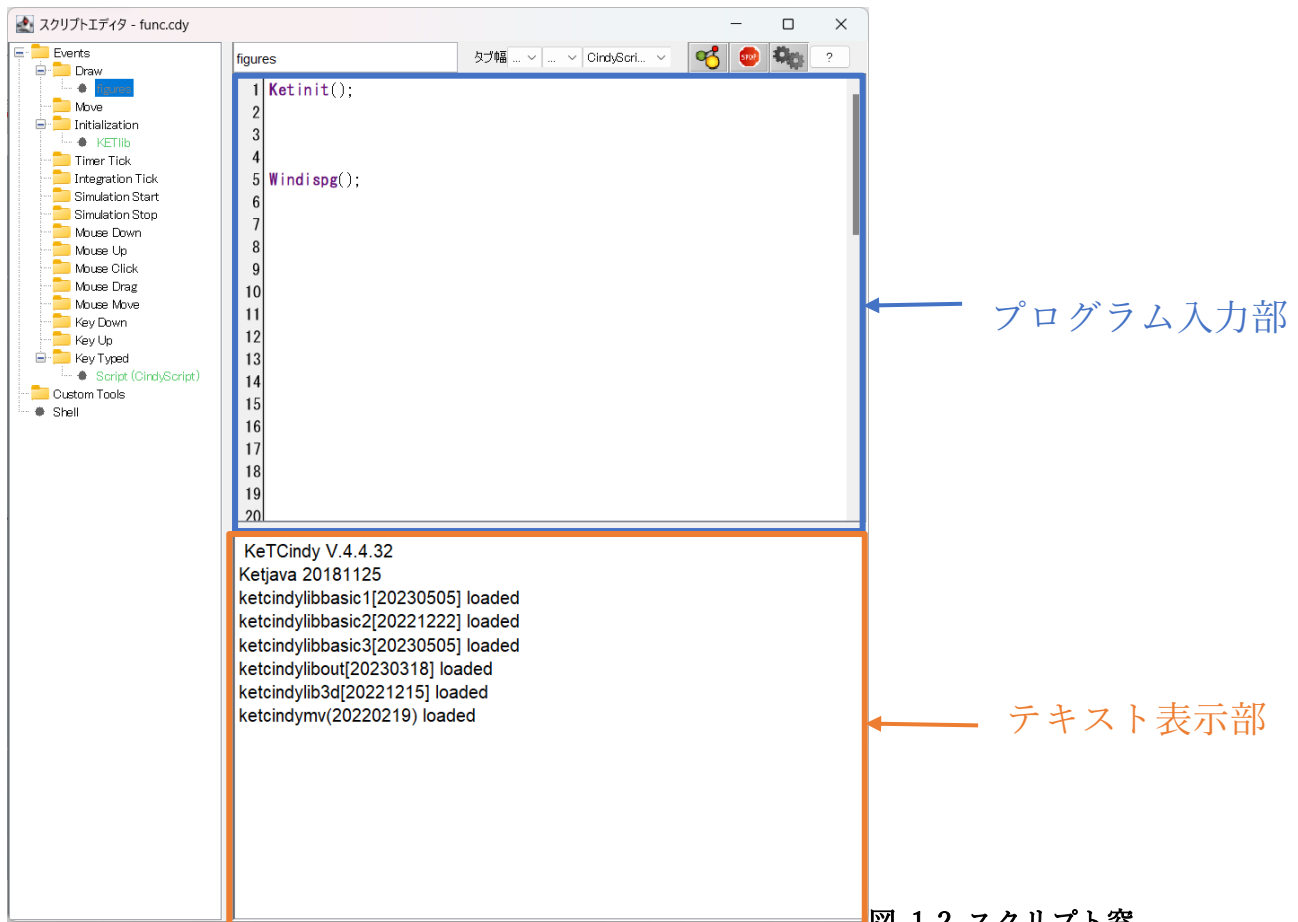


図 1-2 スクリプト窓

このプログラム入力部の Ketinit();と Windispq();の間にプログラムを書いています。  
書いたプログラムを適用させたいときは、

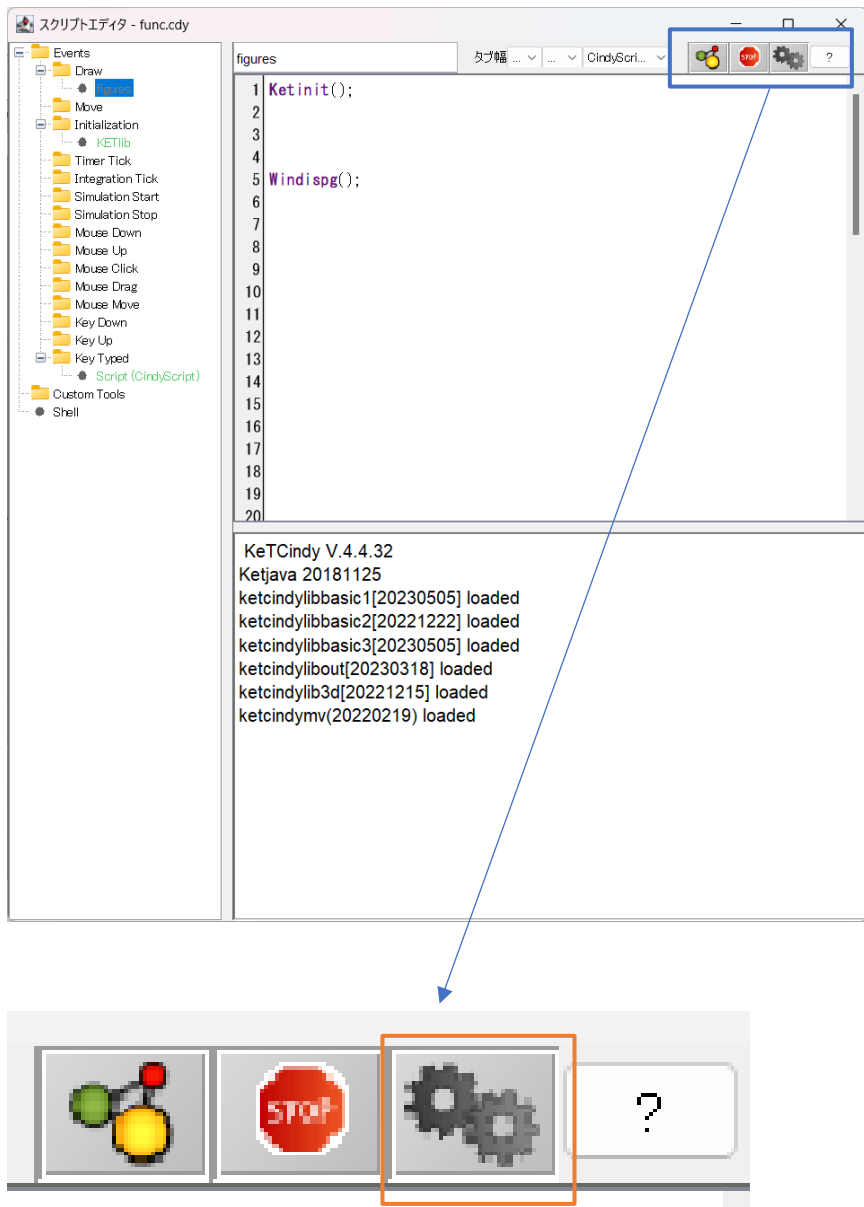


図 1-3 プログラム適用

このボタンの中の左の歯車のボタンを押すことで適用できます。

また、中央の Stop ボタンを押すことでプログラムを停止させることができます。

## 2 Cinderella の四則演算と $\alpha$

足し算：+                      例： $2 + 3 = > 2+3$

引き算：-                      例： $2 - 3 = > 2-3$

かけ算：\*                      例： $2 \times 3 = > 2*3$

割り算：/                      例： $2 \div 3 = > 2/3$

注意：  $3/2*3$  等 割り算の後掛け算を行うとき、この結果は  $\frac{3}{2 \times 3} \dots \textcircled{1}$  にはなりません!

結果は  $\frac{3}{2} \times 3$  になります!

式①のような処理をさせたいときは  $3/(2*3)$ 、 $3/2/3$  等と書くことができます。

階乗    : ^                      例： $3^2 = > 3^2$

### 3 Print 関数

構文： `Print(表示させたいもの);`

例： `Print("みかん");`

上記例では「みかん」を1 ページ目の **テキスト表示部** に出力します。

<pre>1 Ketinit(); 2 3 Print("みかん"); 4 5 Windispg(); 6</pre>	<pre>KeTCindy V.4.4.32 Ketjava 20181125 ketcindylibbasic1[20230505] loaded ketcindylibbasic2[20221222] loaded ketcindylibbasic3[20230505] loaded ketcindylibout[20230318] loaded ketcindylib3d[20221215] loaded ketcindymv(20220219) loaded みかん</pre>
---	---

図 3-1 Print 関数ソースと結果

変数も表示することができます。

例： `a=100;`

`Print(a);`

`Print(" a は" + a + "です。");`

<pre>1 Ketinit(); 2 3 a = 100; 4 Print(a); 5 Print(" a は" + a + "です。"); 6   7 Windispg(); 8</pre>	<pre>KeTCindy V.4.4.32 Ketjava 20181125 ketcindylibbasic1[20230505] loaded ketcindylibbasic2[20221222] loaded ketcindylibbasic3[20230505] loaded ketcindylibout[20230318] loaded ketcindylib3d[20221215] loaded ketcindymv(20220219) loaded 100 aは100です。</pre>
---	--

図 3-2 Print 関数変数表示ソースと結果

このように変数とテキストを同時に出力することもできます。

また、改行を最後に行う `Println` 関数もあります。

## 4 Plotdata 関数

構文： `Plotdata(グラフの名前(テキスト), 式(テキスト), 変数名(と定義域)(テキスト), [オプション]);`

例： `Plotdata("1", "2*x", "x");`

上記例では  $y = 2x$  を Cinderella のグラフ上に gr1 として表示することができます。

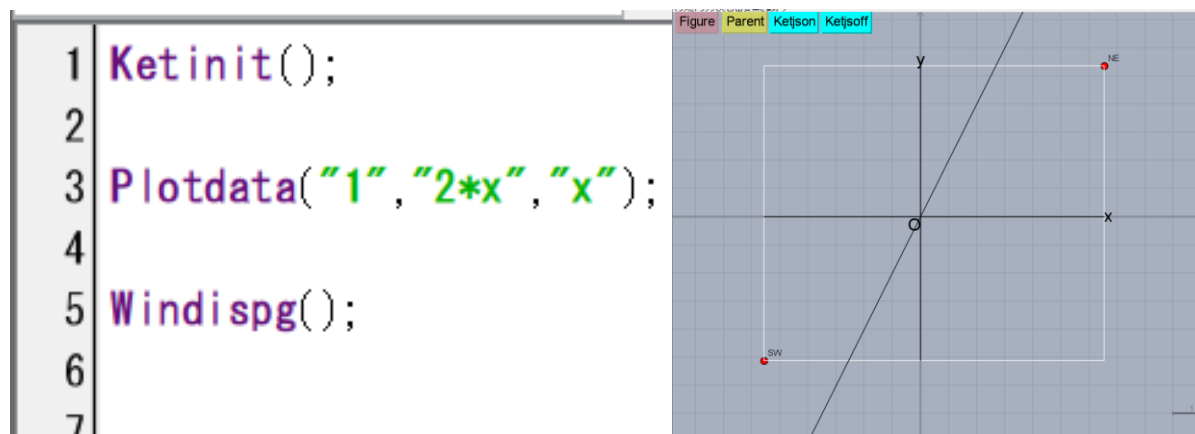


図 4-1 Plotdata 関数ソースと結果

### Tips

$y = 2x$  のグラフを書く際、 $2*x$  のように  $*$  を忘れないように注意。

グラフの名前を同じにすると、グラフを書き換えることができます。

例： `Plotdata("1", "2*x", "x");`

`Plotdata("1", "-2*x", "x"); // このグラフが表示される`

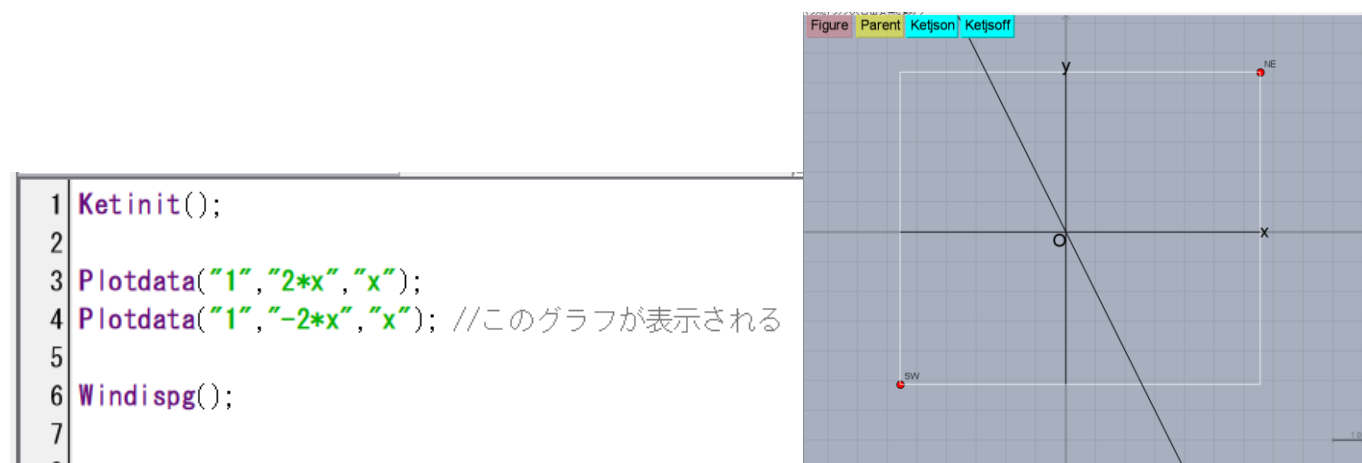


図 4-2 Plotdata 関数更新ソースと結果

オプションの欄には `["Dis = 100", "Num=200"]` のように入力することができます。

オプションの種類

**Dis**： 不連続点を結ばないようにする。

**Num**： 解像度。大きいほどきれいにできるが、動作が重くなる。

**Exc**： 指定した点を除外する。

## 5 Slider 関数

構文: `Slider(点の名前(テキスト), 一端の座標, もう一端の座標);`

例: `Slider("A", [-5,-3], [5,-3]);`

上記例では、A を(-5,5)~(5,-5)の半直線を移動するスライダーとします。

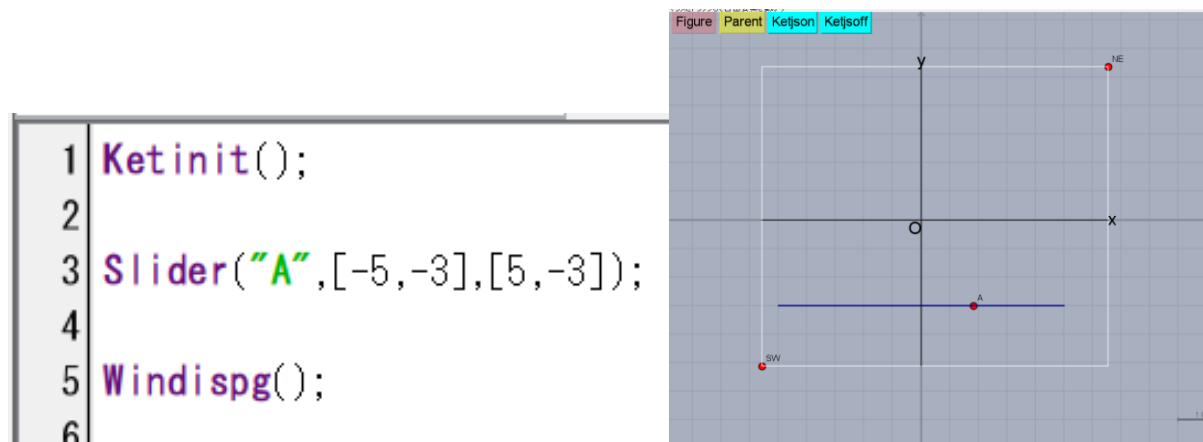


図 5-1 Slider 関数ソースと結果

### Tips

点の名前.x を使うことでスライダー上の点の x 座標を取得することができます。

例: スライダー上の A 点が(3,-5)にある場合

A.x                    取得値: 3

点の名前.y を使うことでスライダー上の点の y 座標を取得することができる

例: スライダー上の A 点が(3,-5)にある場合

A.y                    取得値: -5

点の名前.xy を使うことでスライダー上の点の座標を取得することができる

例: スライダー上の A 点が(3,-5)にある場合

A.xy                   取得値: [3,-5]

逆にこれらに値を代入することで点の位置を指定することもできる。(点は作成しておく必要あり)

例: `A.xy=[1,1];`

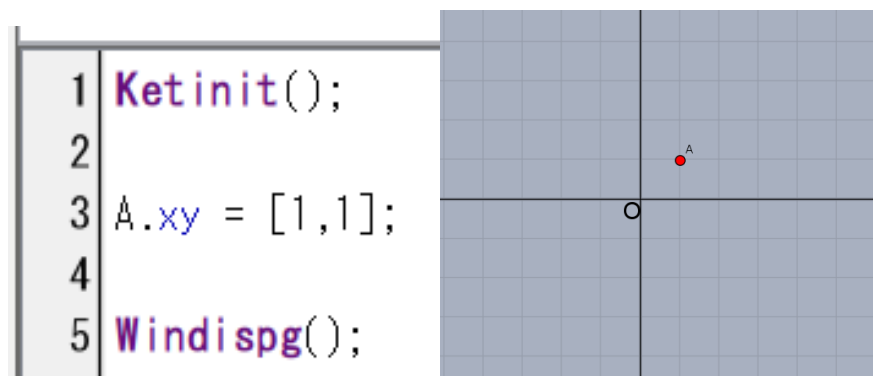


図 5-2 点の場所指定ソースと結果

## 6 Letter 関数

構文： `Letter`(表示したい座標 (点), 方角, テキスト);

例： `Letter(A, "ne", "みかん");`

注： 方角には方位 n(北),s(南),w(西),e(東),c(中央)を使います

上記例では、(作成しておいた) 点 A の ne の方角に「みかん」を出力します。

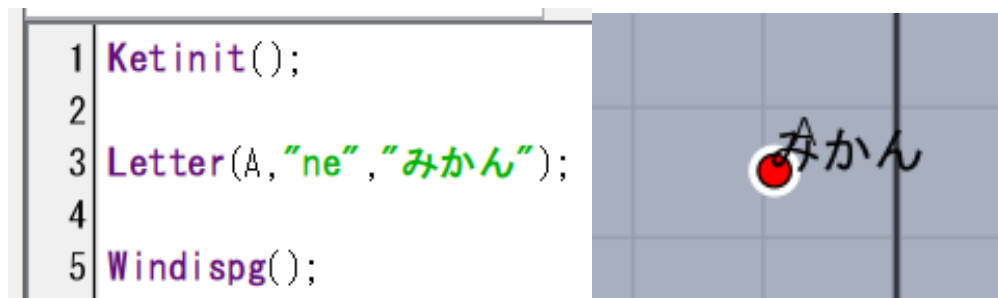


図 6-1 Letter 関数ソースと結果

変数の値も出力することができるため、以下のように変数の値と文字列を点の周りに表示することもできます。

例： `Slider("A", [-5,-3], [5,-3]);`

`Letter(A, "ne", A.x);`

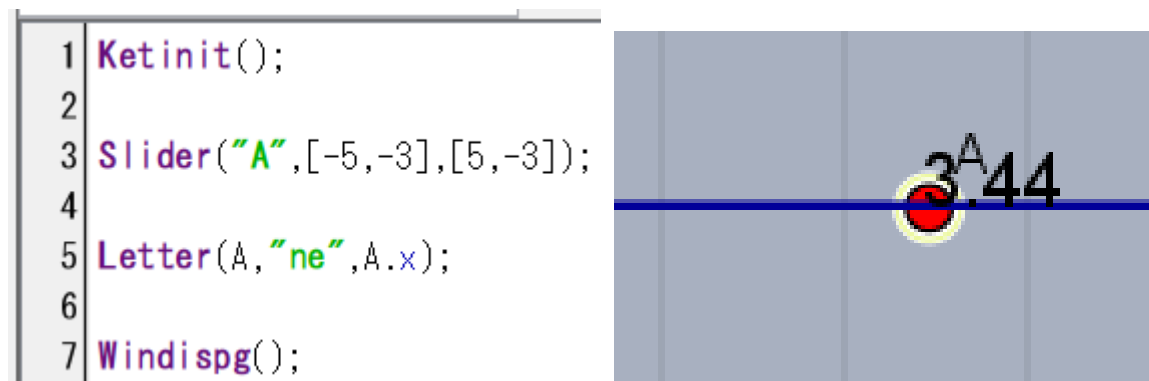


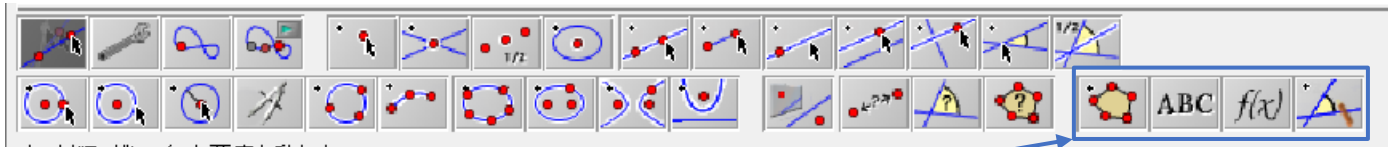
図 6-2 Letter 関数座標参照ソースと結果

## ボタンの作成

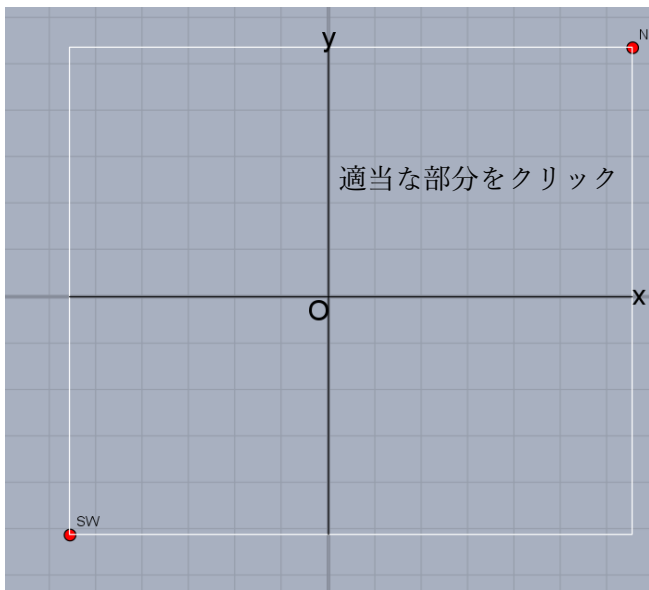
ボタンを作成するときは、

1. ボタンにするテキスト作成
  2. ボタンにする
  3. ボタンを押したときの動作を決める
- の順にボタンを作成します。

### 1. テキストを作成



ABC の部分が適用された状態でグラフの部分をクリック



すると、下のようなウィンドウが出てくると思うので、そこに適当な文字を入力し、OK を押します。

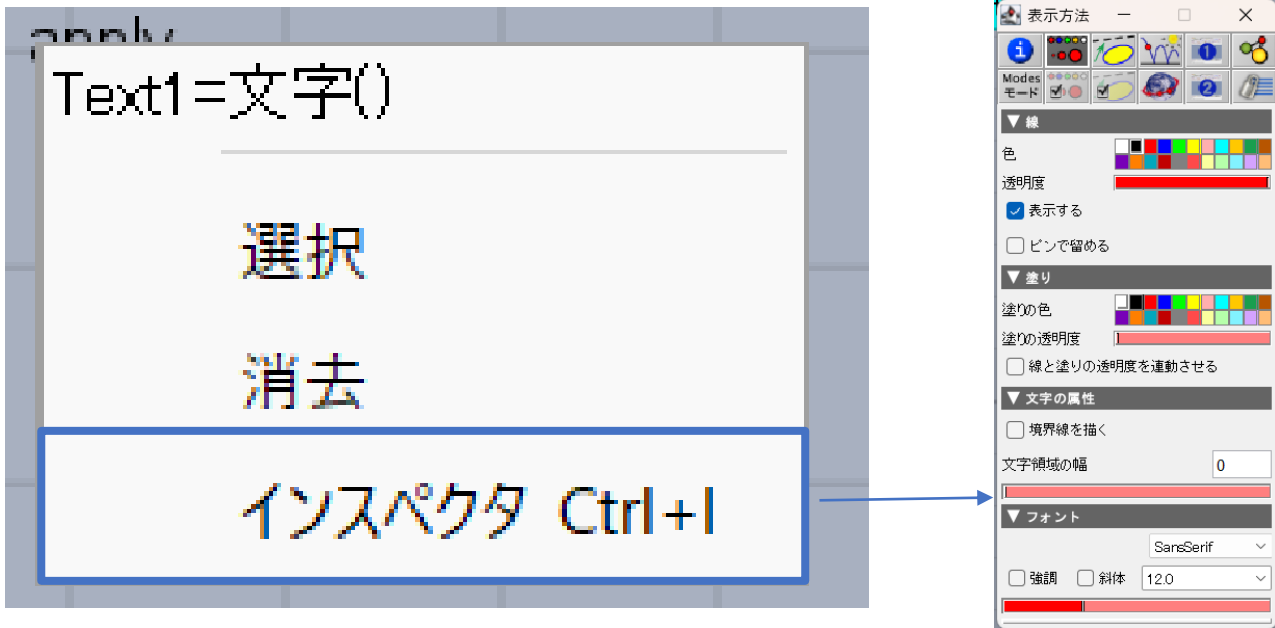


これでボタン用のテキストが作成できます。

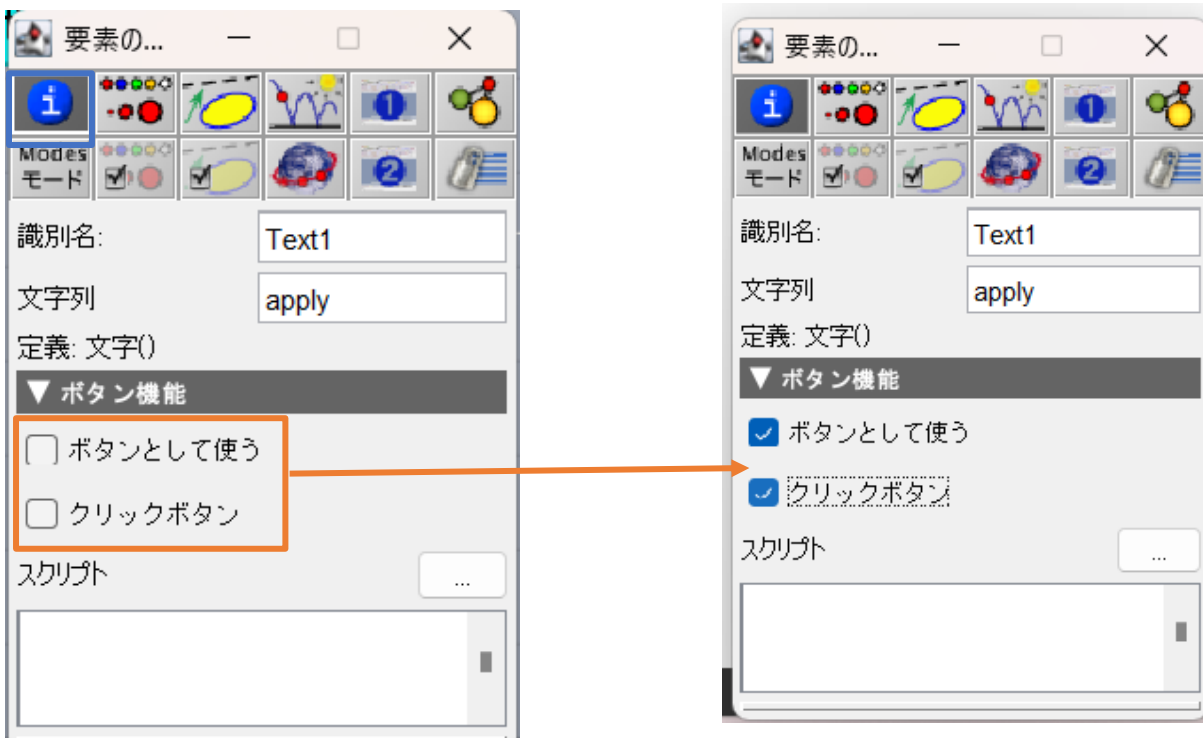


## 2. ボタンにする

次に作成したテキストを右クリックしてインスペクタを表示します。



そして、以下のウィンドウに移動し、ボタンとして使う、クリックボタンにチェックいれます。



そうすると、ボタンが以下のように変遷します。



これでテキストをボタンに変更することができました。

### 3. ボタンにした時の動作を決める

2のインスペクタの画面に映ります



そして下のスクリプトの場所にクリックされたときに実行したいコマンドを入力します。

例： クリックするごとに  $y = x$  と  $y = x^2$  のグラフを交互に表示するプログラム

if 文と初期値の設定について。

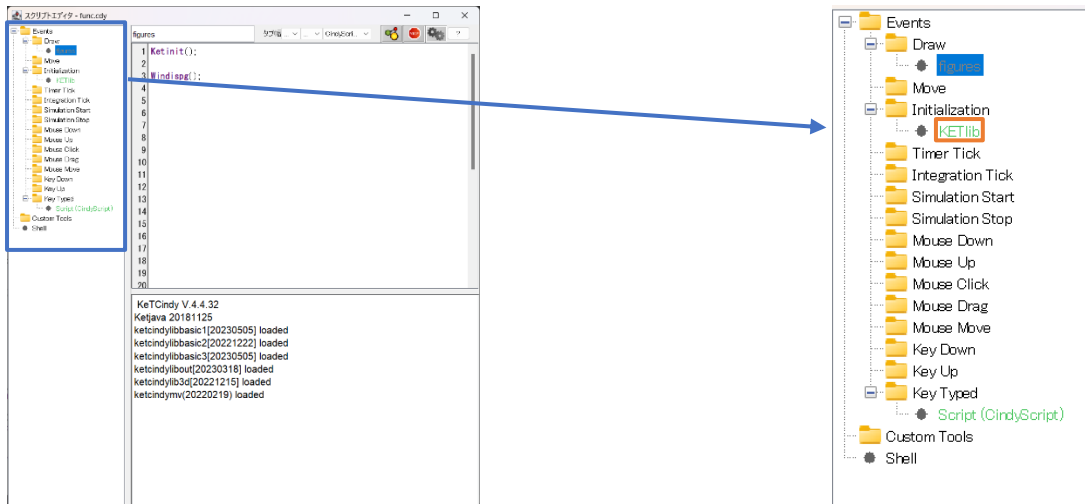
if 文： 指定した条件に合うときに中のプログラムを実行させます。Cinderella での if 文を解説します。

構文： **if(条件式, 実行したい処理, 例外処理);**

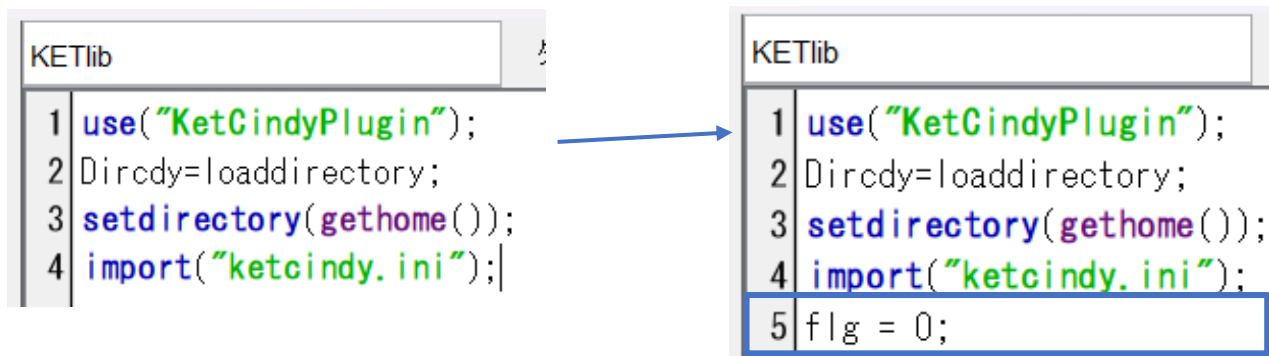
初期値：変数の中に初めに入っていてほしい値を初期値と呼び、それを設定することができます。

初期値は初めに読み込むプログラムの部分で変数に初期値を入れることで設定できます。

スクリプトエディタのファイルの部分の KETlib の部分を開きます。



その後プログラム入力部が以下のようなウィンドウに代わるので、その最後の行で初期値を設定します。

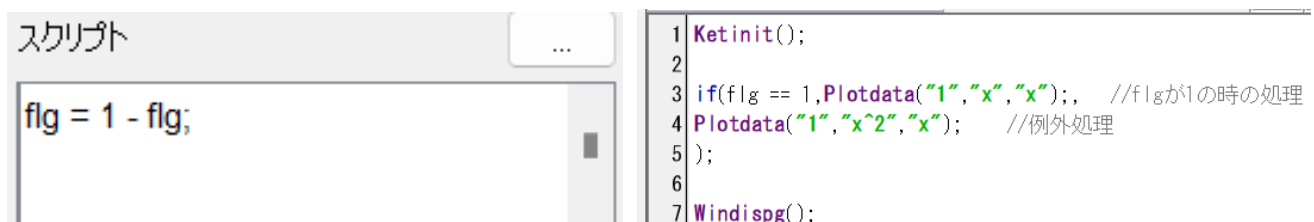


これで  $flg = 0$  と初期値を設定することができました。

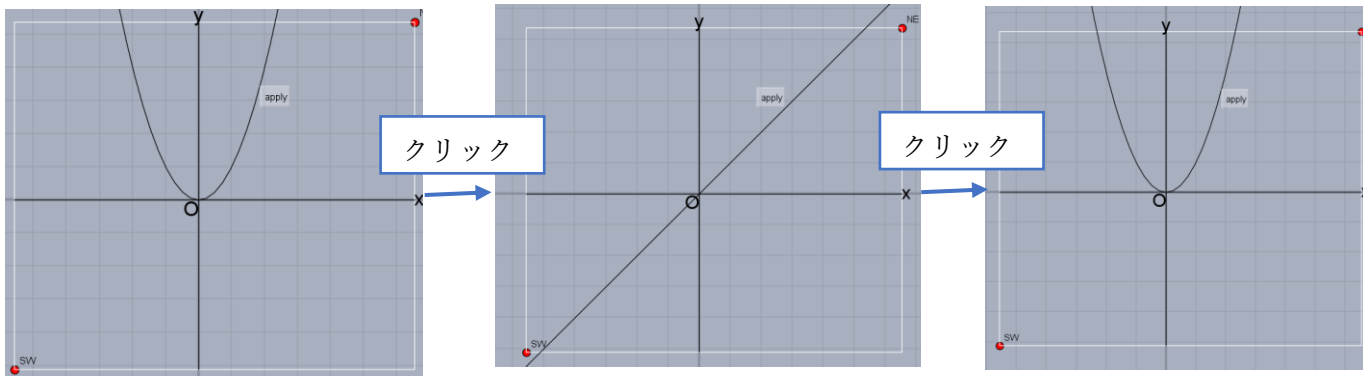
では、今回の例のプログラムについて解説をします。

$flg = 0$  と初期値が設定されていることを前提とします。

以下がボタンのスクリプトとプログラムの例です。



うまくいくと次のようにグラフが変わっていきます

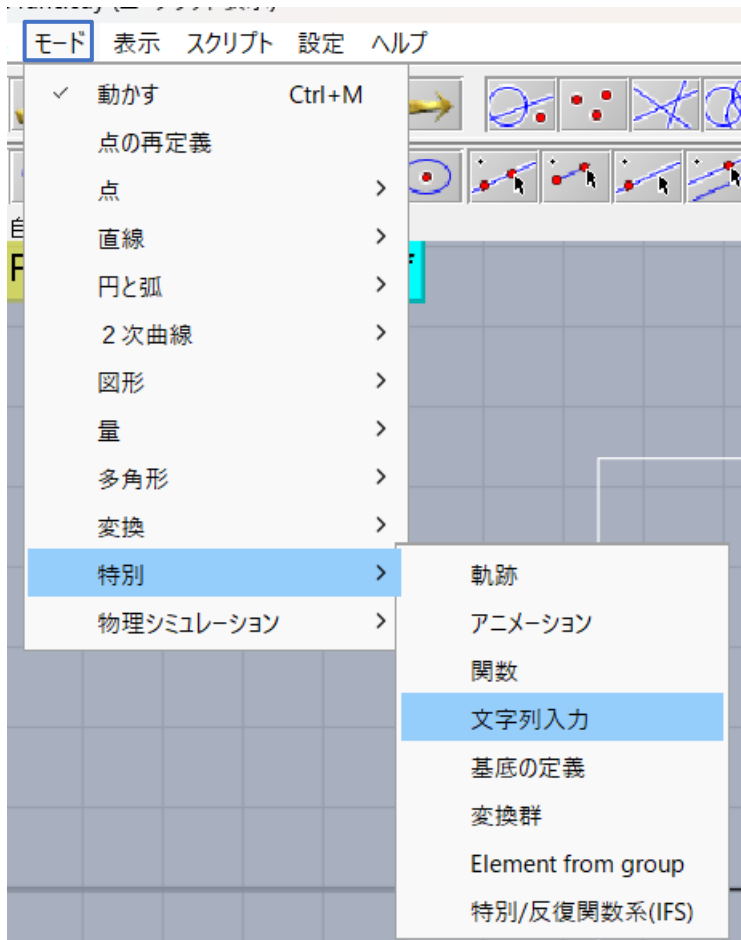


## 入力窓の作成

入力窓を作成する際は

1. Cinderella 上で入力窓を用意
2. KeTCindy で入力窓としての設定を行う
3. KeTCindy で入力窓から値を受け取る

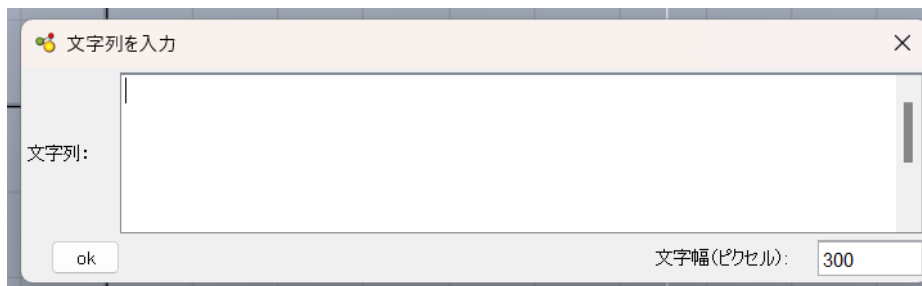
1. Cinderella 上で入力窓を作成



モード → 特別 → 文字列入力 の順に選択していきます。

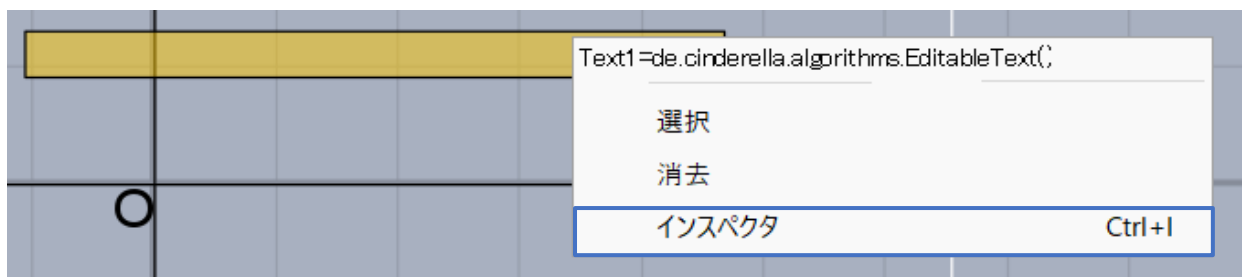
文字列入力を押すと、任意の場所をクリックすると文字列を作成できるモードになります。

任意の場所をクリックすると、以下の画面になります。



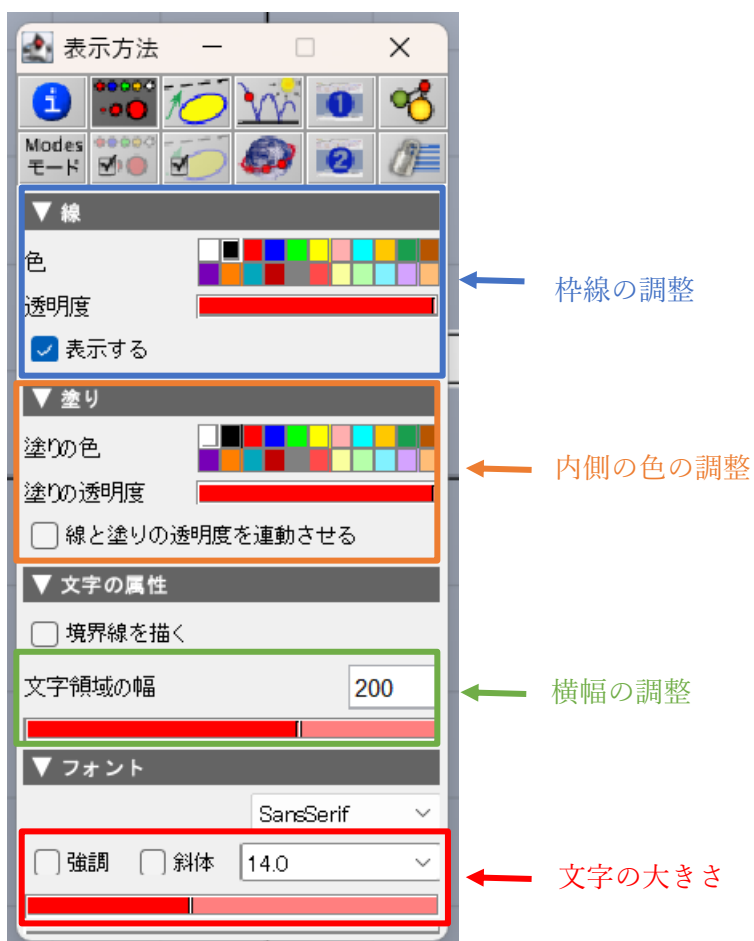
このまま ok を押します。

その後、作成されたものを右クリックし、インスペクタを開きます。



インスペクタで適当に色などを調整します。また、識別番号の確認(設定)も行うので開いたままにしておいてください。

今回は文字を大きくし、白に黒の枠で囲まれた入力窓にしました。



このような入力窓になりました。

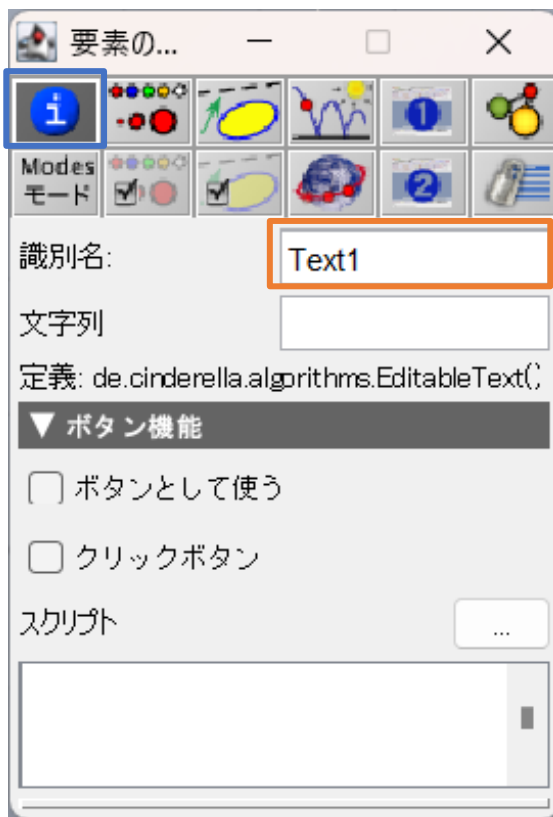


これで Cinderella の入力窓の作成は完了です。

## 2. KeTCindy で入力窓としての設定を行う

KeTCindy で入力窓として読み込むために設定を行います。

初めに識別番号の確認と設定をします。先ほど開いたインスペクタから確認を行います。



ここでは 1 が識別番号となっています。

次にコマンドで入力窓としての設定を行います。

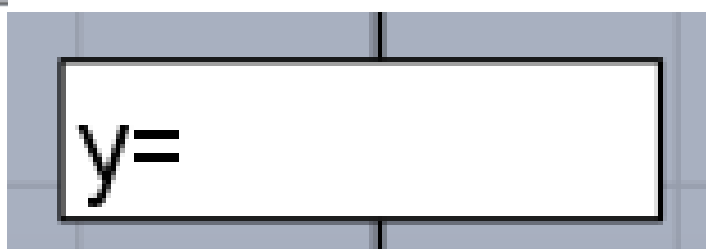
Seteditable コマンド

**Seteditable**(識別番号 , [ 初期文字列, "Size = 文字の大きさ", "Width = 横幅"]);

Size と Width は入力しなくても大丈夫です。入力しない場合は先ほど設定した Cinderella 上での設定が参考されます。初期文字列は空 ("") でもいいので設定してください。

例：

```
1 Ketinit();
2
3 Seteditable(1,["y="]);
4
5 Windispg();
```



今回は「y=」を初期値として設定します。

### 3. KeTCindy で入力窓から値を受け取る

最後に値を受け取ります。

Textedit コマンド

**Textedit**(識別番号 [, 未入力時の値]);

未入力値の値は文字が入力されていないときもしくは「=」の後に文字がないときに返す値を設定します。

デフォルト値は空(“”)でもいいので設定してください。

この関数の返り値として入力された値を得られます。

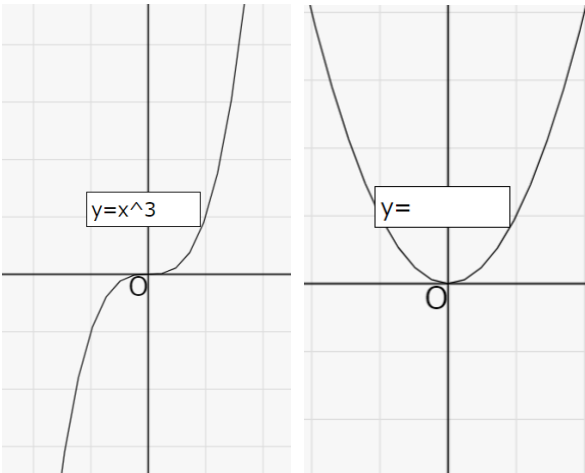
例：

```
1 Ketinit();
2
3 Seteditable(1,["y="]);
4 str = Textedit(1,"x^2");
5
6 Windispg();
```

デフォルト値を「x^2」として str に入力された値を与えています。

最後に応用としてこれまで例として作ったプログラムから、入力された関数を表示するプログラムを作成します。

```
1 Ketinit();
2
3 Seteditable(1,["y="]);
4 str = Textedit(1,"x^2");
5
6 Plotdata("1",str,"x");
7 Windispg();
```



注意： HTML に書き出さないと入力窓を使えません。

Seteditable で初期文字列を設定しないとうまく動作しません。

Textedit でデフォルト値を設定しないとうまく動作しません。

あくまで「=」以下の値を受け取るため、初期値の設定は気を付けてください。