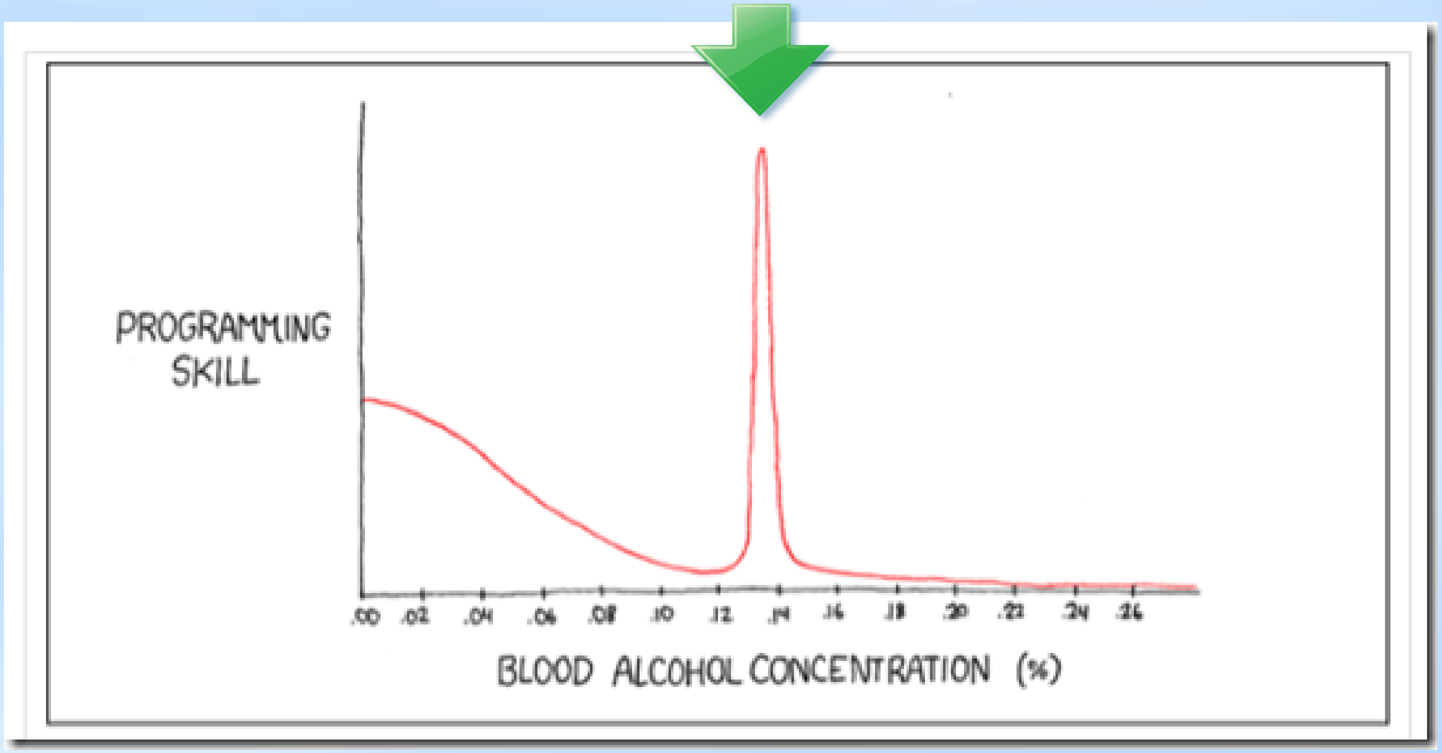
HASKELL!
(by Thomas)

Haskell



https://xkcd.com/323/

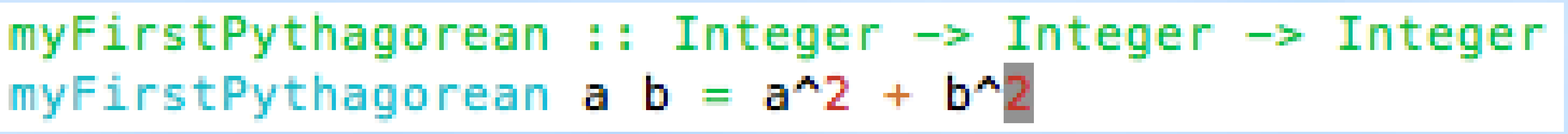
¿Que es Haskell?

- Lazy, pure functional programming language
- Only lazy language that's still around
- WHY?!?!?!?!?!
  - Types!
  - Correctness!
    - ONLY IMPORTANT THING EVER
    - Fast & WRONG < slow & RIGHT
    - Most wrong programs give you wrong answers slowly

Dysfunctional programming???

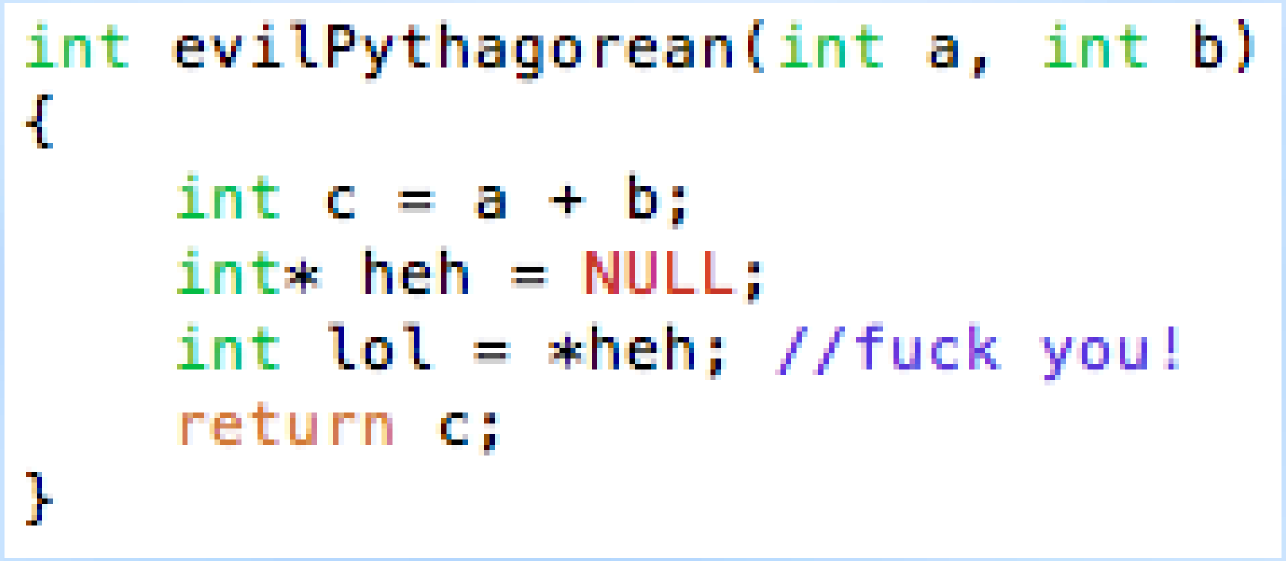
- Math theorem:
  - a^2 + b^2 = c^2
  - But what if *a* isn't a number?? #cproblems
- A function = no side effects
  - What happens if the pythagorean theorem connects to the internet?
- C doesn't have functions
  - *Procedures*, please

:)
Time for some code!



```haskell
myFirstPythagorean :: Integer -> Integer -> Integer
myFirstPythagorean a b = a^2 + b^2
```

Haskell won't crash your computer



```
int evilPythagorean(int a, int b)
{
    int c = a + b;
    int* heh = NULL;
    int lol = *heh; //fuck you!
    return c;
}
```

- Imperative languages play dirty
- Access a file, overwrite global variables, connect to the internet
- Total anarchy

“*Psh,* I'm the master of state”

- How much state is there?
  - AKA how much stuff can C/C++/Java destroy?
- A bunch of memory…
- A lot of disk space…
- The entire internet…
- Time
- Cannot believe you all trust library authors not to break these things

Purity

- Pure = no side effects
- Impure = has side effects
- IO type
  - Main
- Math functions are pure, close() is not
  - What if the file is on remote storage?
  - What if the file doesn't exist?

IO

- IO infects **everything**
- IO a
  - An **action** that *returns* a
- IO ()
  - Basically void
- IO code can *unwrap* IO values and pass them to pure code
- putStrLn :: String → IO ()
- readFile :: FilePath → IO String

IO

- Why is this “unsafe”?
  - unsafePerformIO :: IO a → a
- Used pretty exclusively for interfacing with C
  - Example: regex lib

Laziness

- At the end of the day we want a result
- The main function: main :: IO ()
  - A big chain of IO functions
- *Forcing* a value = actually doing work
- Walk through main until the end

Consequences of Laziness

- Unemployment
- Infinite lists
  - So long as you eventually use <infinity
- *Must* be pure
  - If you never know when or if something will run, have to parameterize everything
  - Somewhat hard transition to writing idiomatic Haskell
- Can force strictness if you want
  - e.g. read from disk
  - Doesn't propagate up!

Laziness example

Time for some syntax

Type Signatures

- Type inference
  - Like Java Generics on steroids
  - Compiler infers most general type
  - Can write your own type signature whenever you want to restrict domain
- Can write type signatures wherever you want

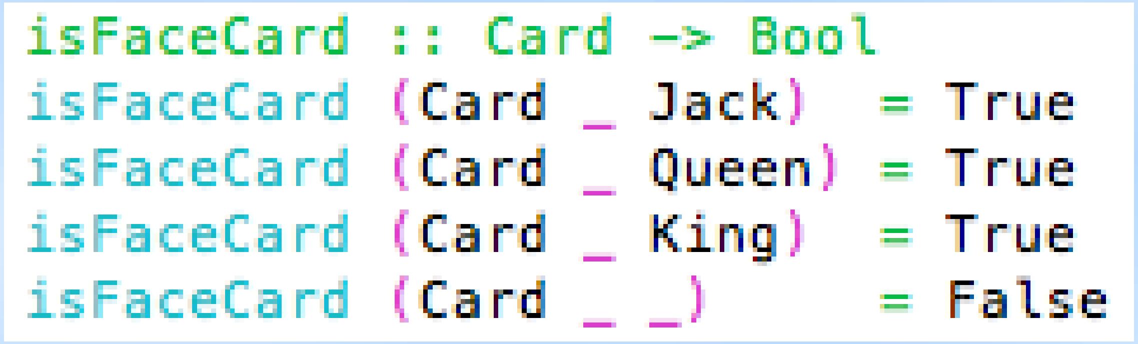
foo :: some_input → another_input → some_output

Data Types

(Time to show you the example!)

Pattern Matching

- Do something for each constructor
- Unwraps values
- Underscore matches anything



```haskell
isFaceCard :: Card -> Bool
isFaceCard (Card _ Jack)  = True
isFaceCard (Card _ Queen) = True
isFaceCard (Card _ King)  = True
isFaceCard (Card _ _)     = False
```

- Suit doesn't matter for face cards
- Anything that isn't a Jack, Queen, or King isn't a face card

Working with Lists

- No loops
  - Loops are imperative: *do this x many times*
- Only recursion
- Map and fold are the workhorses
  - map :: (a → b) → [a] → [b]
    - Do something for every item in the list
  - foldr :: (a → b → b) → b → a → b
    - Woah!
    - Will become clear
    - Used to accumulate a value

map and fold examples

Practical Stuff

- ***But thomas, I wanna go fas!***
  - Memoization optimization
  - Never run a pure function twice
- Lots of compiler optimizations
  - Compiler knows WAY more about your program

Real World Examples

- Huge list: https://wiki.haskell.org/Haskell_in_industry
- Pugs, perl6 interpreter
- Quickcheck
- https://www.fpcomplete.com/wp-content/uploads/2013/05/Bump%20case%20study.pdf
  - Purity is WAY better for concurrency!

Pandoc!

- Document converter and library
- Document conversion = super functional
- SO MANY FORMATS!
- Awesome
- http://pandoc.org/diagram.jpg

Why should I care?

- Haskell drastically changes how you think about programming
  - More aware of side effects = less bugs
- Worth learning even if you never write anything meaningful in it!
- I haven't even covered anything crazy (monads, functors, etc.)
  - IT WILL BLOW YOUR MIND!