Ti advanced generator

> ### *What is Ti advanced?*
> *Ti advanced is a way to make the cration of Ti-basic programms easyer.*

Structure

- A line of Ti extended is composed of a key, and the arguments needed: `key arguments`
- There is librairies which add keywords. To use them, you have to write: `list.add name value`
- To add a comment, write // Comment, # Comment or /* Comment */
- To create a constant, write `cst type value`. You can create constants for `numbers`, `text`, `list`, `matrix`, `vector`, `boolean`, `color` and `math`.
- To create a function constant, write:
  ```
  cst ! func function
  ---
  code args
  ---
  ```
  (No _ nedded)

Index

Variables

Variables types

Type	Content	Part
int	Number (integer or float)	number
str	Text	text
list	List	list
matrix	Matrix	matrix
pict	Image	picture
vect	Vector 2d/3d	vector
bool	Boolean	boolean
color	Color	color
func	Functions	functions
math	Math functions	math

For every type (except boolean), you can delete a variable using `del type name`.

Number

There is 27 numbers variables available: letters from A to Z and θ (thêta).

Action	Syntax
Declaration	`new int number int value`
Changing value	`set int number int value`
Increasing	`inc int number int value (1 by default)`
Other operations	`calc int number operation (for exemple: 3*{name})`

Text

There is 10 text variables available: from chn0 to chn9.

Action	Syntax
Declaration	`new str string str text`
Changing value	`set str string str text`

List

There is 6 existing list variables: from L_1 to L_6.

There is a pretty infinite amount of undeclared lists: L1, L2...

Action	Syntax
Declaration	`new list list int value`
Changing value	`set list list int value`
Changing item	`list.replace list list int value vector2d position`
Append list	`list.append list list int value`
Inserting item	`list.insert list list int value vector2d position`
Fill list	`list.fill list list int value`
Concat	`list.concat list list list list`

Matrix

There is 10 matrix variables available: from [A] to [J]

Action	Syntax
Declaration	`new matrix matrix matrix matrix`
To list	`set ! list list matrix matrix`
Dimentions	`matrix.dim matrix matrix (-> vector2d)`
Exchange rows	`matrix.swap matrix matrix int row1 int row2`
Add two rows	`matrix.add matrix matrix int row1 int row2 (-> matrix)`
Multiply row with value	`matrix.multiply matrix matrix int row1 int value (-> matrix)`

Picture

There is 10 picture variables available: from Pic0 to pic9
(act on the graph background)

Action	Syntax
Save picture	`picture.save int imgindex`
Show picture	`picture.show int imgindex`
Show background	`sbg int bgindex`
Hide background	`hbg`

Vector

Vector variables are stored in a list.

Action	Syntax
Declaration	`new [vector2d|vector3d] vector [vector2d|vector3d] vector`
Changing value	`set [vector2d|vector3d] vector [vector2d|vector3d] vector`
Get value	`[vector2d|vector3d].get [vector2d|vector3d] name [x|y|z||]`

> *A vector is writen like this:* `[vector2d|vector3d] x y z`
> *Or like this:* `<x, y, z>`

Boolean

Boolean variables are stored in a list.

Action	Syntax
Declaration	`new bool boolean bool boolean`
Changing value	`set bool boolean bool boolean`
Invert	`bool.invert bool boolean`

Colors

To get a color, use `colors.color`

Functions

You can create as many functions as you want, they are coded into the file.

Action	Syntax
Declaration	`def func function list args`
Return	`back type value`
Less related	
Go to	`goto int line`
Execute	`exe str programm`

Math functions

You can create 9 math functions: from Y_0 to Y_9.

Action	Syntax
Declaration	`new math mathfunction math value`
Changing value	`set math mathfunction math value`

Controls

Indentation is replaced by _ *code*

Statement	Syntax
If	`if` *type* *value* *operation* *type* *value*
Elif	`elif` *type* *value* *operation* *type* *value*
Else	*No argument*
For	`for int` *number* `in int` *start* `int` *end* `int` *increment*
Foreach	`list.foreach int` *number* `in` *iterable* *list*
While	`while` *type* *value* *operation* *type* *value*
While not	`!while` *type* *value* *operation* *type* *value*
Pause	`pause` *type* *value*
Stop all	`stop`
Stop subprogramm	`ret`

Interactions

Action	Syntax
Output	`out` *type* *value* `int` *x* `int` *y*
Ask choice	`ask` *type* *question* *type* *choice1* `func` *action* *You can add as many options as you want.*
Input	`inp str` *string* *type* *question* *prompt*
Get key	`keyboard.get (-> int)`
Clear	`clr`
Send var	`usb.send` *type* *variable*

Draw

Action	Syntax
Erase	`draw.clear`
Line	`draw.line vector2d point1 vector2d point2 colors.color`
Horizontal line	`draw.horizontal int position colors.color`
Vertical line	`draw.vertical int position colors.color`
Draw function	`draw.drawfunc function`
Circle	`draw.circle vector2d center int radius colors.color`
Text	`draw.text int line int column str text`
Set pixel	`draw.pixel vector2d position [set|none|edit] colors.color`
Set point	`draw.point vector2d position [set|none|edit] colors.color`

Librairies

Action	Syntax
Import	`with str librairy`
Execute	`librairy`

Included librairies :

- list
- bool
- keyboard
- usb
- colors
- draw
- viewport

Other Ti basic functions

To run another Ti basic function, use `run func function`.
Like this, the function will be directly writen into the programm.