

Introduction to Haskell and its REPL





A close-up photograph of two hands holding a small, rectangular piece of white paper against a solid black background. The paper has a jagged, torn edge on its right side. The text "I can't" is written on the paper in a bold, black, sans-serif font. The word "I" is on the left, followed by "can", and the apostrophe and "t" are on the right side of the paper. The hands are positioned on either side of the paper, with the thumbs and index fingers visible, holding the edges. The skin tone of the hands is light, and the fingernails are painted a light pink color. The lighting is soft, highlighting the texture of the paper and the skin.

I can't

1.

```
$ docker run -it --rm \  
  mitchty/alpine-ghc ghci
```

1.

```
$ docker run -it --rm \  
    mitchty/alpine-ghc ghci
```

```
docker pull mitchty/alpine-ghc:latest
```

So... what now?

- $1 + 2$
- 2^{1000}
- $(+) 1 2$
- $:t (+)$
- $:i (+)$

Functions Everywhere

■ $(+) :: a \rightarrow a \rightarrow a$

```
In [1]: def add2Integers(a, b):  
        ...:     return a + b  
        ...:  
  
In [2]: add2Integers(1, 2)  
Out[2]: 3
```

List

- [1, 2, 3]
- [1..]
- [1..10]
- [1, 1.25 .. 4.0]

List

- $(:) :: a \rightarrow [a] \rightarrow [a]$
- $(++) :: [a] \rightarrow [a] \rightarrow [a]$

List

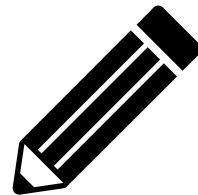
- `head :: [a] -> a`
- `tail :: [a] -> [a]`
- `take :: Int -> [a] -> [a]`
- `drop :: Int -> [a] -> [a]`

List

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

GHCI

- `:t <expression>` ← type inspection
- `:i <expression>` ← info
- `:l <filename>` ← load .hs file
- `:r` ← reload files



BYO Editor

- Spacemacs <http://spacemacs.org/>
- Atom <https://atom.io/>
- Visual Studio <https://code.visualstudio.com/>

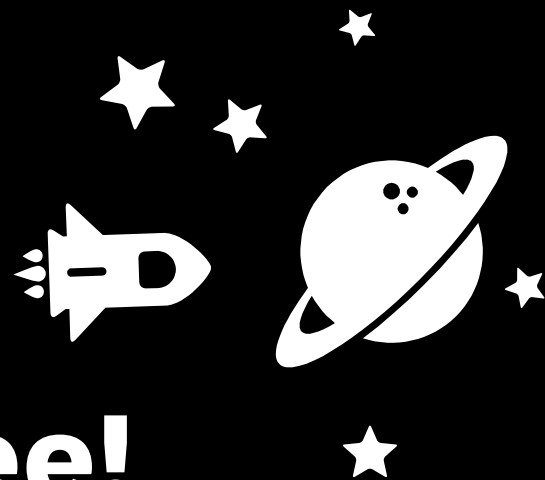
2.

```
$ docker run -it --rm \  
  -v `pwd`: /root \  
  mitchty/alpine-ghc ghci
```

Wanna try an exercise together?

Let's reverse a list





Theorems for free!

– Philip Walder, 1989

<https://people.mpi-sws.org/~dreyer/tor/papers/wadler.pdf>

What's the only possible implementation of `foo`?

- `foo :: a -> a`
- `foo2 :: b -> a`
- `foo3 :: a -> b -> a`

A photograph of a gravel path leading into a forest. The path is made of small, light-colored stones and leads from the bottom center towards the background. The background is filled with green trees and foliage, which are out of focus, creating a bokeh effect. The lighting is soft and natural, suggesting a sunny day. The entire image is framed by a white border.

A journey of a thousand miles

**BEGINS WITH
A SINGLE STEP.**

Lao Tzu

Thanks!



Any questions?

@filippovitale