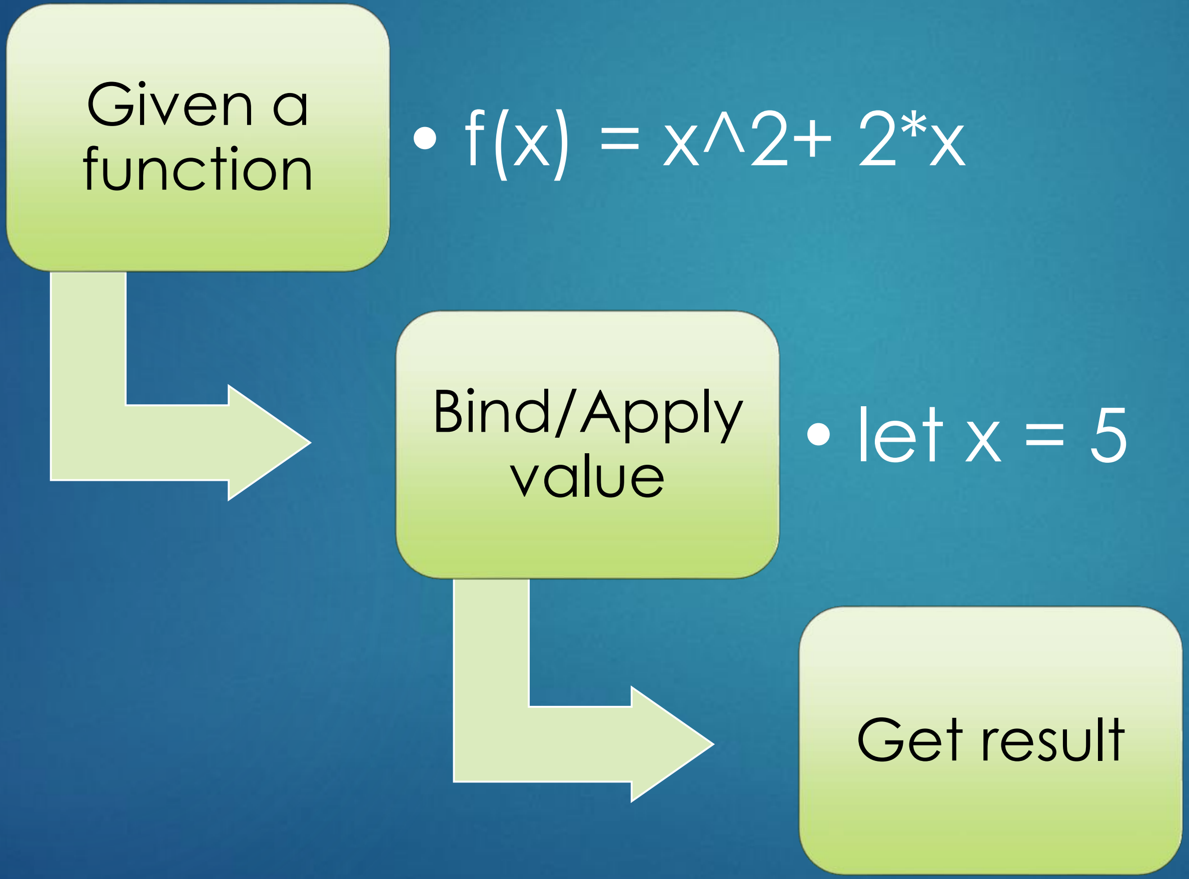
Functional Programming Explains

KIMMY LEO @ CUIT <KENPUSEY@OUTLOOK.COM>

A Mathematic Problem

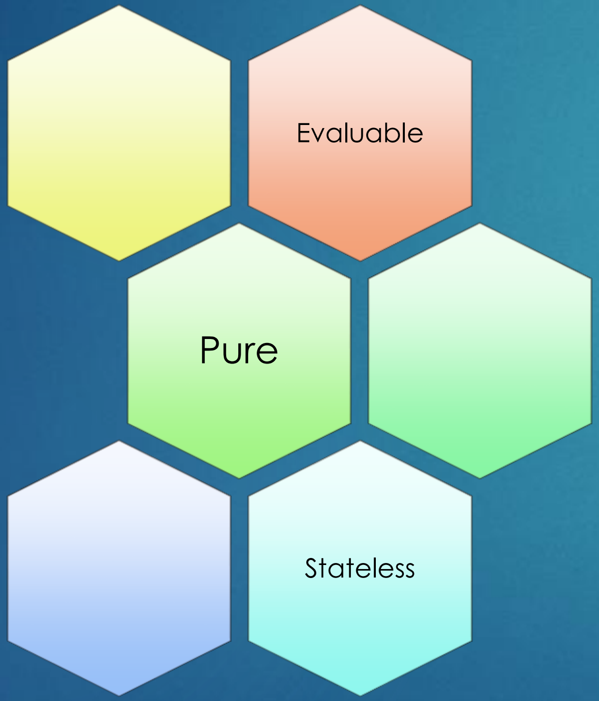


- $f(x) = x^2 + 2*x$
- let $x = 5$
- then $f(x) = 35$

Try others

f(x)=	sin(x)^2
	sqrt[log(x)/log(x^2+1)]
	3
g(x)=	x!
	$\begin{cases} f(x) & x \geq 0 \\ g(f(x)) & x < 0 \end{cases}$

Functions



- Evaluable
  - Always returns a specified value when given legal arguments
- Pure
  - Returns strictly same value when given same legal input
- Stateless
  - Function behavior dose not change.

Functional Programming

Technology?

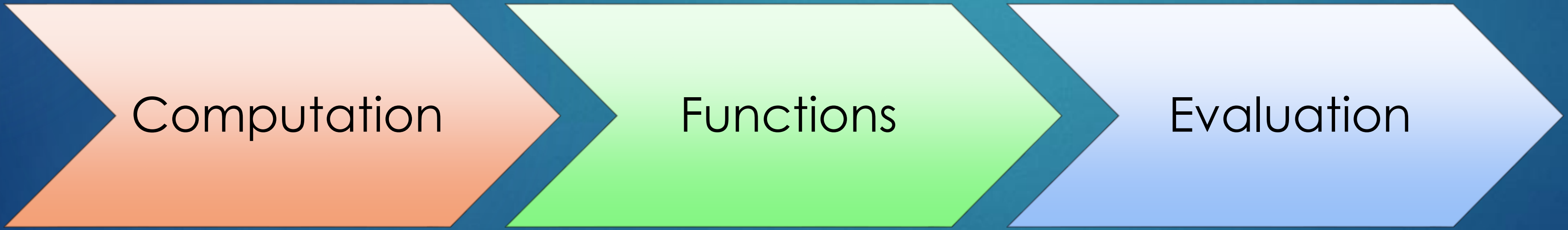
Pattern?

Style?

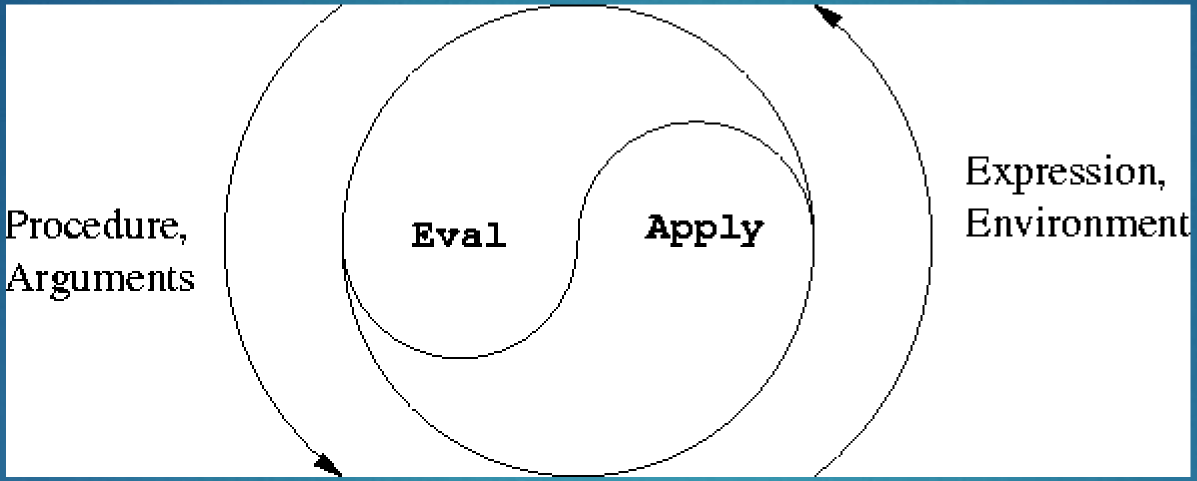
Convention?

Functional Programming

Paradigms	Object-oriented
	Aspect-oriented
	Message-driven
	Event-driven
	Functional



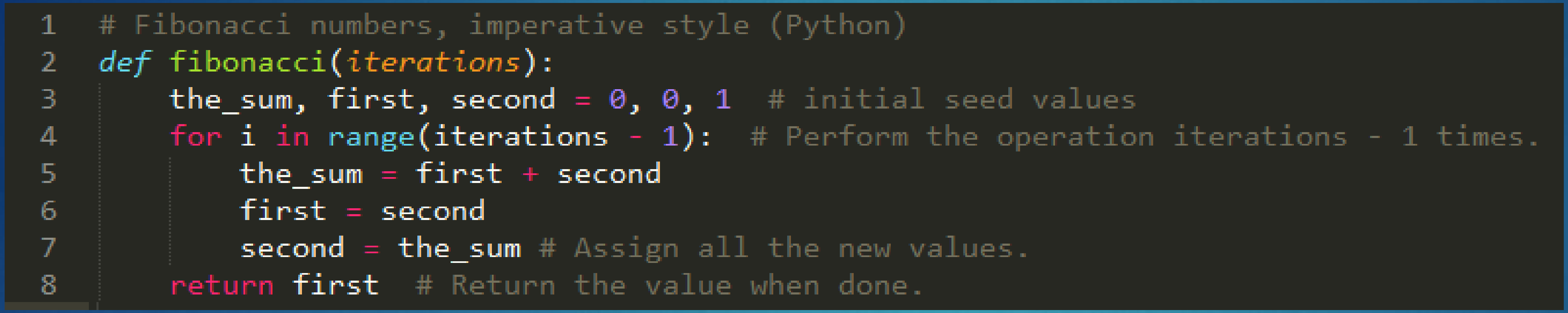
Functional Programming



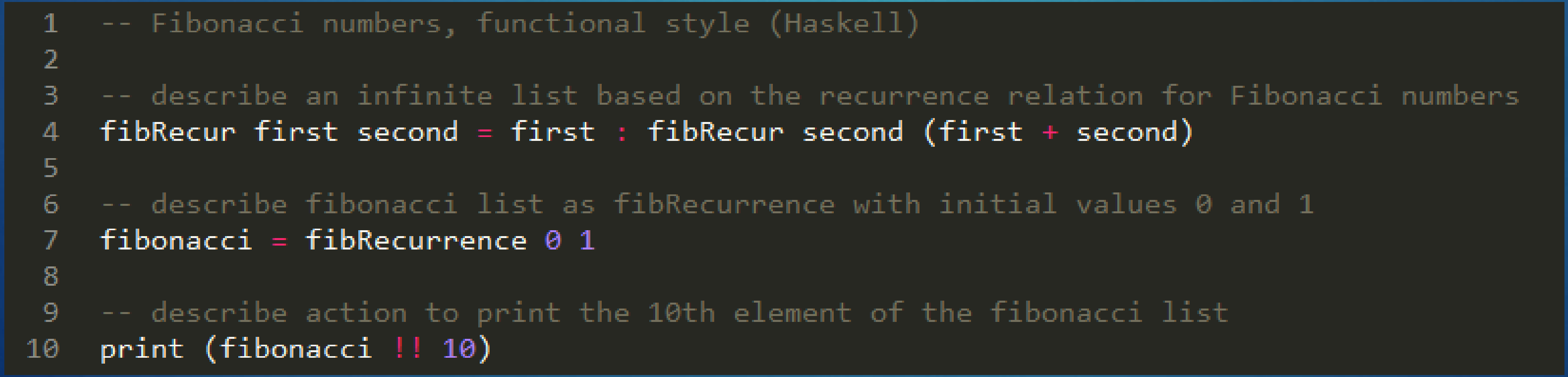
Principles

- Everything is (immutable)value
- Avoiding side-effect
- Data-flow based
- Bottom-up style
- Massive recursions & nested/chained calls

Real World Example (1)

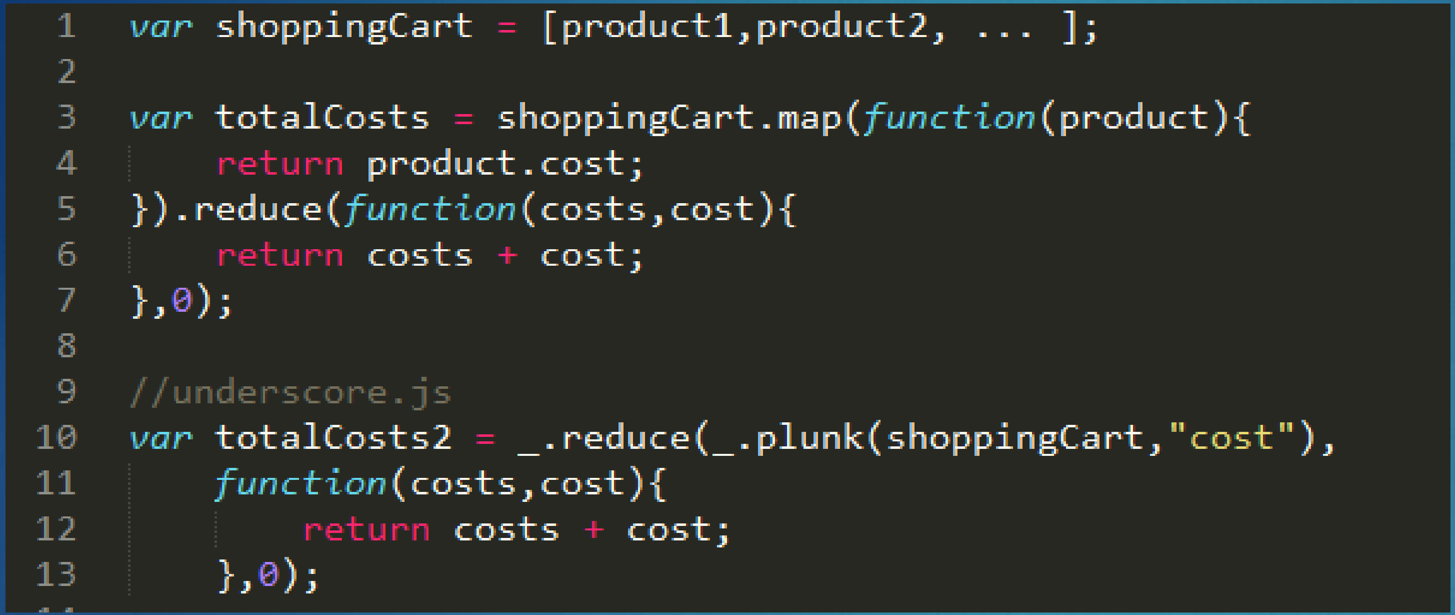


```python
# Fibonacci numbers, imperative style (Python)
def fibonacci(iterations):
    the_sum, first, second = 0, 0, 1  # initial seed values
    for i in range(iterations - 1):  # Perform the operation iterations - 1 times.
        the_sum = first + second
        first = second
        second = the_sum # Assign all the new values.
    return first  # Return the value when done.
```



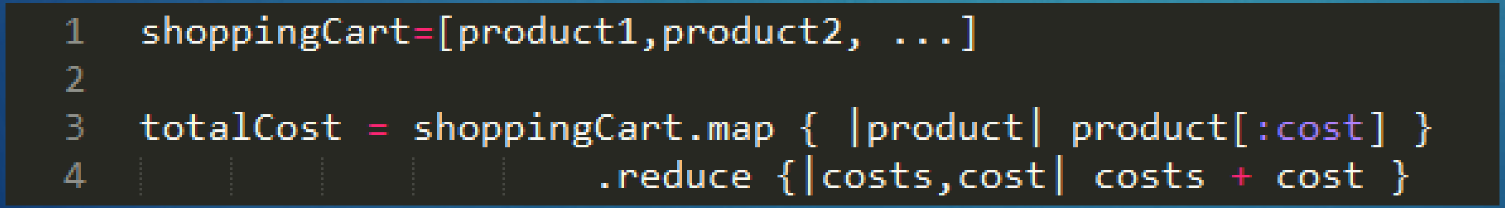
```haskell
-- Fibonacci numbers, functional style (Haskell)

-- describe an infinite list based on the recurrence relation for Fibonacci numbers
fibRecur first second = first : fibRecur second (first + second)

-- describe fibonacci list as fibRecurrence with initial values 0 and 1
fibonacci = fibRecurrence 0 1

-- describe action to print the 10th element of the fibonacci list
print (fibonacci !! 10)
```

Real World Example (2)



```javascript
var shoppingCart = [product1,product2, ... ];

var totalCosts = shoppingCart.map(function(product){
    return product.cost;
}).reduce(function(costs,cost){
    return costs + cost;
},0);

//underscore.js
var totalCosts2 = _.reduce(_.plunk(shoppingCart,"cost"),
    function(costs,cost){
        return costs + cost;
    },0);
```

//JavaScript



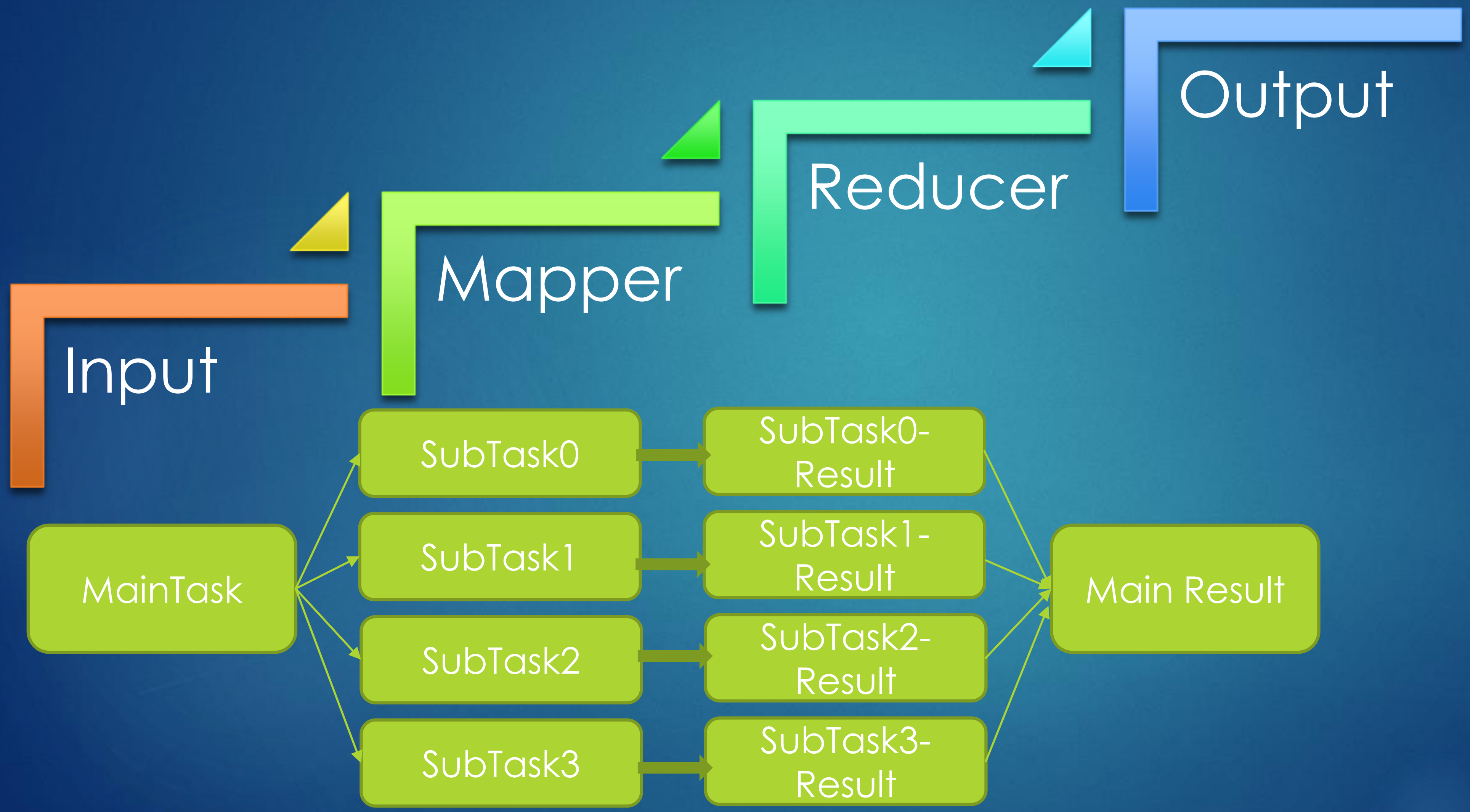
```ruby
shoppingCart=[product1,product2, ...]

totalCost = shoppingCart.map { |product| product[:cost] }
                     .reduce {|costs,cost| costs + cost }
```

//Ruby

Strengths

- Declarative Style
  - Clear / Simply
- Better List / KV-Map Processing
  - LISP (LISt Processing) (ancestor of functional programming)
  - Real World Data Schema
- Better for Parallelism
  - Multicore(process / thread) System
  - Distributing System
  - and the trends

MapReduce


Input
Mapper
Reducer
Output
MainTask
SubTask0
SubTask1
SubTask2
SubTask3
SubTask0-Result
SubTask1-Result
SubTask2-Result
SubTask3-Result
Main Result

Practical FP

- *Programming Language*
  - LISPs (scheme / Clojure …)
  - MLs (Standard ML / Haskell / F# …)
  - Others (C++11 / Java 8 / Scala , Python / Ruby …)
- *Facilities*
  - Anonymous Function (a.k.a lambda expression)
    - functor in early C++ & anonymous internal class in java
  - Closure
- *Libraries*
  - C++ STL Algorithms / Functor, Guava …
  - …

e.g.

- Map
  - Enumerable#map(Ruby), map(Python), std::transform(C++)
  - Iterables.transform(Java/Guava)
  - foldl(Haskell)…
- Reduce
  - Enumerable#reduce, reduce, std::accumulate …
- Filter
  - std::find_if …

Summary…

- Functional Programming is
  - Just another programming **paradigm**
    - A new way to organize & express your thoughts
  - Non-specific tool
    - Depends on ***NOTHING*** more than your language
      - even in C / Assembly
  - Handy, efficient.

Reference

- Wiki: *Functional Programming*
- Wiki: *Programming Paradigms*
- Wiki: *Lambda Calculus*
- *Elements of Programming*: written by father of STL, thinking programs in mathematical way.
- *SICP*: introductive textbook of MIT CS Major, using **Scheme** to explain the art of programming
- *The Well-Grounded Java Developer*: Practical FP on JVM using Scala / Groovy and Clojure.

Questions?

Thank you!

by @KimmyLeo <kenpusney@outlook.com>