Why Haskell?

Siddharth Bhat

IIIT Theory group
Seminar Saturday

October 18th, 2019

Why should I learn haskell?

docs.python.org/3/library/stdtypes.html#str.join

hackage.haskell.org/package/base-4.14.0.0/docs/Data-List.html#v:intercalate

```
" ".join(["a", "b", "c", "d"])
```

`str.join(iterable)`

Return a string which is the concatenation of the strings in *iterable*. A `TypeError` will be raised if there are any non-string values in *iterable*, including `bytes objects`. The separator between elements is the string providing this method.

```
intercalate " " ["a", "b", "c", "d"]
```

```
intercalate :: [a] -> [[a]] -> [a]
```

`intercalate xs xss` is equivalent to `(concat (intersperse xs xss))`. It inserts the list `xs` in between the lists in `xss` and concatenates the result.

Why should I learn haskell?

https://docs.python.org/3/library/functions.html#sum

hackage.haskell.org/package/base-4.14.0.0/docs/Data-List.html#v:intercalate

```
sum([1, 2, 3, 4])
```

```
sum(iterable, /, start=0)
```

Sums *start* and the items of an *iterable* from left to right and returns the total. The *iterable*'s items are normally numbers, and the start value is not allowed to be a string.

```
sum [1, 2, 3, 4]
```

```
sum :: (Foldable t, Num a) => t a -> a
```

The `sum` function computes the sum of the numbers of a structure.

Foldable in detail

```haskell
class Foldable t where
  -- | Map each element of the structure to a monoid, and combine the results.
  foldMap :: Monoid m => (a -> m) -> t a -> m
```

Foldable instances are expected to satisfy the following laws:

```haskell
foldr f z t = appEndo (foldMap (Endo . f) t ) z
foldl f z t = appEndo (getDual (foldMap (Dual . Endo . flip f) t)) z
fold = foldMap id
length = getSum . foldMap (Sum . const  1)
```

https://hackage.haskell.org/package/base-4.14.0.0/docs/Data-Foldable.html#t:Foldable

Fibonacci

```haskell
let fib = 0:1:(zipWith (+) fib (tail fib))
```

Equational reasoning

```
k x y = x
k 10 (error "urk")
```

```
def k(x, y): return x
k(x, input())
```

What is `input` anyway?

$$\emptyset \mapsto \texttt{char}$$

Mathematically impossible!

Effects, or the "M" word

Keep every element, and drop every element.

```haskell
powerset xs = filterM (const [True, False]) xs
```