

## Pointfrip Quickinfo

2023-09-08

The following is about programming at the **function-level** with combinators

--- bitte Calibri und mich dann löschen ---

### Rule

As a rule, **right-before-left** applies, but there are exceptions, e.g. in condition terms.

**Parentheses** must be used to change the evaluation of the terms.

**Infix notation** applies as in:  $a + b$

For functions you write:  $function^{\circ} argument$

### Data Types

$[0], [1], [2], \dots, [i],$   
 $[_{123}]$  are selectors that access the values of a list  
or a dict -- or are integer\* numbers

name is an identifier for an associated function

$_{123.5678e_{30}}$  is a real number

$(10 ; 20 ; 30 ; 40 ; 50 ;)$  is a list of real numbers

$(10 a 20 b 30 c 40 d 50 e)$  is a dict\* with values and keys

$()$  empty list / *null*

$(head\ infix\ ..\ tail)$  data cell / *prop*

"abcdef" is a string

true / false are of type *bool*

\*note that the constant combinator should be used.

## Definition of Functions/Constants/Operators

<i>identifier == term</i>	assigns a term to the identifier
<i>constname == ' literal</i>	Constants use the constant combinator
<i>oprname == ( ... ) ° ee</i>	Operators often use an ee and [0] and [1]

## Combinators

<i>'name</i>	is the Constant combinator
<i>function1 ° function2</i>	is the Composition, <b>o</b> can also be used (right-pipe)
<i>fun1 , fun2 , ... , funm ,</i>	is the Construction of a list
<i>(test -&gt; then ; else)</i>	is the Condition combinator with an alternative
<i>(test -&gt; * term)</i>	is a While loop
<i>(function aa)</i>	is the Apply-to-All combinator (map)
<i>(function \ )</i>	is the Inserttr combinator (reduce)
<i>function1 ee function2</i>	evaluates the functions and creates a pair from them
<i>#name</i>	picks the value for the name from a dict
<i>function : argument</i>	is an Application -- <i>function(argument)</i>
<i>list insl 'func</i>	is the Insertl operator; <b>insr</b> for Inserttr operation
<i>func _s</i>	Single function is executed
<i>func1 app func2</i>	Apply function to execute Functionals
<i>func1 swee func2</i>	like <b>ee</b> , only the elements in the pair are swapped
<i>(func aa0) ° list,x,y, ... ,</i>	Mixture of <b>aa</b> and <b>distr</b> , expanded
<i>(list,x,y, ... ,) map0 'func</i>	Mixture of <b>map</b> and <b>distr</b> , extended
<i>list filter 'boolfunc</i>	is the Filter operator

## List Processing Functions and Operators

<code>val0 ; val1 ; val2 ; ... ;</code>	building lists with literal values
<code>head ° prop</code>	extracts the first value of a list
<code>tail ° prop</code>	extracts the rest of a list
<code>infix ° prop</code>	extracts the infix value of a list/dict
<code>prop ° hd, inf, tl,</code>	creates a data cell with three values
<code>term ° combi</code>	extracts the Term value from a Combine data type
<code>arg ° combi</code>	extracts the Arg value from a Combine data type
<code>type ° data</code>	supplies a name for the data type
<code>list at num</code>	picks the <i>num</i> -th value from the <i>list</i>
<code>func , list</code>	the comma adds an element before the <i>list</i>
<code>iota ° num</code>	creates a list of numbers from 1 upwards to <i>num</i>
<code>num1 to num2</code>	produces a list of numbers from <i>num1</i> to <i>num2</i>
<code>reverse ° list</code>	reverses a list; also works with a string
<code>trans ° matrix</code>	Transpose a list of lists (matrix)
<code>data distl list</code>	Distribution Left
<code>list distr data</code>	Distribution Right
<code>data make num</code>	creates a list of <i>num</i> <i>data</i> -values
<code>list take num</code>	returns a list of the first <i>num</i> elements
<code>list drop num</code>	returns the remainder list without the first <i>num</i> elements
<code>list1 ++ list2</code>	concatenates two lists into a new list
<code>length ° list</code>	returns the length of a <i>list</i>
<code>list count data</code>	returns the number of <i>data</i> in the <i>list</i>
<code>list find data</code>	returns the first position of <i>data</i> in the <i>list</i>

## Numerische Funktionen und Operatoren

$num1 + num2$	Addition von Zahlen gleichen Typs
$num1 - num2$	Subtraktion von Zahlen gleichen Typs
$num1 * num2$	Multiplikation von Zahlen gleichen Typs
$num1 / num2$	Division von Zahlen gleichen Typs
$num1 \wedge num2$	Potenzierung von Zahlen gleichen Typs
$num1 \text{ idiv } num2$	Division von Integerzahlen
$num1 \text{ imod } num2$	Modulo von Integerzahlen
$\text{pred} \circ num$	Vorgängerfunktion
$\text{succ} \circ num$	Nachfolgerfunktion
$\text{sign} \circ num$	Signumfunktion
$\text{abs} \circ num$	Betrag der Zahl
$\text{neg} \circ num$	Negation der Zahl
$\_ \circ num$	Negation der Zahl
$\text{floor} \circ num$	Abrunden der Zahl
$\text{ceil} \circ num$	Aufrunden der Zahl
$\text{float} \circ num$	wandelt in eine Realzahl um
$\text{round} \circ num$	rundet in eine Integerzahl um
$\text{trunc} \circ num$	Integerzahl ohne Beachtung der Nachkommawerte
$\text{real roundto } num$	Rundet auf die $num$ -te Nachkommastelle

<b>exp ° num</b>	Exponentialfunktion der Zahl
<b>ln ° num</b>	natürlicher Logarithmus der Zahl
<b>lg ° num</b>	Zehnerlogarithmus der Zahl
<b>sq ° num</b>	das Quadrat der Zahl
<b>sqrt ° num</b>	die Quadratwurzel der Zahl
<b>cbrt ° num</b>	die Qubikwurzel der Zahl
<b>pi</b>	Funktion liefert die Zahl Pi
<b>2pi</b>	Funktion liefert den Umfang des Einheitskreises
<b>sin ° num</b>	Sinusfunktion der Zahl in Radiant
<b>cos ° num</b>	Cosinusfunktion der Zahl in Radiant
<b>tan ° num</b>	Tangensfunktion der Zahl in Radiant
<b>arcsin ° num</b>	Arcussinusfunktion
<b>arccos ° num</b>	Arcuscosinusfunktion
<b>arctan ° num</b>	Arcustangensfunktion
<b>y arctan2 x</b>	Phase (oder Arg) zu (x,y)
<b>sinh ° num</b>	Sinus Hyperbolicus Funktion
<b>cosh ° num</b>	Cosinus Hyperbolicus Funktion
<b>tanh ° num</b>	Tangens Hyperbolicus Funktion
<b>deg ° num</b>	wandelt Radiant in Degree um
<b>rad ° num</b>	wandelt Degree in Radiant um

## Boolische Funktionen und Operatoren

<i>data1</i> = <i>data2</i>	prüft auf Gleichheit
<i>data1</i> != <i>data2</i>	prüft auf Ungleichheit
<i>data1</i> <> <i>data2</i>	prüft auf Ungleichheit, alternativ
<i>data1</i> < <i>data2</i>	Vergleich auf Kleiner-als
<i>data1</i> > <i>data2</i>	Vergleich auf Größer-als
<i>data1</i> <= <i>data2</i>	Vergleich auf Kleiner-Gleich
<i>data1</i> >= <i>data2</i>	Vergleich auf Größer-Gleich
<i>data1</i> min <i>data2</i>	Minimum von <i>data1</i> und <i>data2</i>
<i>data1</i> max <i>data2</i>	Maximum von <i>data1</i> und <i>data2</i>
not ° <i>bool</i>	Boolische Nicht-Funktion
<i>bool1</i> and <i>bool2</i>	Boolische Und-Funktion
<i>bool1</i> or <i>bool2</i>	Boolische Oder-Funktion
<i>bool1</i> xor <i>bool2</i>	Boolische Exklusiv-Oder-Funktion
isatom ° <i>data</i>	Prüft, ob <i>data</i> zu den Atom-Typen gehört
isnull ° <i>data</i>	Prüft, ob <i>data</i> der Wert ( ) also Null ist
isprop ° <i>data</i>	Prüft, ob <i>data</i> eine Datenzelle ist
islist ° <i>data</i>	Prüft, ob <i>data</i> eine Liste ist
isnum ° <i>data</i>	Prüft, ob <i>data</i> eine Zahl ist, generisch
iszero ° <i>data</i>	Prüft, ob <i>data</i> die Zahl 0 ist, generisch
ispos ° <i>num</i>	Prüft, ob <i>num</i> eine positive Zahl ist, generisch
isneg ° <i>num</i>	Prüft, ob <i>num</i> eine negative Zahl ist, generisch
isident ° <i>data</i>	Prüft, ob <i>data</i> ein Bezeichner ist

<b>isint</b> ° <i>data</i>	Prüft, ob <i>data</i> eine Integerzahl ist
<b>isreal</b> ° <i>data</i>	Prüft, ob <i>data</i> eine Realzahl ist
<b>isstring</b> ° <i>data</i>	Prüft, ob <i>data</i> ein Zeichenstring ist
<b>iscons</b> ° <i>data</i>	Prüft, ob <i>data</i> eine List-Datenzelle ist
<b>isquote</b> ° <i>data</i>	Prüft, ob <i>data</i> ein Quotewert ist
<b>isivar</b> ° <i>data</i>	Prüft, ob <i>data</i> ein Instanz-Variablen-Selektor ist
<b>iscombi</b> ° <i>data</i>	Prüft, ob <i>data</i> ein Combine-Wert ist
<b>isact</b> ° <i>data</i>	Prüft, ob <i>data</i> ein Act-Wert ist
<b>isbool</b> ° <i>data</i>	Prüft, ob <i>data</i> ein boolischer Wert ist
<b>isbound</b> ° <i>ident</i>	Prüft, ob der Bezeichner schon definiert ist
<b>isundef</b> ° <i>data</i>	Prüft, ob <i>data</i> der Wert <b>_undef</b> ist
<i>data</i> <b>in</b> <i>liste</i>	Prüft, ob <i>data</i> als Element in der <i>liste</i> enthalten ist

## Dict Funktionen und Operatoren

<b>#ident</b> ° <i>dict</i>	der Selektor pickt zum <i>ident</i> -Key den Wert aus dem <i>dict</i>
<i>dict</i> <b>iget</b> <i>ident</i>	zum <i>ident</i> *-Key wird der Wert aus dem <i>dict</i> herausgepickt
<i>dict</i> <b>iput</b> <i>ident</i> , <i>value</i> ,	zum <i>ident</i> *-Key wird der <i>value</i> neu im <i>dict</i> angelegt
<i>dict</i> <b>get</b> <i>key</i>	zum <i>key</i> wird der Wert aus dem <i>dict</i> herausgepickt
<i>dict</i> <b>put</b> <i>key</i> , <i>value</i> ,	zum <i>key</i> wird der <i>value</i> neu im <i>dict</i> angelegt
( <i>ident</i> <b>:=</b> <i>func</i> ) ° <i>dict</i>	wie bei <b>iput</b> geschieht diese "Variablen"-Zuweisung
( <i>func</i> <b>&lt;-</b> <i>x</i> ; <i>y</i> ; ... ;) ° <i>liste</i>	<i>func</i> wendet das erzeugte Dict, wie nach einem Assign, an
<b>keys</b> ° <i>dict</i>	erzeugt eine Liste mit allen Keys aus dem <i>dict</i>
<b>values</b> ° <i>dict</i>	erzeugt eine Liste mit allen Values aus dem <i>dict</i>
<b>it</b> ° <i>dict</i>	Pickt aus dem <i>dict</i> den zu <b>_it</b> zugehörigen Wert

## String Funktionen und Operatoren

<b>length</b> ° <i>string</i>	gibt die Länge des Strings an
<b>substring</b> ° <i>string, i, len,</i>	Kopiert einen Teilstring aus <i>string</i>
<i>string1</i> & <i>string2</i>	verkettet zwei Strings
<i>string1</i> <b>concat</b> <i>string2</i>	verkettet zwei Strings
<i>string</i> <b>indexof</b> <i>substr</i>	sucht die Position von <i>substr</i> im <i>string</i> von links
<b>trim</b> ° <i>string</i>	schneidet links und rechts die Leerzeichen ab
<b>triml</b> ° <i>string</i>	schneidet links die Leerzeichen ab
<b>trimr</b> ° <i>string</i>	schneidet rechts die Leerzeichen ab
<b>upper</b> ° <i>string</i>	wandelt den String in Großbuchstaben um
<b>lower</b> ° <i>string</i>	wandelt den String in Kleinbuchstaben um
<b>capitalize</b> ° <i>string</i>	wandelt den String in ein Hauptwort um
<b>char</b> ° <i>num</i>	erzeugt ein Zeichen nach dem Unicodewert
<b>unicode</b> ° <i>string</i>	bestimmt den Unicodewert des ersten Zeichens
<b>parse</b> ° <i>string</i>	parst den String mit dem Pointfrip-Parser
<b>value</b> ° <i>string</i>	wandelt Zahlen, Wörter,Listen im String in Daten um
<b>string</b> ° <i>data</i>	wandelt die Daten in einen Printstring um
<b>unpack</b> ° <i>string</i>	zerlegt den String in eine Liste von Einzelzeichen
<i>string</i> <b>split</b> <i>delstr</i>	zerlegt den <i>string</i> in eine Liste mit Strings ohne <i>delstr</i>
<i>liste</i> <b>join</b> <i>insstr</i>	verbindet die Strings der Liste mit <i>insstr</i> dazwischen



## Matrix Funktionen und Operatoren

<i>matrix1</i> <b>add</b> <i>matrix2</i>	Addiert zwei Matrizen, komponentenweise
<i>matrix1</i> <b>sub</b> <i>matrix2</i>	Subtrahiert <i>matrix2</i> von <i>matrix1</i>
<i>matrix1</i> <b>mul</b> <i>matrix2</i>	Multipliziert zwei Matrizen
<i>num</i> <b>mul</b> <i>matrix</i> <i>matrix</i> <b>mul</b> <i>num</i>	Multipliziert die <i>matrix</i> mit einem Skalarwert
<b>ismat</b> ° <i>data</i>	Prüft, ob <i>data</i> eine Matrix ist, vereinfachte Form
<b>trans</b> ° <i>matrix</i>	Transpose der <i>matrix</i>
<b>det</b> ° <i>matrix</i>	Berechnet die Determinante der <i>matrix</i>
<b>inv</b> ° <i>matrix</i>	Berechnet die Inverse Matrix
<i>num1</i> <b>zeromat</b> <i>num2</i>	erzeugt eine Matrix mit lauter Nullen
<b>idmat</b> ° <i>num</i>	Identitätsmatrix der Größe <i>num</i>
<b>fail</b> ° <i>infodata</i>	erzeugt Standardfehlermeldung für einen Fail
<i>liste</i> <b>IP</b> <i>liste</i>	Inner Product nach John Backus
<i>matrix</i> <b>MM</b> <i>matrix</i>	Matrixmultiplikation nach John Backus
<b>rnd</b> ° <i>matrix</i>	Rundet <i>matrix</i> auf fünf Nachkommastellen
<b>zero</b> ° <i>data</i> <b>zero</b> ° <i>matrix</i>	generiert eine Null, typenabhängig
<b>one</b> ° <i>data</i> <b>one</b> ° <i>matrix</i>	generiert eine Eins, typenabhängig

## Misc Funktionen und Operatoren

<b>undef</b>	erzeugt Fehlermeldung für undefinierte Funktion
<b>id</b> ° <i>data</i>	Identitätsfunktion liefert <i>data</i>
<b>name</b> ° <i>ident</i>	extrahiert den String des Bezeichners
<b>body</b> ° <i>ident</i>	extrahiert den Definitionswert des Bezeichners
<b>info</b> ° <i>ident</i>	extrahiert den Compilerstring des Bezeichners
<b>identlist</b>	gibt eine Liste mit allen verwendeten Bezeichnern aus
<b>quote</b> ° <i>data</i>	macht aus <i>data</i> einen Quote-Wert
<i>ident</i> <b>error</b> <i>string</i>	gibt eine Fehlermeldung mit <i>ident</i> und <i>string</i> aus
<i>'func1</i> <b>comp</b> <i>'func2</i>	verkettet die Funktionen zu einer neuen Funktion
<i>int</i> <b>act</b> <i>dict</i>	erzeugt einen Act-Wert mit den Daten - (Monade)
<i>act</i> <b>bind</b> <i>'func</i>	legt die <i>func</i> im bind-Feld von einem neuen <i>act</i> an
<i>act</i> <b>&gt;&gt;</b> <i>func</i>	legt die <i>func</i> im bind-Feld von <i>einem neuen act</i> an
<i>fname</i> <b>load</b>	liest den Text von der Datei <i>fname</i> ins Display
<i>fname</i> <b>save</b>	sichert den Text vom Display in die Datei <i>fname</i>
<b>files</b>	gibt eine Liste mit allen Dateinamen aus
<i>fname</i> <b>loadtext</b>	lädt den String aus der Datei <i>fname</i> im Ordner "pf/"
<i>fname</i> <b>savetext</b> <i>string</i>	sichert den <i>string</i> in der Datei <i>fname</i> im Ordner "pf/"
<b>stopvm</b>	bricht die Berechnung ab, mit Fehlermeldung
<b>dump</b>	zeigt alle Bezeichner mit ihren Zuordnungen an
<b>savedump</b> (for test)	zeigt alle info-Strings der Bezeichner an
<b>help</b>	Linkadresse zu aktuellem Hilfe-PDF
<b>pim</b> ° <i>num</i>	gibt eine Liste aller Primfaktoren einer Zahl an, Beispiel
( <i>test</i> <b>try then;else</b> )° <i>argum</i>	Prüft <i>test</i> auf Error -> then/else mit ( <i>result</i> ; <i>argum</i> ;)

## Hinweise zum Laden und Speichern von Programmdateien

<code>"filename" save</code>	ein Programmtext wird unter dem Namen <i>filename</i> im Ordner „pf/“ gesichert
<code>"filename" load</code>	ein Programmtext der Datei <i>filename</i> aus dem Ordner „pf/“ wird mit den Definitionen eingelesen
<code>files</code>	gibt Liste mit allen Dateinamen im Ordner „pf/“ aus

Mit **identlist** oder **dump** bekommt man einen Überblick über die verwendeten Wörter.

\*man beachte, daß der Konstanten-Kombinator verwendet werden sollte.

(CC0)