FP trivia Quickinfo

2021-03-29

Im Folgenden geht es ums Programmieren auf Funktionsniveau mit Kombinatoren

Regel

In der Regel gilt **Rechts-vor-Links**, es gibt aber Ausnahmen z.B. bei der Kondition.
Für eine geänderte Auswertung der Terme müssen **Klammern** gesetzt werden.
Es gilt **Infixnotation** wie bei: *a* **+** *b*
Bei Funktionen schreibt man: *funktion* **°** *argument*

Datentypen

[0] , [1] , [2] , ... , **[** ***i*** **]** , [_123]	sind Selektoren, die auf die Werte einer Liste, einem Dict oder einem Array zugreifen; oder sind Integer*
name	ist ein Bezeichner für eine ihm zugeordnete Funktion
_123.5678e_30	ist eine Real-Zahl
(10 ; 20 ; 30 ; 40 ; 50 ;)	ist eine Liste von Real-Zahlen
(10 a 20 b 30 c 40 d 50 e)	ist ein Dict* mit Werten und Schlüsseln
( )	leere Liste

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

Definiton von Funktionen/Konstanten/Operatoren

bez **==** *term*	weist dem Bezeichner einem Term zu
cnst **==** **'** *literal*	Konstanten verwenden den Konstanten-Kombinator
opr **==** (...) **°** **ee**	Operatoren verwenden häufig ein ee und [0] und [1]

Kombinatoren

'*name*	ist der Konstanten-Kombinator
funktion1 **°** *funktion2*	ist die Komposition
fun1 **,** *fun2* **,** ... **,** *funm* **,**	ist die Konstruktion einer Liste
(*test* **->** *dann* **|** *sonst*)	ist der Kondition-Kombinator mit einem Alternal
(*test* **->*** *term*)	ist eine While-Schleife
(*funktion* **aa**)	ist der Apply-To-All-Kombinator (map)
(*funktion* ****)	ist der Insert-Kombinator (reduce)
funktion1 **ee** *funktion2*	wertet die Funktionen aus und erzeugt daraus ein Paar
#*name*	pickt den Wert zum Namen aus einem Dict

Funktionen

id	Identitätsfunktion
iota	erzeugt eine Liste von Zahlen ab 1 aufwärts bis zur Zahl
head	extrahiert den ersten Wert einer Liste
tail	extrahiert den Rest einer Liste
infix	extrahiert den Infixwert einer Liste/Dicts
prop	erzeugt eine Zelle aus Erstem,Infix,Rest,
top	wie head, aber nicht für Objekte
pop	wie tail, aber nicht für Objekte
tag	extrahiert den Typus oder Infixwert
reverse	kehrt eine Liste um

length	liefert die Länge einer Liste
sin	berechnet den Sinus einer Zahl
ln	berechnet den natürlichen Logarithmus einer Zahl
islist	prüft, ob es sich um eine Liste handelt

Operatoren

erster **,** *rest*	Komma erzeugt eine Liste
num1 **+** *num2*	Arithmetische Operatoren für Addition,
num1 **-** *num2*	für Subtraktion
num1 * *num2*	für Multiplikation
num1 / *num2*	für Division
dict **get** *bez*	ermittelt den Wert zum Schlüssel bez*
dict **put** *bez*,*wert*,	erzetzt/legt einen Schlüssel* mit Wert im Dict an
(*bez* **:=** *wert*) ° *dict*	wie put, aber als Wert-Zuweisung im Dict
num1 **=** *num2*	prüft auf Gleichheit und liefert dann **true** oder **false**
etc	

Objekte und Klassen

(turtle **::** ( ) stack 0 x 0 y ...)	ist das Turtleobjekt mit den Attributen stack, x, y, ...
turtle == **..** { dict ... ... }	ist die Turtleklasse mit den ... Methoden

Monaden und Effekte

('turtle new) (**draw** eff 'io)	erzeugt eine Monade zum Zeichnen der Turtlespur

io == .. { ... ... } sind die System-Effekte, sozusagen der Treiber (?)

et cetera zu finden in der Reference/blaues Fragezeichen

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