Pointfrip Language Reference

2022-05-06

Naming Conventions

name	the name itself
name	the type / the class
(?)	Uncertainties
*	Footnote / Note

Data Types

Data type	Syntax	Type identifier
data		// General
null	( )	_null
*int**	[_123]	_integer
real	_31.415e_123	_real
string	"abc"	_string
ident	abc or +-*/	_ident
prefix	@...	_prefix
index	[abc]	_index
array	{a b c}	_array
error	(*index* _error ... ...)	_error
table/dict**	(a x b y c z)	// In pairs
list	(a ; b ; c ;)	;
object	(*ident* :: a x b y c z)	::
turtle	(turtle :: ... ...)	// Object
combi	(*term* _combine .. *arg*)	_combine
monad	(*int* _act ... ...)	_act
bool	true or false	// Idents
etc...		

* note that *int*- and *dict*-literals require the constant combinator!

Comments

codetext // *comment*

Definition of Identifiers

ident **==** *term*
ident **≡** *term*

Script Structure

term definition1 definition2 definition3 ...

Ddot

prop = *head infix* **..** *tail*

Include Files

coreimport == "*Script1.txt*" ; "*Script2.txt*" ; "*Script3.txt*" ; ... ;

userimport == "*Script1.txt*" ; "*Script2.txt*" ; "*Script3.txt*" ; ... ;

corepath?

userpath?

List/Dict Functions and Operators

dict = (first$_1$ infix$_1$ first$_2$ infix$_2$ … … first$_m$ infix$_m$)
list = (element$_0$; element$_1$; element$_2$; … ;)

[*i* **]** ° *list* -- *element*$_i$

head ° *dict* -- first
head ° *list* -- *first*
First element of the list.
head ° *object* --

tail ° *dict* -- rest
tail ° *list* -- *rest*
List without the first element and first infix.
tail ° *object* --

infix ° *dict* -- infix value
infix ° *object* --

prop ° *first*,*infix*,*rest*, -- dict

top ° *dict* -- first
top ° *list* -- first

pop ° *dict* -- rest
pop ° *list* -- rest

tag ° *data* -- typus // als typeof
tag ° *dict* -- infix value

term ° *combi* -- term value

arg ° *combi* -- arg value

termoarg -- term o arg

first **,** *rest* -- list
Appendleft

length ° *dict* -- *real*
length ° *list* -- *real*
length ° object --
Number of list items.

reverse ° *dict* -- *dict*
reverse ° *list* -- *list*
reverse ° *object* --
Reverses the list items.

data **distl** *list* -- matrix

list **distr** *data* -- matrix

dict **++** *dict* -- *dict*
list **++** *list* -- *list*
Concatenate the lists.

dict **take** *num* -- *dict*
list **take** *num* -- *list*
Takes the first *num* elements from the list.

dict **drop** *num* -- *dict*
list **drop** *num* -- *list*
Drops the first *num* elements in the list.

trans ° *matrix* -- matrix
transpose ° *matrix* -- matrix

num **pick** *list* -- element
num **sel** *list* -- element

last ° *list* --

(*num* **r**) ° *list* --

tailr ° *list* -- list
tailr ° *dict* -- dict

rotl ° *list* -- list

rotr ° *list* -- list

list **count** *data* -- real

data **make** *num* -- list

list **find** *data* -- real

iota ° *num* -- *list*
ι ° *num* -- *list*
Generates a list of numbers from 1 to *num*.

iota0 ° *num* -- *list*
Generates a list of numbers from 0 to *num*-1.

int **to** *int* -- list
real **to** *real* -- list

int **upto** *int* -- list
real **upto** *real* -- list

int **downto** *int* -- list
real **downto** *real* -- list

swap ° *x*,*y*,*list* -- *y*,*x*,list

Math Functions and Operators

int **+** *int* -- *int*
real **+** *real* -- *real*
Addition of numbers.

int **-** *int* -- *int*
real **-** *real* -- *real*
Subtraction of numbers.

int ***** *int* -- *int*
real ***** *real* -- *real*
int **×** *int* -- *int*
real **×** *real* -- *real*
Multiplication of numbers.

num **/** *num* -- *real*
num **÷** *num* -- *real*
Division of numbers.

int **^** *int* -- *int*
real **^** *real* -- *real*
Power of numbers.

int **idiv** *int* -- *int*
Integer division

int **imod** *int* -- *int*
Integer modulo

pred ° *int* -- *int*
pred ° *real* -- *real*
Predecessor function

succ ° *int* -- *int*
succ ° *real* -- *real*
Successor function

sign ° *int* -- *int*
sign ° *real* -- *real*
Sign function

abs ° *int* -- *int*
abs ° *real* -- *real*
Absolute value function

neg ° *int* -- *int*
neg ° *real* -- *real*
_ ° *int* -- *int*
_ ° *real* -- *real*
Negation of a number.

round ° *num* -- *int*
round ° *complex* -- *int*
Rounding to an integer.

trunc ° *num* -- *int*
Truncate to an integer.

int ° *num* -- *real*
Integer part of the number as a real number.

frac ° *num* -- *real*
Fraction part of a real number.

float ° *num* -- *real*
float ° *complex* – *real*
Conversion to the real number.

num **roundto** *num* -- real

exp ° *real* -- *real*
Exponential function

ln ° *real* -- *real*
Natural logarithm.

lg ° *real* -- *real*
Decadic logarithm.

ld ° *real* -- *real*
Binary logarithm.

sq ° *int* -- *int*
sq ° *real* -- *real*
Square of a number.

sqrt ° *num* -- *real*
Square root of a number.

pi -- 3.141592653589793
Ludolph's number π

2pi -- 6.283185307179586
Scope of the unit circle.

sin ° *real* -- *real*
Sine function

cos ° *real* -- *real*
Cosine function

tan ° *real* -- *real*
Tangent function

cot ° *real* -- *real*
Cotangent function

sec ° *real* -- real

csc ° *real* -- real

arcsin ° *real* -- *real*
Arcsine function

arccos ° *real* -- *real*
Arccosine function

arctan ° *real* -- *real*
Arctangent function

num **arctan2** *num* -- real

arccot ° *real* -- real

arcsec ° *real* -- real

arccsc ° *real* -- real

sinh ° *real* -- *real*
Hyperbolic sine function

cosh ° *real* -- *real*
Hyperbolic cosine function

tanh ° *real* -- *real*
Hyperbolic tangent function

coth ° *real* -- real

sech ° *real* -- real

csch ° *real* -- real

arsinh ° *real* -- real

arcosh ° *real* -- real

artanh ° *real* -- real

arcoth ° *real* -- real // Library “complex.txt“

arsech ° *real* -- real // Library “complex.txt“

arcsch ° *real* -- real // Library “complex.txt“

real **root** *real* -- real // Library “complex.txt“

deg ° *num* -- *real*
Radiant-to-Degree function

rad ° *num* -- *real*
Degree-to-Radiant function

hex ° *num* – *string*
Number as hexadecimal string.

real **mod** *real* -- *real*
Modulo of real numbers.

sum ° *list* -- *num*
Sum of the list items.

prod ° *list* -- *num*
Product of the list items.

avg ° *list* -- *real*
Average value of the list items.

integral

dd

zero ° *data* --

one ° *data* --

half ° *data* -- … // Library “complex.txt“

Dictionary Operators and Combinators

dict is a table for pattern matching treatment

dict = (*value0 key0 value1 key1 value2 key2 ... ...*)

_super
Key for the super dictionary.

dict **get** *key* -- *value*
Get the *value* for the *key* from a *dict*.

dict **put** *key*,*value*, -- *dict*
Replaces the *value* to a *key* in the *dict*.

dict **iget** *ident* -- *value*
dict **iget** *index* -- *value*
API-Get for identical keys.

dict **iput** *ident*,*value*, -- *dict*
dict **iput** *index*,*value*, -- *dict*
API-Put for identical keys.

#*ident* ° *dict* -- *value*
(*ident* **_v**) ° *dict* -- *value*
Instance variable value.

(*ident* **:=** *value*) ° *dict* -- *dict*
Substitution of an instance variable with a *value*.

func **<-** *key1* ; *key2* ; ... ;
func **←** *key1* ; *key2* ; ... ;
Assign combinator, general.

func **<-** *key1 isfunc1 key2 isfunc2 ... ...*
func **←** *key1 isfunc1 key2 isfunc2 ... ...*
Assign combinator, typed.

keys ° *dict* -- list

values ° *dict* -- list

Boolean Functions and Operators

bool = **true** or **false**

'**true** -- *bool*
Value for true.

'**false** -- *bool*
Value for false.

data **=** *data* -- *bool*
Check for equality.

data **<>** *data* -- *bool*
data **!=** *data* -- *bool*
data **≠** *data* -- *bool*
Check for inequality.

data **<** *data* -- *bool*
Checks whether smaller.

data **>** *data* -- *bool*
Checks whether larger.

data **<=** *data* -- *bool*
Checks whether less than or equal.

data **>=** *data* -- *bool*
Checks whether greater than or equal to.

¬ ° *bool* -- *bool*
not ° *bool* -- *bool*
not ° *int* -- *int*
NOT function

bool **and** *bool* -- *bool*
int **and** *int* -- *int*
AND operator

bool **or** *bool* -- *bool*
int **or** *int* -- *int*
OR operator

bool **xor** *bool* -- *bool*
int **xor** *int* -- *int*
Exclusive-OR operator

isatom ° *data* -- bool
Checks whether the *data* is a basic data type. (?)

isprop ° *data* -- *bool*
Checks whether the *data* is a triple value. (?)

islist ° *data* -- *bool*
Checks whether the *data* is a list.

isbool ° *data* -- *bool*
Checks whether the *data* is a Boolean identifier.

isnum ° *data* -- *bool*
Checks whether the *data* is a number. Generic function.

iszero ° *data* -- bool
Checks whether the data is zero. Generic function.

ispos ° *data* -- *bool*
Checks whether the *data* is greater than zero. Generic function.

isneg ° *data* -- *bool*
Checks whether the *data* is less than zero. Generic function.

isnil (?)

ispreg (?)

isnull ° *data* -- *bool*
isint ° *data* -- *bool*
isreal ° *data* -- *bool*
isstring ° *data* -- *bool*
isident ° *data* -- *bool*
isprefix ° *data* -- *bool*
isindex ° *data* -- *bool*
isarray ° *data* -- *bool*
iscons ° *data* -- *bool*
iscombi ° *data* -- *bool*
isalt ° *data* -- *bool*
isobj ° *data* -- *bool*
isquote ° *data* -- *bool*
isivar ° *data* -- *bool*
isact ° *data* -- *bool*
Predicates to check the appropriate data type.

isbound ° *ident* -- *bool*
isbound ° *prefix* -- *bool*
Checks whether an identifier is bound.

isundef ° *data* -- *bool*
Testing for _undef

iscomplex ° *complex* -- *bool*
Checks whether it is a complex number. (?)

ismatrix ° *object* -- bool

isodd ° *int* -- bool
isodd ° *real* -- bool

object **is** *ident* -- *bool*
Checks whether the *ident* is the same as the class identifier of the *object*. (?)

(*ident* **hastag**) ° *data* -- bool (?name)

Combinators for Program Execution (?)

combi = (*term* **_combine** .. *arg*)

func **_s**
Single function evaluation

' *literal*
literal **k**
literal **_q**
Constant combinator

f **:** *x*
Application // to be used for closed and lift

func1 **°** *func2*
func1 **o** *func2*
func1 ∘ *func2* // unicode: 0x2218
Composition of functions.

functional **app** *argument*
Apply operator

func1 **,** *func2* **,** *func3* **,** ... **,**
Construction of lists.

test **->** *then* **|** *else*
test **→** *then* **|** *else*
test **->** *then* **;** *else*
Condition with Alternal/Cons

test **->*** *func*
test **→*** *func*
while Loop

func **loopif** *test*
do-while Loop

(*func* **do**)°*num*,*num*,*num*,

functional **for** *num*,*num*,*num*,

list **map** *functional*
Map operator

(*func* **aa**) ° *list*
(*func* **α**) ° *list*
Apply-to-all combinator

list **insl** *functional*
Insertl operator

list **insr** *functional*
Insertr operator

(*func* \) ° *list*
Insertr combinator

list **filter** *functional*
Filter operator

(*list*,*arg1*,*arg2*,...,) **map0** *functional*

(*func* **aa0**) ° *list*,*arg1*,*arg2*,...,
Combination of **aa** and **distr**, extended.

func1 **ee** *func2*
ee ° *data*,*data*,
Eval-Eval combinator for infix notation.

func1 **swee** *func2*
swee ° *data*,*data*,
Swap-Eval-Eval combinator

(*func1* **eea** *func2*) ° *argum* -- (x ; y ; argum ;)

(*func* **dip**) ° *list*
(*func* **dip**) ° *object*
Dip combinator (stolen from Joy)

(*test* **try** *then* **|** *else*) ° *argument*
in then/else with (*testresult* ; *argument* ;)

‘*expr* **step** *list*,*akku*,

‘*expr* **times** *rep*,*akku*,

ifnull

ifprop

data1 **??** *data2* -- data

(func Y)
Y-Combinator...

quote ° *data* -- func
Quote functional

func1 **comp** *func2* -- func
Compose functional

(*func* **any**) ° *list* -- bool

(*func* **all**) ° *list* -- bool

Misc Functions and Operators

undef -- *error*
Function is defined as undefined.

id ° *argument* -- *argument*
Identity function.

out ° *argument* -- *argument* // *Side effect
Output for debugging.

data **min** *data* -- *data*
min ° *data*,*data*, -- *data*
Minimum of two values.

data **max** *data* -- *data*
max ° *data*,*data*, -- *data*
Maximum of two values.

name ° *ident* -- *string*
Print name of an identifier.

body ° *ident* -- *value*
The assigned *value* of an identifier.

address ° *data* -- *real*
Address value of the triple cell.

identlist -- *list*
List of all used identifiers. (?)

indexdict -- *dict*
Dict of all index types with integers.

maxcell -- int

pointersize -- int
win32 = [32], win64 = [64]

_reserve
Value for an unbound identifier.

_undef
Value for undefined.

gc ° *argument* -- *argument*
Turns on the garbage collector.

String Functions and Operators

substring ° *string*,*num*,*num*, -- string

string **concat** *string* -- *string*
string **&** *string* -- *string*
Concatenates the strings.

string **indexof** *substr* -- real

list **join** *sepstr* -- string

string **split** *sepstr* -- list

string **repeat** *num* -- string

string **delete** *num*,*num*, -- string

string **insert** *num*,*string*, -- string

length ° *string* -- *real*
Length of the string.

string **mid** *num*,*num*, -- string

string **left** *num* -- string

string **right** *num* -- string

char ° *num* -- string

unicode ° *string* -- real

trim ° *string* -- *string*
Trims the *string* on the left and right side.

triml ° *string* -- *string*
Trims the *string* on the left.

trimr ° *string* -- *string*
Trims the *string* on the right.

upper ° *string* -- *string*
AnsiUpperCase of the string.

lower ° *string* -- *string*
AnsiLowerCase of the string.

capitalize ° *string* -- string

parse ° *string* -- *list*
Precompiles the *string* into a *list*.

value ° *string* -- *data*
Converts the *string* to a *data* type.

string ° *data* -- *string*
Converts the *data* to its text representation.

unpack ° *string* -- *list*
Splits the *string* into a list of individual string characters.

pack ° *list* -- *string*
Concatenates the strings in the *list*.

timetostring ° *real* -- string

datetostring ° *num* -- string

weekday ° *num* -- num (?)

OOP

object = (*cap* **::** *inst*) // Object classes

pair = *object* , *parameter* ,

self ° *pair*

para ° *pair*

index **op** *func*

index **swop** *func*

index **fn** *func*

(*object* (*index* **cb** *func*) *parameter*) ° argum -- *method* ° [0],[1],argum,

cap ° *list* -- ( )
cap ° *object* -- (*cap* **::**)

ident **obj** *list* -- (*ident* **::** *list*)
ident **obj** *dict* -- (*ident* **::** *dict*)

ident **new** *parameter*

object **as** ident (?) -- object

box ° *primdata* -- object

unbox ° *object* -- primdata

(*func* **objdip**) ° *pair* -- object

object == .. { ( ) ... ... }
Object class

list == .. { object ... ... }
List class

dict == .. { object ... ... }
Dict class

Monads and Effects

monad = (*int* **_act ..** *dict*) // absolute
monad = (*index* **_act ..** *dict*) // relative

it ° *dict* -- #_it ° *dict*
Result of a monad action. // monad ... name (?)

#_it

#_self

#_para

_bind
Continuation

_eff
Effects

monad **>>** *term* -- monad // _bind := *term*

int **act** *dict* -- monad
index **act** *dict* -- monad
monad **act** *dict* -- monad
nun auch die Möglichkeit gegeben eine actbox zu bauen //bitte in engl!
(verschachteltes Act)

monad **eff** *array* -- monad
monad **eff** *ident* -- monad

monad **var** *data* -- monad
monad **var** *dict* -- monad

(*ident* **define** *data*) ° *dict* -- monad
//(prefix define data) ° dict

(*ident* **redefine** *data*) ° *dict* -- monad
//(prefix redefine data) ° dict

(*data* **showgraph**) ° *dict* -- monad // *+ (*x* eff 'io)

(*data* **showinfo**) ° *dict* -- monad // *+ (*x* eff 'io)

(*data* **print**) ° *dict* -- monad // *+ (*x* eff 'io)

(*string* **input**) ° *dict* -- monad // *+ (*x* eff 'io)
(*string* **input** *string*) ° *dict*

(*fname* **loadtext**) ° *dict* -- monad // *+ (*x* eff 'io)

(*fname* **savetext** *string*) ° *dict* -- monad // *+ (*x* eff 'io)

(*string* **run**) ° *dict* -- monad // *+ (*x* eff 'io)

quit -- monad

time ° *dict* -- monad // *+ (*x* eff 'io)

date ° *dict* -- monad // *+ (*x* eff 'io)

beep ° *dict* -- monad // *+ (*x* eff 'io)

io == .. { ... ... }
System effects class

Runtime Errors(?)

error = (*index* **_error** *string* ; ... ...)

index **error** *string*, -- error

fail ° *argument* -- error
Use for selector signatures(?)

stop ° *argument* -- error
Generally, e.g. Program termination, etc

raise ° *string* -- exception
An exception is thrown.

_error == .. { ... ... }
Class for redirects...

// try

Complex Numbers

complex = (**complex ::** *real* **re** *real* **im**) // Library “complex.txt“

i -- (complex :: 0 re 1 im)
Square root of _1

real **j** *real* -- complex // für schnelle Schreibweise

real **cval** *real* -- *complex*
To form a complex number from real numbers.

re ° *complex* -- *real*
Real part of the complex number.

im ° *complex* -- *real*
Imaginary part of the complex number.

complex **+** *complex* -- *complex*
Addition of complex numbers.

complex **-** *complex* -- *complex*
Subtraction of complex numbers.

complex * *complex* -- *complex*
complex **×** *complex* -- *complex*
Multiplication of complex numbers.

complex **/** *complex* -- *complex*
complex **÷** *complex* -- *complex*
Division of complex numbers.

zero ° *complex* -- (complex :: 0 re 0 im)

one ° *complex* -- (complex :: 1 re 0 im)

half ° *complex* -- (complex :: 0.5 re 0 im)

iszero ° *complex* -- bool

isnum ° *complex* -- true

complex **=** *complex* -- bool

conj ° *complex* -- complex

neg ° *complex* -- complex

abs ° *complex* -- real

phase ° *complex* -- real // wie Arg(z)

sq ° *complex*	--	complex	
exp ° *complex*	--	complex	
ln ° *complex*	--	complex	// Hauptzweig
lg ° *complex*	--	complex	// Log10(z)
ld ° *complex*	--	complex	// Log2(z)
complex **^** *complex*	--	complex	
complex **root** *complex*	--	complex	
sqrt ° *complex*	--	complex	
sin ° *complex*	--	complex	
cos ° *complex*	--	complex	
tan ° *complex*	--	complex	
cot ° *complex*	--	complex	
sec ° *complex*	--	complex	
csc ° *complex*	--	complex	
arcsin ° *complex*	--	complex	
arccos ° *complex*	--	complex	
arctan ° *complex*	--	complex	
arccot ° *complex*	--	complex	
arcsec ° *complex*	--	complex	
arccsc ° *complex*	--	complex	
sinh ° *complex*	--	complex	
cosh ° *complex*	--	complex	
tanh ° *complex*	--	complex	
coth ° *complex*	--	complex	
sech ° *complex*	--	complex	
csch ° *complex*	--	complex	

arsinh ° *complex* -- complex

arcosh ° *complex* -- complex

artanh ° *complex* -- complex

arcoth ° *complex* -- complex

arsech ° *complex* -- complex

arcsch ° *complex* -- complex

iscomplex ° *object* -- bool

complex == .. { dict ... ... }
Complex-class with the complex methods.

Matrix Functions and Operators

matrix = (*list* ; *list* ; ... ;) oder
matrix = (**matrix** :: *list* ; *list* ; ... ;) // Library "matrix.txt"

IP ° *list*,*list*, // Backus Turing Lecture
list **IP** *list*

MM ° *matrix*,*matrix*, // Backus Turing Lecture
matrix **MM** *matrix*

det ° *matrix* -- real

inv ° *matrix* -- matrix

matrix **+** *matrix* -- matrix

matrix **–** *matrix* -- matrix

matrix * *matrix* -- matrix
matrix **×** *matrix* -- matrix

trans ° *matrix* -- matrix // transpose

num **scale** *matrix* -- matrix // (?) scalar

matrix **each** '*func* -- matrix // (?)

sq ° *matrix* -- matrix

zero ° *matrix* -- matrix // zeromatrix

one ° *matrix* -- matrix // idmatrix

num **zeromatrix** *num* -- matrix

num **onematrix** *num* -- matrix

idmatrix ° *num* -- matrix

ismatrix ° *data* -- bool

tomatrix ° *list* -- matrix

num **like** *data* -- *num* // with type of *data*

matrix == .. { list ... ... }
Matrix-class for MM (*), det, inv, trans, add, sub, Aij, negifodd

Turtle Graphics

turtle = (**turtle ::** *list* **stack** *real* **x** *real* **y** *real* **angle**
bool **pen** *num* **color** *num* **size** *num* **brush**) // Library “turtlegraphics.txt“

pair = (x , y ,)

// 2pi

initturtle
'*turtle* **new** // recommended

pair **moveto** *turtle*

pair **moverel** *turtle*

real **move** *turtle*

real **turnto** *turtle*

real **turn** *turtle*

penup ° *turtle*

pendown ° *turtle*

num **pencolor** *turtle*

num **pensize** *turtle*

num **brushcolor** *turtle*

real **circle** *turtle*

rectangle ° *turtle* // rect

(*turtle* (**draw** eff 'io)) ° *dict* -- monad
For drawing the turtle trail.

#x ° *turtle* -- real
#y ° *turtle* -- real
#angle ° *turtle* -- real
etc
Attributes of the turtle object.

colors == '(... ...)
#red ° colors for the color value red.

turtle == .. { dict ... ... }
Turtle class,
own turtle classes can also be created through inheritance.

(xlist (**plot0** eff 'io) 0-y) ° *dict* -- monad