

The language of Phi: Whitepaper

Rishi Kothari

Contents

1	Motivations	1
2	Design	2
2.1	Syntax	2
2.2	Output	2
2.3	Basics	2
2.3.1	Ambiguity	2
2.4	Relations to Mathematics	3
2.4.1	Lambda Calculus	3
2.5	Tokenization	3
3	Grammar	4

Summary

Mathematics and Computer Science are deeply intertwined; some universities even offer CS as a math course. However,

Chapter 1

Motivations

Computer Science is arguably one of the most interesting fields of **math**, so why is there such a big disconnect between the math one learns in university and high school CS and the programming language of choice?

Take the example of a simple sum function in math:

$$f(a, b) = \sum_b^a b, \{a, b\} \in \mathbb{N}$$

This is an example of Sigma notation, a much-used concept in many parts of math. Taking a look at the equivalent expression in Python,

```
1  def sum_loop (top_bound, start):
2      accumulator = 0
3      for i in range(start, top_bound+1):
4          accumulator += i
5          i+=1
6      return accumulator
```

One might see that the two have absolutely nothing in common.

This may not seem like a problem; the programmer just needs to learn programmatical intuition. However, this can pose a challenge for a *mathematician* to learn programming, because of the existing mathematical intuition that needs to be replaced

Chapter 2

Ideology, Design

2.1 Syntax

Φ is a mathematical language, and so needs to be designed with rigour in mind. To understand what is meant by this, reading "[The design side of programming language design](#)", by Tomas Petricek is advised.

2.2 Output

2.3 Basics

2.3.1 Ambiguity

Φ is a language that doesn't allow for any ambiguity. All characters and keywords must have *one* purpose.

For instance, in Swift, the `:` operator serves at least functions: One for type annotation, and the other for assigning values:

```
1 //Use one: Type annotation
2 var x : String = "Hello World"
3
4 func printString(str : String) -> Void {
5     print(str)
6 }
7
8 //Use two: Value assignment
9 printString(str: x)
```

The fact that the `:` operator can be used in multiple cases creates ambiguity, and Φ aims to solve that.

In Φ 's case, the `:` operator will only be used for type annotation, and the `=` operator will be used as an assignment operator.

Similarly, the `?` operator will serve as an equality operator, in comparison statements:

```
if(equality: { x ?= y }, action: {
    print("x equals y")
})
```

```
if(equality: { x ?: y })
```

2.4 Relations to Mathematics

2.4.1 *Lambda Calculus*

2.5 Tokenization

Chapter 3

Grammar