# Standalone functions

Standalone functions always return a value and are therefore used in contexts that expect a value, such as in the right hand side of a variable declaration (var) or assignment (set) statement, either on their own or as part of a larger expression. Standalone functions usually require at least one argument to be passed in brackets – corresponding to the parameters defined for that function.

## Processing strings

Unicode

## Processing numbers

mod

div

floor

round

toPrecision ?? maybe get rid of this now we have round

ceiling

parseAsFloat

parseAsInt

range

## Maths functions

pi

sin

cos

sqrt

TODO – more

## Higher order functions

filter

map

reduce

max

maxBy

min

minBy

any

contains

size

## Initialisation

These two are inconsistent, also inconsistent with new ArrayList(3) say.

initialise2DArrayList

initialisedGraphics

# Standalone procedures

## pause

Pauses the execution of the program for a specified number of milliseconds (minimum 1). For example, to pause for 1/10th of a second:

call pause(100)

There are two uses of pause:

* to control the speed of events – for example in a dynamic game
* to allow the Console and/or Graphics displays to refresh. See Console and Graphics. (For this purpose, pause(1) is sufficient to enable the display refresh and causes minimum delay in program execution).

## clearKeyBuffer

All keystrokes go into ‘queue’ called the keyboard ‘buffer’. If you are reading keys (see Reading keystrokes ) and you wish to prevent the user from adding keystrokes faster than they can be consumed, then:

call clearKeyBuffer()

## print & printTab

These procedures may be called as an alternative to using the print statement. The differences are that the print or printTab *procedure*:

* does not automatically add a ‘newline’ at the end, so you may subsequently print something else on the same line. If you wish to use the print procedure and include one or more newlines in specific places, just include \n (the standard form for a newline) within the string.
* Require the data to be printed to be of type String. If you want to print a value of another type, you will either need to add .asString() to it, or put the value into braces within an ‘interpolated’ string.

For print, the data to be printed is the only argument. For example:

for I from 1 to 10 step 1  
 call print("{i}")  
end for

printTab helps in the layout of information printed to the console, in particular, the printing of columns of data. printTab works like the print procedure, but requires an additional argument specifying the tab position (number of characters from the left of the display). For example:

call printTab(0, "No.")  
call printTab(10, "Square")  
call printTab(20, "Cube\n")  
for x from 1 to 10 step 1  
 call printTab(0, x.asString())  
 call printTab(10, "{x^2}")  
 call printTab(20, "{x^3}\n")  
end for

## clearConsole

Equivalent to pressing the Clear button on the Console, but automatically at specific point(s) in the program execution:

call clearConsole()

# System methods

‘System methods’ refers to a set of specific methods provided by the Elan standard library, that depend on the system (outside of the Elan language) in some way. They *appear* to work like functions – in that they return a value – and may be used in the same way as a regular function, *but may be used only within* main *or a* procedure*.* This is because each system method has a dependency on something more than the arguments (if any) passed into it, and/or generates side effects. Thus, system methods are *not* ‘pure’ functions,

## Input methods

Not yet implemented.

## Clock

The clock methods returns an integer representing the current time in milliseconds since ‘the epoch’ (midnight, January 1, 1970 UTC). This is useful for measuring elapsed time. For example:

var startTime set to clock()  
# Your code here  
print "Elapsed time in milliseconds {clock - startTime}"

## getKeystroke & getKeystrokeWithModifier

Example use:

var k set to getKeystroke()

* If the user has pressed a key that key will be returned as a String.
* If no key has been pressed, the method will return an empty string: "".
* If the key is a printable character, it will be returned as a single-character string, for example: "a", "X","3",":"
* Non-printable keys are returned as words, for example: "Space", "Backspace","Enter","ArrowDown"
* getKeystrokeWithModifier allows you to check whether the keystroke was ‘modified’ by, for example, the **Shift**, **Ctrl**, or **Alt** keys. The method returns a 2-tuple consisting of the key and the modifier (which may be an empty string). Learn how this works with this example:

See also clearKeyBuffer

## Random numbers

random

randomInt

# Dot methods

## On String

substring

trim

indexOf

isBefore

isAfter

isBeforeOrSameAs

isAfterOrSameAs

## On ArrayList

add

insert

remove

removeFirst

removeAll

asImmutableList

## On ImmutableList

get

getRange

with

withInsert

withRemove

withRemoveFirst

withRemoveAll

asArrayList

## On Dictionary

removeKey

keys

values

## On ImmutableDictionary

getKey

hasKey

withKey

withRemoveKey

## On Tuple

first

second

## On Graphics

drawAsGraphics (procedure)

putAt

getAt

putChar

putText

getChar

putForeground

getForeground

putBackground

getBackground

## On instance of a class

typeAndProperties

## On many different types

asString

asIter

## Count TODO difference between length and count ?

length

head

# Console

## See

print

printTab

clearConsole

## Input

# Graphics

About Graphics

See

clearGraphics

See also pause

initialisedGraphics

# Reading & writing data files

Not yet implemented – but will be included in v1.0.0