

Beginning Haskell

~~f(a, b)~~

f a b

sqrt 2 + sqrt 3

map sqrt [1..5]

filter odd [1..5]

[1, 3, 5]

```
map (*2) [1..5]  
[2, 4, 6, 8, 10]
```

```
filter (>3) [1..5]  
[4, 5]
```

```
map (^2) [1..5]  
[1, 4, 9, 16, 25]
```

```
map (2^) [1..5]  
[2, 4, 8, 16, 32]
```

number }
number } $\xrightarrow{(\ *)}$ *number*

number $\xrightarrow{(\ * 2)}$ *number*

```
map (^2)  
  (filter odd [1..5])
```

```
sum  
  (map (^2)  
    (filter odd [1..5]))
```

`sin (cos (tan (sqrt 2)))`

~~`sin cos tan sqrt 2`~~

`sin (some stuff)`

`sin $ some stuff`

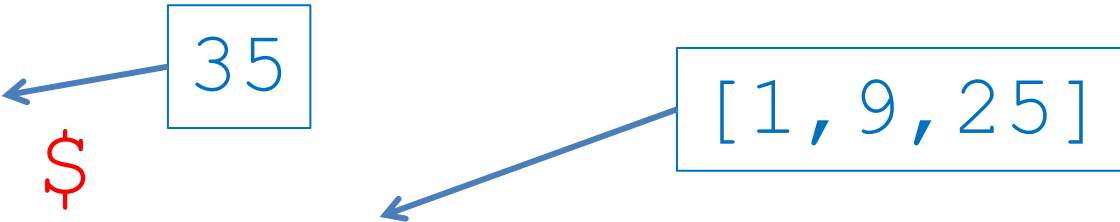
`sin $ cos (tan (sqrt 2))`

`sin $ cos $ tan $ sqrt 2`

```
map (^2) (filter odd [1..5])
```

```
map (^2) $ filter odd [1..5]
```

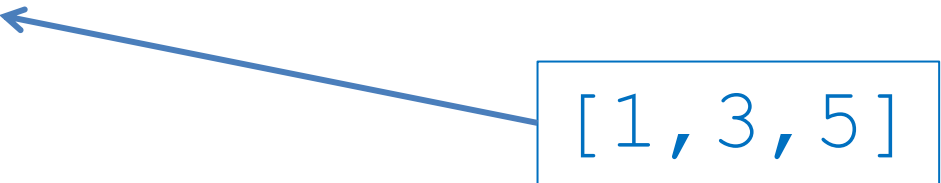
```
sum $
```



35

[1, 9, 25]

```
map (^2) $  
  filter odd [1..5]
```



[1, 3, 5]

*The sum of
the squares of
some odd numbers*

[1..]

Demo

```
takeWhile (<100) $  
  map (2^) [1..]  
      [2, 4, 8, 16, 32, 64]
```


Exercises 1

<http://dominicprior.github.io/>

Defining Functions

$$\text{sq } x = x^2$$

$$\text{sq} = \backslash x \rightarrow x^2$$

$$\text{sq} = (^2)$$

$$\text{h } x \ y = x + 2 * y$$

Factorial

$$f\ 0 = 1$$

$$f\ n = n * f\ (n-1)$$

$$4! = 4 * 3!$$

$$f\ 2$$

$$2 * f\ 1$$

$$2 * (1 * f\ 0)$$

$$2 * (1 * 1)$$

[2, 3] ++ [4, 5, 6]

[2, 3, 4, 5, 6]

~~1++~~

[5, 6, 7] !! 2

7

4 : [5, 6]

[4, 5, 6]

[2, 4]

2 : [4]

2 : (4 : [])

2 : 4 : []

"hi" ['h', 'i']

"hello" !! 1 'e'

Writing our own sum function

`sum [] = 0`

`sum [2, 3, 4] = 2 + sum [3, 4]`

`sum (x:xs) = x + sum xs`

Exercises 2

<http://dominicprior.github.io/>

Types

Int		[Int]
Integer		[Integer]
Float		[Float]
Double		[Double]
Char	<i>'a'</i>	[Char]
String	<i>"hi"</i>	[String]
Bool	<i>True</i>	[Bool]
	<i>False</i>	<div>[[Int]]</div>

Int	[Int]
Integer	[Integer]
Float	
Double	(Int, String) (3, "hi")
Char	(Char, Bool, [Int])
String	Set Int
Bool	Set Integer

odd :: Int -> Bool

length :: [a] -> Int

take :: Int -> [a] -> [a]

sum :: Num a => [a] -> a

map sqrt [1..5]

map :: some function -> a list ->
another list

map :: (a -> b) -> [a] -> [b]

Exercises 3

<http://dominicprior.github.io/>

```
map (*2) [1,2,3]
```

```
zipWith (*) [1,2,3] [4,5,6]  
[4,10,18]
```

```
foldr (+) 0 [1,2,3] 6
```

```
foldl' (+) 0 [1,2,3] 6
```

```
map (sqrt . sqrt) [1,2,3]
```

1/0

`f :: String -> String`

`g :: String -> IO String`


`readFile :: String -> IO String`

`getChar :: IO Char`

`putStr :: String -> IO ()`

`main :: IO ()`

```
readFile :: String -> IO String
```



```
main = do  
    str <- readFile "foo.txt"  
    let str' = take 100 str  
    writeFile "a.txt" str'
```

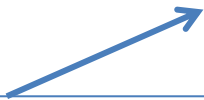


```
writeFile :: String -> String -> IO ()
```

```
main = do
  str <- getContents
  let str' = f str
  putStr str'

f = take 100
```

```
main = interact $ take 100
```



```
interact :: (String -> String) -> IO ()
```

Exercises 4

<http://dominicprior.github.io/>

Laziness

Defining our own map function

`map :: (a -> b) -> [a] -> [b]`

`map _ [] = []`

`map f [1,2,3] = [f 1, f 2, f 3]`

`= f 1 : [f 2, f 3]`

`= f 1 : map f [2,3]`

`map f (x:xs) = f x : map f xs`