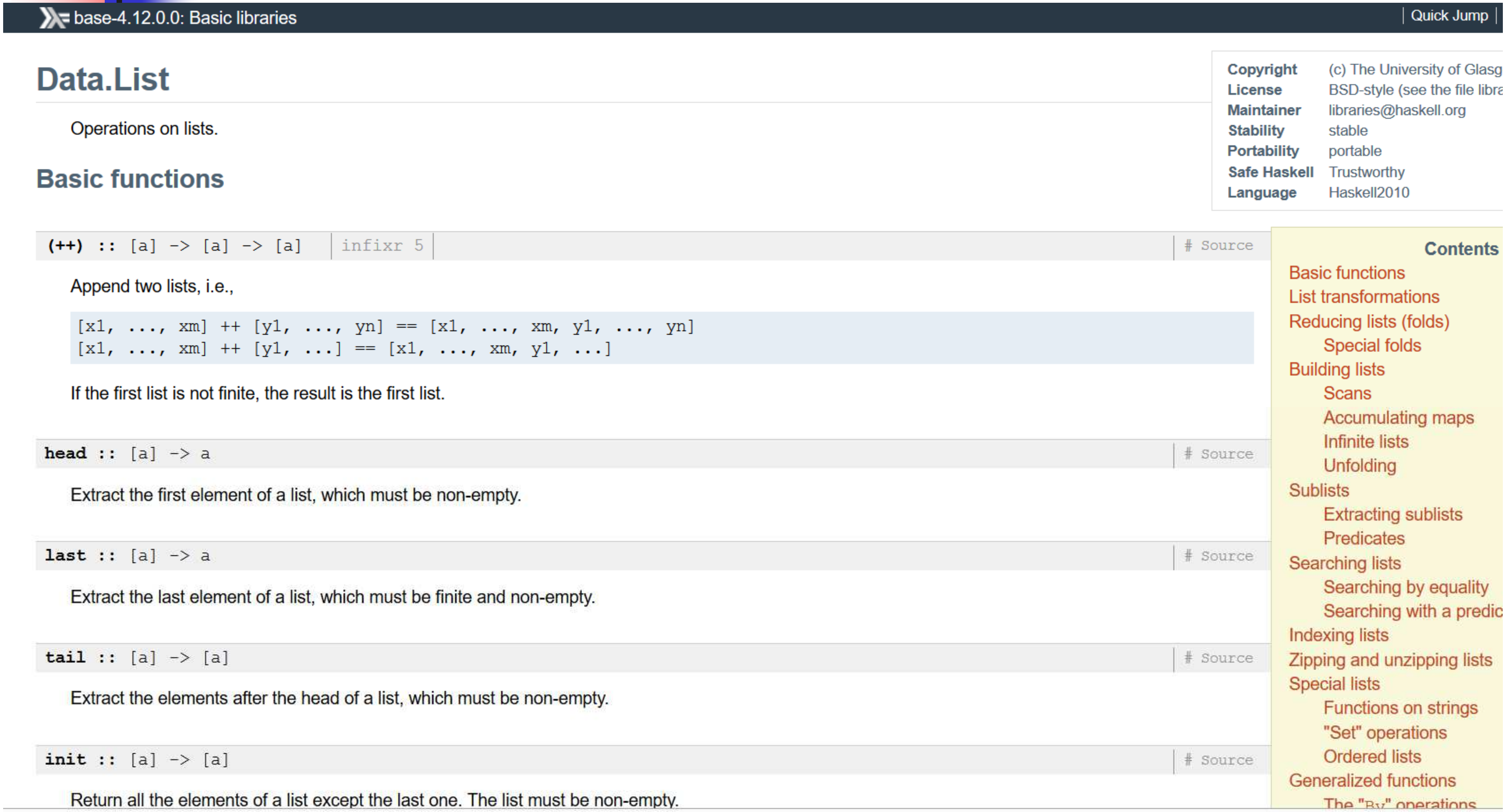
Zoznamová zoznamka

Haskell:	Python:	
xs = [1,2,3,4,5] [1..5]	[1,2,3,4,5]	
length xs	len xs	
xs!!i	xs[i]	.. indexy 0..length xs-1
neexistuje-immutable list	xs[i]=…	
head xs	xs[0]	1
tail xs	xs[1:]	[2,3,4,5]
last xs	xs[len(xs)-1]	5
init xs	xs[:len(xs)-1]	[1,2,3,4]
take n xs	xs[:n]	
drop n xs	xs[n:]	
take m (drop n xs)	xs[n:n+m]	
xs++xs	xs+xs	[1,2,3,4,5,1,2,3,4,5]
reverse xs	xs.reverse()	returns void

import Data.List

http://hackage.haskell.org/package/base-4.12.0.0/docs/Data-List.html



base-4.12.0.0: Basic libraries

Quick Jump

Data.List

Copyright	(c) The University of Glasg
License	BSD-style (see the file libra
Maintainer	libraries@haskell.org
Stability	stable
Portability	portable
Safe Haskell	Trustworthy
Language	Haskell2010

Operations on lists.

Basic functions

```
(++) :: [a] -> [a] -> [a]    infixr 5                # Source
```

Append two lists, i.e.,

```
[x1, ..., xm] ++ [y1, ..., yn] == [x1, ..., xm, y1, ..., yn]
[x1, ..., xm] ++ [y1, ...] == [x1, ..., xm, y1, ...]
```

If the first list is not finite, the result is the first list.

```
head :: [a] -> a                                      # Source
```

Extract the first element of a list, which must be non-empty.

```
last :: [a] -> a                                      # Source
```

Extract the last element of a list, which must be finite and non-empty.

```
tail :: [a] -> [a]                                    # Source
```

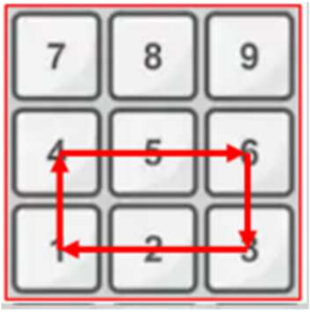
Extract the elements after the head of a list, which must be non-empty.

```
init :: [a] -> [a]                                    # Source
```

Return all the elements of a list except the last one. The list must be non-empty.

Contents

Obĺžnikové čísla



Príklady obĺžnikových čísel (prvé 4 sú z obrázku): 4631, 6314, 3146, 1463, 1287,4521,2563,7931,8998,7777, ...

```
*Magic11> obdlznikoveCisla
[7777,7887,7997,7744,7854,7964,7711,7821,7931,8778,8888,8998,8745,
8855,8965,8712,8822,8932,9779,9889,9999,9746,9856,9966,9713,9823,9
933,4477,4587,4697,4444,4554,4664,4411,4521,4631,5478,5588,5698,54
45,5555,5665,5412,5522,5632,6479,6589,6699,6446,6556,6666,6413,652
3,6633,1177,1287,1397,1144,1254,1364,1111,1221,1331,2178,2288,2398
,2145,2255,2365,2112,2222,2332,3179,3289,3399,3146,3256,3366,3113,
3223,3333]
*Magic11> length obdlznikoveCisla
81
```

```
module Magic11 where

keys = [[7,8,9],
         [4..6],
         [1..3]
        ]
kontrapriklad :: Int
kontrapriklad = if null $ filter (\x -> x `mod` 11 > 0)
                       [
                         1000*keys!!r1!!s1+100*keys!!r1!!s2+
                         10*keys!!r2!!s2+keys!!r2!!s1
                        | r1<-[0..2], s1<-[0..2], r2<-[0..2], s2<-[0..2]
                       ]
                then 0
                else 99999 -- dorobte doma :)
```

Kritérium delietel’nosti 11

- rodné číslo 786115 3333 (ženské, *15.nov1978)
- 7861153333 `mod` 11 == 0
- 11 | 7861153333 iff 11 | 7+6+1+3+3 –(8+1+5+3+3) = 0

- naše rodné čísla sú delitelné 11, ľahká kontrola

- čísla kariet majú tiež kontrolu, Luhnnov algo, DÚ1

- čo bankové účty
- 7000155733 / 8180 – soc.poisťovňa
- cifry násobíme váhami 6,3,7,9,10,5,8,4,2,1, sčítame, výsledok delietený 11
- 11 | 7*6+0*3+0*7+0*9+1*10+5*5+5*8+7*4+3*2+3*1
- (sum $ zipWith (*) [7,0,0,0,1,5,5,7,3,3] [6,3,7,9,10,5,8,4,2,1]) `mod` 11
- (sum $ zipWith (*) [2,7,0,1,1,3,2,4,4,3] [6,3,7,9,10,5,8,4,2,1]) `mod` 11

Všetko,čo by ste chceli vediet o Haskelli, ale báli ste sa spýtať

- že **a b c d = (((a b) c) d)**…lebo operátor aplikácie funkcie na argument je *ľavo asociatívny*, teda ak zabudnem zátvorky, tak ich chápe doľava

- **Int -> Int -> Int -> Char = Int -> (Int -> (Int -> Char))** … lebo operátor funkčného typu **->** je *pravo asociatívny*, teda ak zabudnem zátvorky, tak ich chápe doprava. Explicitne, (Int->Int) -> (Int -> Int)

- **Int -> Int -> String != (Int, Int) -> String**… lebo prvé je funkcia, ktorá vráti funkciu, ktorá vráti String. Vďaka *currying* ju volám takto f 4 5, čo je (f 4) 5. Druhé je funkcia, ktorá čaká dvojicu. Musím ju volať takto: g (4,5), a vyzerám, že som Javista, a na Haskelli prvý týždeň**...**

- **Int != Integer** … lebo Int z interval minBound::Int **...** maxBound::Int =9223372036854775807=2^63-1, ergo to je **long**. Integer je BigInteger

- ako sa konvertuje **Int, Integer, Float** … to neviem ani ja, googlim…

Všetko,čo potrebujem vedieť, ma mali naučiť v materskej škôlke

- **Klauzálna definícia:**

~~slova 0 = [ ]~~
slova 0 = [[]]   -- to isté ako slova 0 = [" "]
slova k = [ch:w | w <- slova (k-1), ch <- "ABCDEF"]

- **Aritmetický pattern** už nie je podporovaný:

slova ~~**(k+1)**~~ = [ch:w | w <- slova k, ch <- "ABCDEF"]

- **Guards** alias bachari, či strážci:

slova k **| k == 0**   = [[]]
slova   **| otherwise** = [ch:w | w <- slova (k-1), ch <- "ABCDEF"]

- **where** patrí klauzule a nie je to výraz:

slova k | k == 0   = [[]]
slova | otherwise = [ch:w | w <- ws, ch <- "ABCDEF"]
                                        where ws = slova (k-1)

slova.hs

Bojím sa spýtať, čo všetko ma nenaučili v materskej škôlke...

Na typoch záleží (aj keď ‛detstvo’ bez nich bolo krásne a jednoduché):

- [[t]] nikdy nebude [t] (List<List<Integer>> nie je List<Integer>)

preto nemôžem napísať

- [ch+(slova k) | ch <- "ABCDEF"]

```
>:type "ABCDEF"
"ABCDEF" :: [Char]
```

slova k :: [String] == [[Char]], … lebo **type** String = [Char]

ch+(slova k) znamená Char + [[Char]]

Okrem toho, zreťazenie zoznamov je (**++**) :: [t] -> [t] -> [t]
Ale ani ch++(slova k) nie je dobre, lebo je to Char ++ [[Char]], nepasuje…

Prilep ako hlavu k zoznamu je (**:**) :: t -> [t] -> [t]
Ale ani ch : (slova k) nie je dobre, lebo je to Char : [[Char]], nepasuje…

Píšte (si) typy (kdekoľvek sa dá), **sú zdravé**, a hlášky GHC potom čitateľnejšie

Slova, která jsem si přál napsat sám – Robert Fulghum

```
module Slova where
import Data.List   -- pozrite si, koľko užitočných funkcií obsahuje

slova   :: Int -> [String]
slova   0   = [[]]
slova   k = [ ch:w | w <-slova (k-1), ch <- "ABCDEF"]

slova’  :: Int -> [String]
slova’  0   = [[]]
slova’  k = slova' (k-1) ++ [ ch:w | w <-slova' (k-1), ch <- "ABCDEF"]
```

```
length $ slova 3 = 216
```

```
length $ slova' 2 = 49 != 1+6+36 = 43
slova' 2 =
["","A","B","C","D","E","F","A","B","C","D
DA","EA","FA","AB","BB","CB","DB","EB",
,"EC","FC","AD","BD","CD","DD","ED","FD
E","FE","AF","BF","CF","DF","EF","FF"]
```

O(n^2). The nub function removes duplicates. The name nub means `essence’.)

```
length $ nub $ slova' 2 = 43
```

koľko je 1+6+36+...+6^k (počet slov dĺžky najviac k) ?

[1,7,43,259,1555,9331,55987,335923,2015539,12093235,72559411, ...]

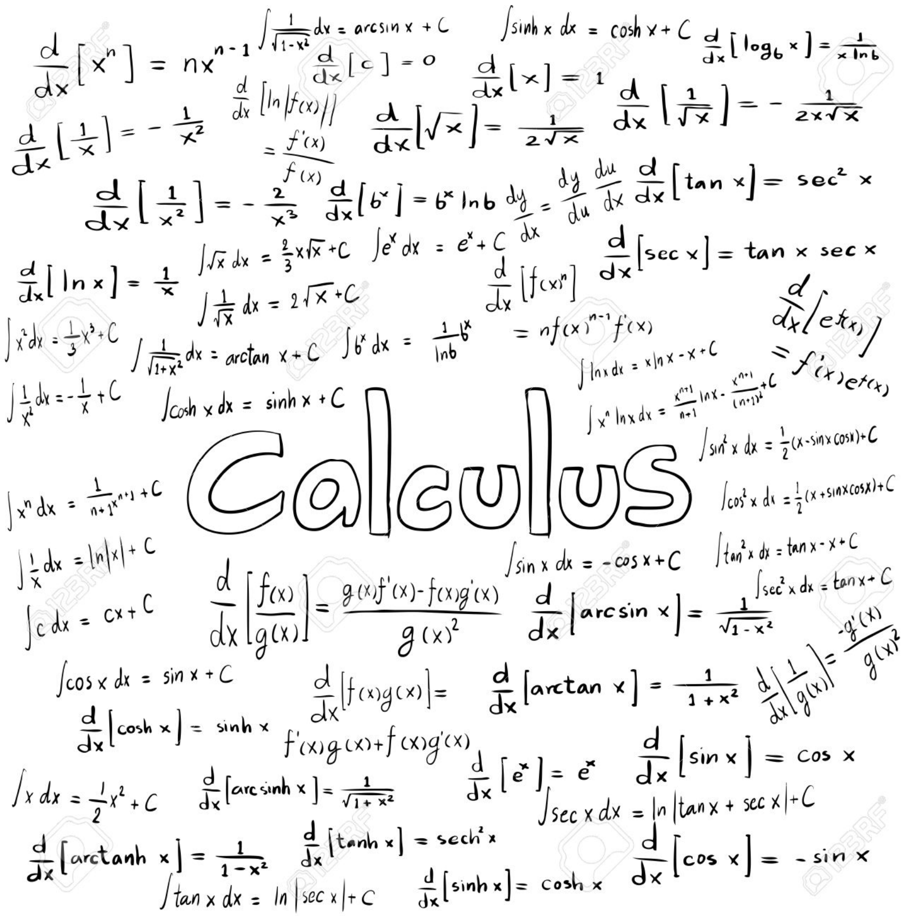
where:

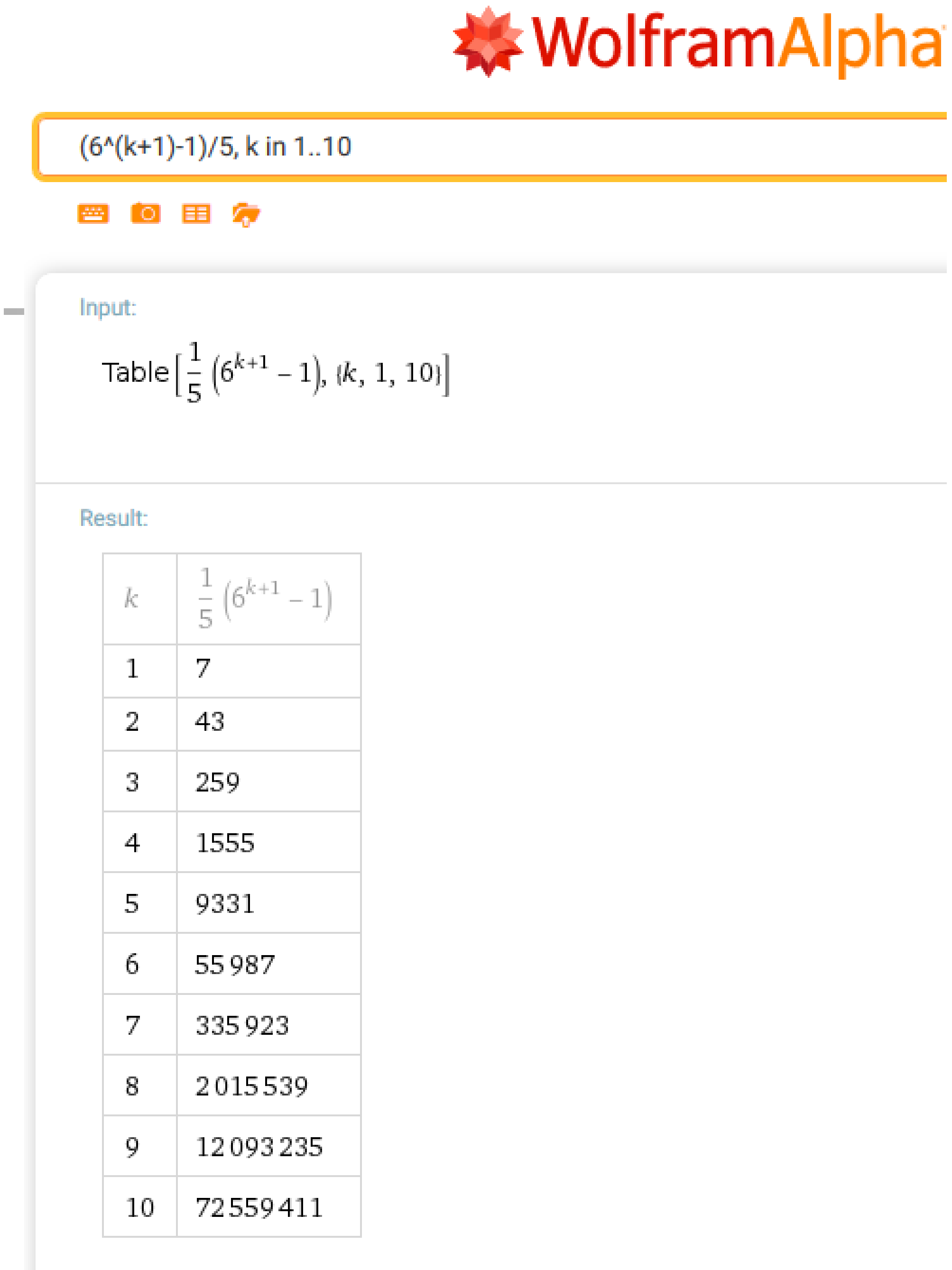
```
slova" k = ws ++ [ ch:w | w <-ws, ch <- "ABCDEF"]  where ws = slova" (k-1)
```

let:

```
slova"' k = let ws = slova"' (k-1) in ws ++ [ ch:w | w <-ws, ch <- "ABCDEF"]
```

slova.hs


Calculus
$\frac{d}{dx}[x^n] = nx^{n-1}$
$\int \frac{1}{\sqrt{1-x^2}}dx = \arcsin x + C$
$\int \sinh x\,dx = \cosh x + C$
$\frac{d}{dx}[\log_b x] = \frac{1}{x\ln b}$
$\frac{d}{dx}[c] = 0$
$\frac{d}{dx}[x] = 1$
$\frac{d}{dx}\left[\frac{1}{x}\right] = -\frac{1}{x^2}$
$\frac{d}{dx}[\ln|f(x)|] = \frac{f'(x)}{f(x)}$
$\frac{d}{dx}[\sqrt{x}] = \frac{1}{2\sqrt{x}}$
$\frac{d}{dx}\left[\frac{1}{\sqrt{x}}\right] = -\frac{1}{2x\sqrt{x}}$
$\frac{d}{dx}\left[\frac{1}{x^2}\right] = -\frac{2}{x^3}$
$\frac{d}{dx}[b^x] = b^x \ln b$
$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$
$\frac{d}{dx}[\tan x] = \sec^2 x$
$\frac{d}{dx}[\ln x] = \frac{1}{x}$
$\int \sqrt{x}\,dx = \frac{2}{3}x\sqrt{x} + C$
$\int e^x dx = e^x + C$
$\frac{d}{dx}[\sec x] = \tan x \sec x$
$\int \frac{1}{\sqrt{x}}dx = 2\sqrt{x} + C$
$\frac{d}{dx}[f(x)^n] = nf(x)^{n-1}f'(x)$
$\frac{d}{dx}[e^{f(x)}] = f'(x)e^{f(x)}$
$\int x^2 dx = \frac{1}{3}x^3 + C$
$\int \frac{1}{1+x^2}dx = \arctan x + C$
$\int b^x dx = \frac{1}{\ln b}b^x$
$\int \ln x\,dx = x\ln x - x + C$
$\int \frac{1}{x^2}dx = -\frac{1}{x} + C$
$\int \cosh x\,dx = \sinh x + C$
$\int x^n \ln x\,dx = \frac{x^{n+1}}{n+1}\ln x - \frac{x^{n+1}}{(n+1)^2} + C$
$\int \sin^2 x\,dx = \frac{1}{2}(x - \sin x\cos x) + C$
$\int x^n dx = \frac{1}{n+1}x^{n+1} + C$
$\int \cos^2 x\,dx = \frac{1}{2}(x + \sin x\cos x) + C$
$\int \frac{1}{x}dx = \ln|x| + C$
$\int \sin x\,dx = -\cos x + C$
$\int \tan^2 x\,dx = \tan x - x + C$
$\int \sec^2 x\,dx = \tan x + C$
$\int c\,dx = cx + C$
$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{g(x)^2}$
$\frac{d}{dx}[\arcsin x] = \frac{1}{\sqrt{1-x^2}}$
$\int \cos x\,dx = \sin x + C$
$\frac{d}{dx}[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$
$\frac{d}{dx}[\arctan x] = \frac{1}{1+x^2}$
$\frac{d}{dx}\left[\frac{1}{g(x)}\right] = \frac{-g'(x)}{g(x)^2}$
$\frac{d}{dx}[\cosh x] = \sinh x$
$\frac{d}{dx}[\sin x] = \cos x$
$\int x\,dx = \frac{1}{2}x^2 + C$
$\frac{d}{dx}[\text{arcsinh}\, x] = \frac{1}{\sqrt{1+x^2}}$
$\frac{d}{dx}[e^x] = e^x$
$\int \sec x\,dx = \ln|\tan x + \sec x| + C$
$\frac{d}{dx}[\text{arctanh}\, x] = \frac{1}{1-x^2}$
$\frac{d}{dx}[\tanh x] = \text{sech}^2 x$
$\frac{d}{dx}[\cos x] = -\sin x$
$\int \tan x\,dx = \ln|\sec x| + C$
$\frac{d}{dx}[\sinh x] = \cosh x$



WolframAlpha

(6^(k+1)-1)/5, k in 1..10

Input:

$$\text{Table}\left[\frac{1}{5}\left(6^{k+1}-1\right), \{k, 1, 10\}\right]$$

Result:

k	$\frac{1}{5}\left(6^{k+1}-1\right)$
1	7
2	43
3	259
4	1555
5	9331
6	55 987
7	335 923
8	2 015 539
9	12 093 235
10	72 559 411

TED Ideas worth spreading

Stephen Wolfram | TED2010

Computing a theory of all knowledge

https://www.ted.com/talks/stephen_wolfram_computing_a_theory_of_everythinc

Všetko, čo ste chceli zmeniť, a nikdy sa vám to nepodarilo

- zoznam (“pole”) xs vieme indexovať indexami i <- [0..length xs-1]

```
        xs!!i                                   -- getter
```

- neexistuje setter xs[i] = value

```
set  ::  [t] -> Int -> t -> [t]
set xs i value | i < 0                 = xs    -- out of range
               | i >= length xs        = xs    -- out of range
               | otherwise             = (if i == 0 then value else y):set ys (i-1) value
                                         where (y:ys) = xs
               | otherwise             = let (y:ys) = xs in
                                         (if i == 0 then value else y):set' ys (i-1) value
set''  ::  [t] -> Int -> t -> [t]
set'' xs i value | i < 0               = xs    -- out of range
                 | i >= length xs      = xs    -- out of range
                 | otherwise           = [xs!!j | j <- [0.. i-1] ] ++ [value] ++
                                           [xs!!j | j <- [i+1..length xs-1] ]
```

zoznam.hs

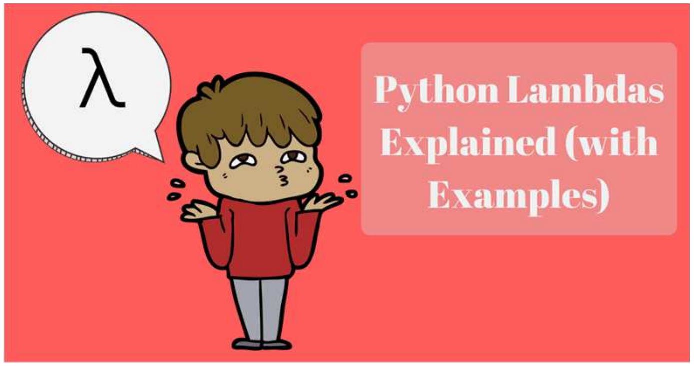
Haskell homework tu nevidím...

- Share everything.
- Play fair.
- Don’t hit people.
- Put things back where you found them.
- Clean up your own mess.
- Don’t take things that aren’t yours.
- Say you’re sorry when you hurt somebody.
- Wash your hands before you eat.
- Flush.
- Warm cookies and cold milk are good for you.
- Live a balanced life—learn some and think some and draw and paint and sing and dance and play and work every day some.
- Take a nap every afternoon.

Robert
Fulghum
VŠECHNO, CO OPRAVDU POTŘEBUJU ZNÁT, JSEM SE NAUČIL V MATEŘSKÉ ŠKOLCE

https://sk.wikipedia.org/wiki/Robert_Fulghum

Python Kvíz



```
print(map(lambda x: x*x, [1,2,3,4,5]))              <map object at 0x037
print(list(map(lambda x: x*x, [1,2,3,4,5])))        [1, 4, 9, 16, 25]
print(list(filter(lambda y:y>10,map(lambda x: x*x, [1,2,3,4,5]))))
                                                    [16, 25]

from functools import reduce
print(reduce((lambda x, y: x * y), [1, 2, 3, 4]))   24

print(reduce((lambda x, y: x + y), [1, 2, 3, 4]))   10
print(reduce((lambda x, y: x - y), [1, 2, 3, 4]))   -8

def compose(f, g):
        return lambda x: f(g(x))
print(compose( lambda x: x+1, lambda x: x*3 )(10))  31

def composeMany(*fs):
                                                    33
        return reduce(compose, fs)
print(composeMany(lambda x:x+1, lambda x:x+2, lambda x:x*3)(10))
```

lambdas.hs

Does not matter much…

for job: Better choice would be Scala (modern Java)
https://www.coursera.org/learn/progfun1

for school: Haskell

List-comprehension

Každý poriadny kurz FP začína funkcionálmi map a filter:

...ale my sme trénovali list-comprehension:

```
                    [ f x | x <- xs, p x]       [ f(x) for x in xs if p(x)]
```

```
map            ::   (a -> b) -> [a] -> [b]
map f xs       =    [ f x | x <- xs]

filter         ::   (a -> Bool) -> [a] -> [a]
filter p xs    =    [ x | x <- xs, p x]
```