Introduction to Gura Programming Language

www.gura-lang.org

LL Diver on Aug 23, 2014

Author

Name	Yutaka Saito
Experience	Embedded Firmware to GUI
Favorite Lang	C++, Gura
Job History	Electric-app maker, chip maker, venture
Current Job	Free, unemployed rather **devoted to Gura**

Agenda

What's Gura?

Basic Specification

Iterator Operation

Extension Module

What's Gura?

Repeat processes appear in programs so often.

```
for (i = 0; i < 10; i++) {
}
```

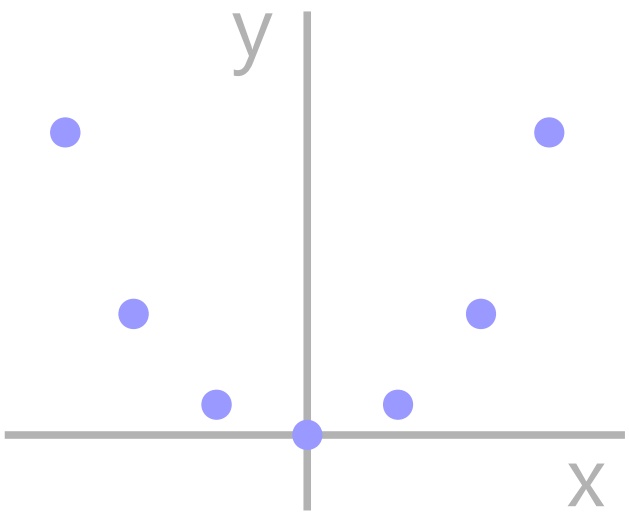
```
people.each do |per
end
```

```
for x in range(10):
    hoge
```

Can I deal them without verbose control sequence?

Case Study (1)

Here's a number sequence: -3, -2, -1, 0, 1, 2, 3.
Make a list of squared values of them.



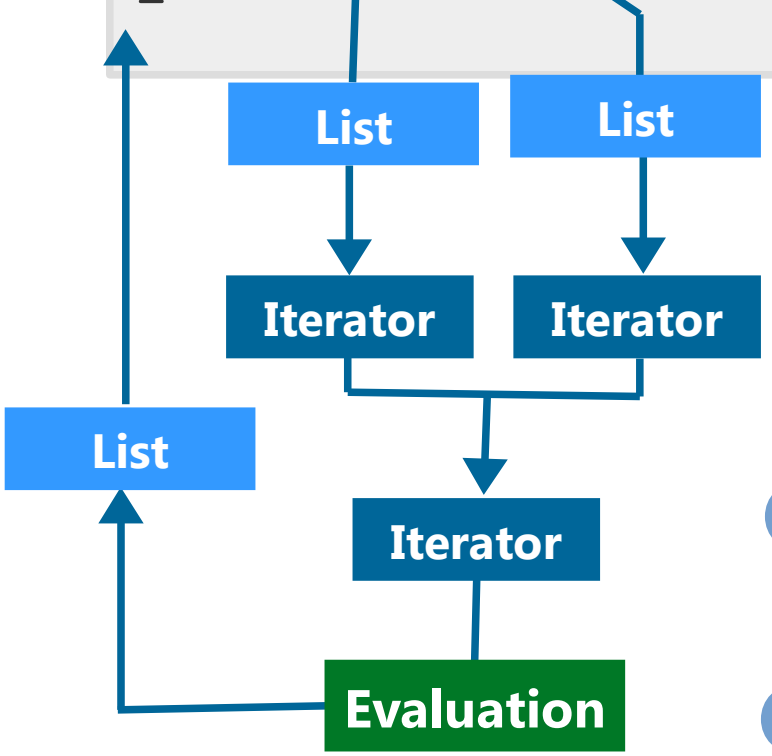
Think it with your favorite language ...

Gura's Program (1)

```
x = [-3, -2, -1, 0, 1, 2, 3]
y = x * x
```

Gura's Program (1)

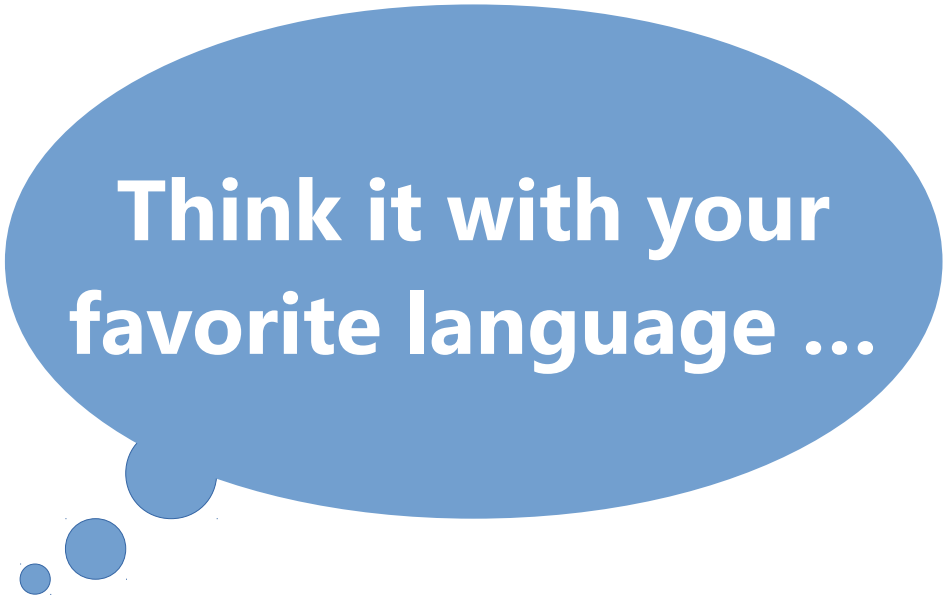
```
x = [-3, -2, -1, 0, 1, 2, 3]
y = x * x
```



1. Lists generate iterators.
2. Multiplying iterator is generated.
3. Its evaluation creates list.

Case Study (2)

Create a program that reads a text file and prints its content along with line numbers.



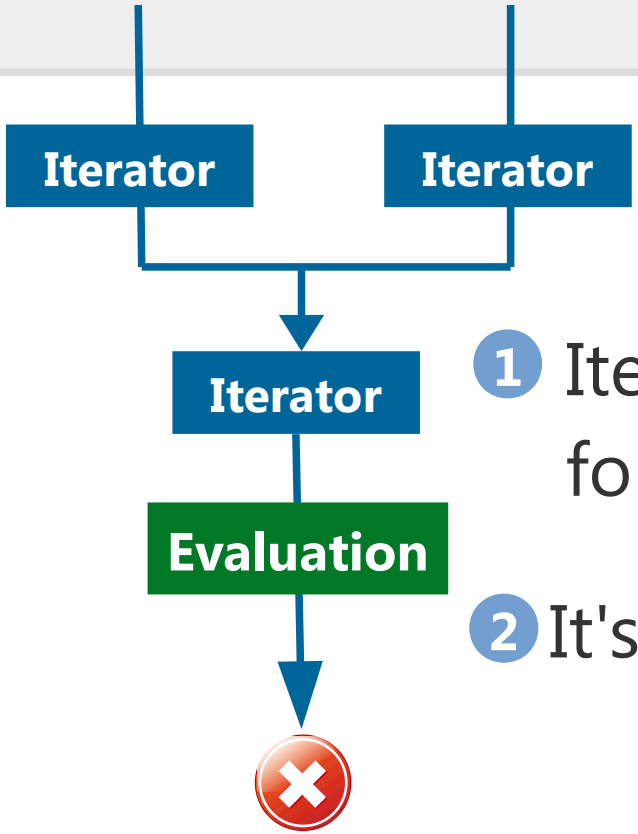

```
1: #include <std
2: int main()
3: {
4:     printf("H
5: }
```

Gura's Program (2)

```
printf('%d: %s',
       1.., readlines('hello.c'))
```

Gura's Program (2)

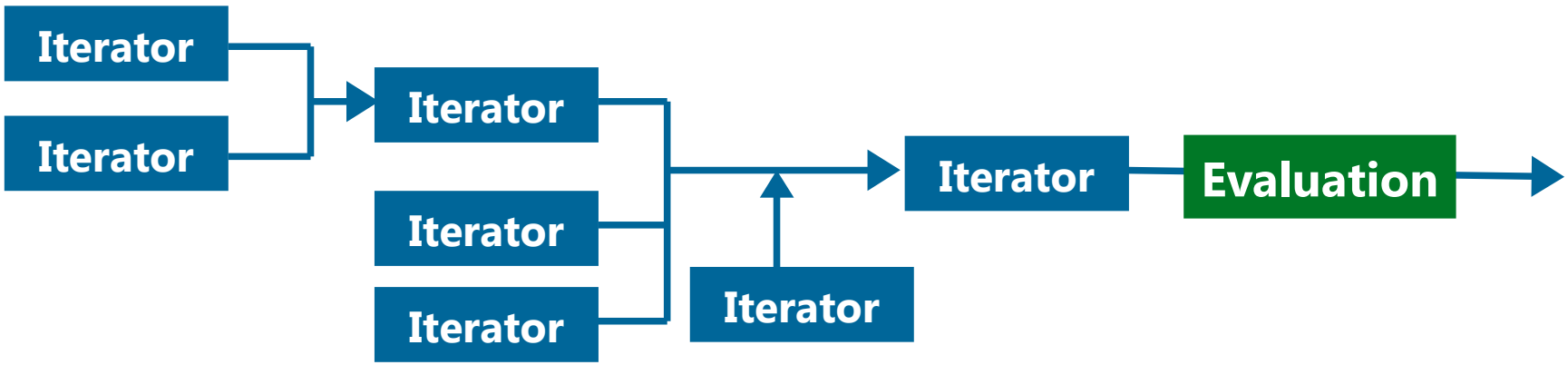
```
printf('%d: %s',
       1.., readlines('hello.c'))
```



1. Iterator that executes `printf` for each item is generated.
2. It's destroyed after evaluated.

After All, Gura is ..

A language that can generate iterators from iterators and evaluate them.

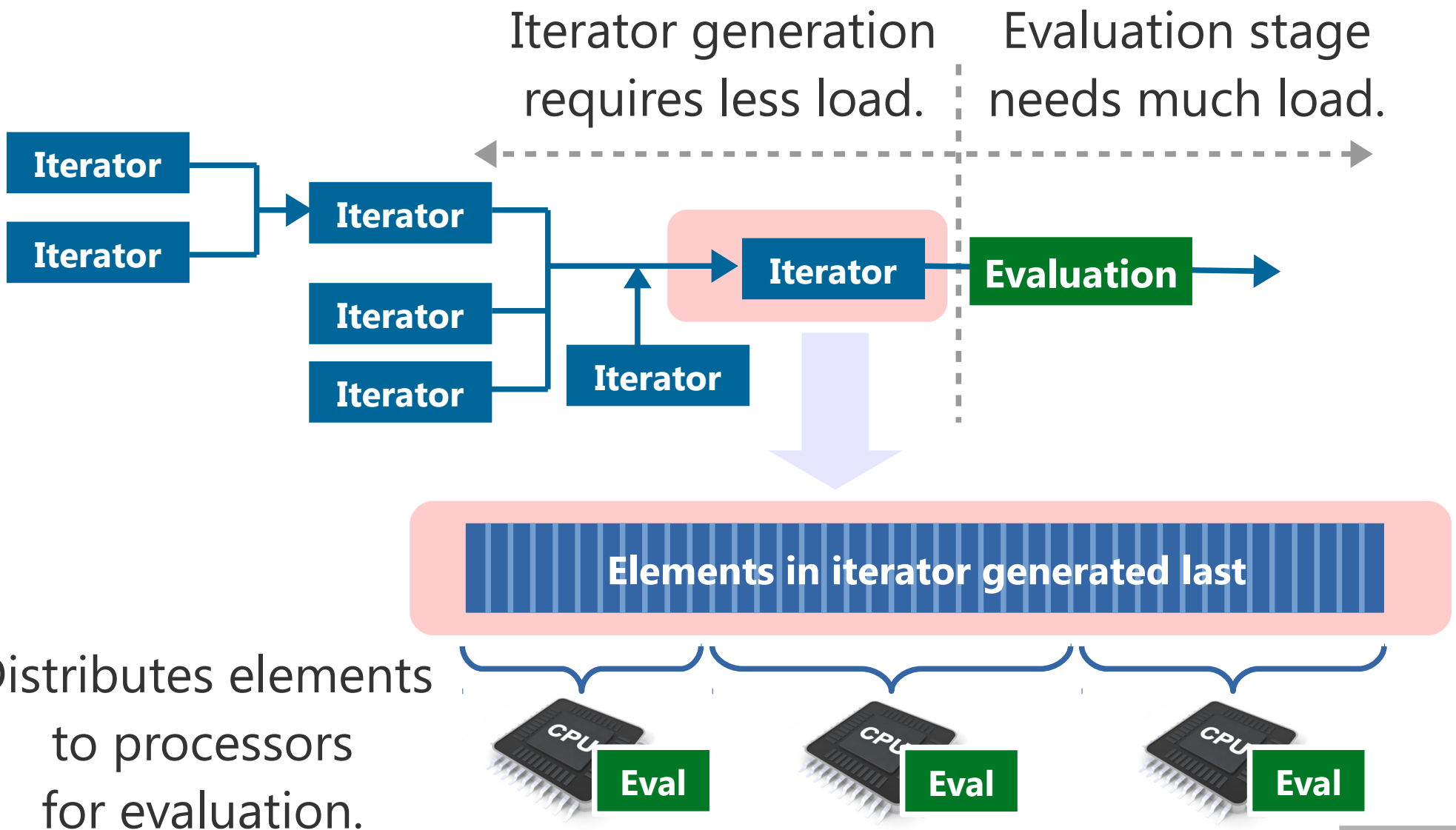


Gura calls this operation “Mapping”.

Expected Benefits

1. Simplifies codes of repeat process.
2. Facilitates parallel computing, maybe.

Idea for Parallel Computing



Distributes elements to processors for evaluation.

Agenda

- What's Gura?
- **Basic Specification**
- Iterator Operation
- Extension Module

Basic Specification

- Function
- Control Sequence
- OOP
- Collection
- Scope Management

Basic Spec (1) Function

Definition (1)

```
f(a:number, b:number) = {
    a * a + b * b
}
```

Type can be specified

Call

```
x = f(3, 4)
```

```
x = f(a => 3, b => 4)
```

Named arguments

Basic Spec(1) Function

Definition (2)

```
f(a, b*) = {
    // any job
}
```

Variable-length argument that takes more than 0 value.
`b+` takes more than 1 value.

Call

```
f(3)           // a=3, b=[]
f(3, 1)        // a=3, b=[1]
f(3, 1, 4, 1) // a=3, b=[1,4,1]
```

Basic Spec(1) Function

Definition (3)

```
my_loop(n) {block} = {
    while (n > 0) {
        block()
        n -= 1
    }
}
```

Takes block expression as function object.
`{block?}` means an optional block .

Call

```
my_loop(3) {
    println('hello')
}
```

Basic Spec(2) Control Sequence

Repeat

```
for (…) {
}
```

```
repeat (…) {
}
```

```
while (…) {
}
```

```
cross (…) {
}
```

Branch

```
if (…) {
} elsif (…) {
} elsif (…) {
} else {
}
```

Exception

```
try {
} catch (…) {
} catch (…) {
}
```

Basic Spec(3) OOP

Class Definition

Constructor

```
Fruit = class {
      __init__(name:string, price:number) = {
            this.name = name
            this.price = price
      }
      Print() = {
          printf('%s %d\n', this.name, this.price)
      }
}
```

Instantiation and Method Call

```
fruit = Fruit('Orange', 90)
fruit.Print()
```

Basic Spec(3) OOP

Inheritance

```
A = class {
      __init__(x, y) = {
          // any jobs
      }
}

B = class(A) {
      __init__(x, y, z) = {|x, y|
          // any jobs
      }
}
```

Arguments for base class constructor

Basic Spec(4) Collection

List

```
a = [3, 1, 4, 1, 5, 9]
b = ['zero', 'one', 2, 3, 'four', 5]
```

Dictionary

```
c = %{ `a => 3, `b => 1, `c => 4 }
d = %{
    'いぬ' => 'dog', 'ねこ' => 'cat'
}
```

Basic Spec(5) Scope Management

Each function has lexical scope.

Closure

```
create_counter(n:number) = {
    function {
        n -= 1
    }
}
```

```
c = create_counter(4)
c()  // returns 3
c()  // returns 2
c()  // returns 1
```

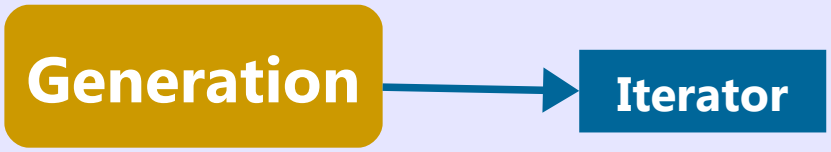
Agenda

- What's Gura?
- Basic Specification
- **Iterator Operation**
- Extension Module

Iterator Operation

Iterator Operation: **Mapping** and **Generation**

- Implicit Mapping
- Member Mapping



- Function
- Repeat Control

Gura's List and Iterator

List All the elements are stored in memory.

```
['apple', 'orange', 'grape']
```

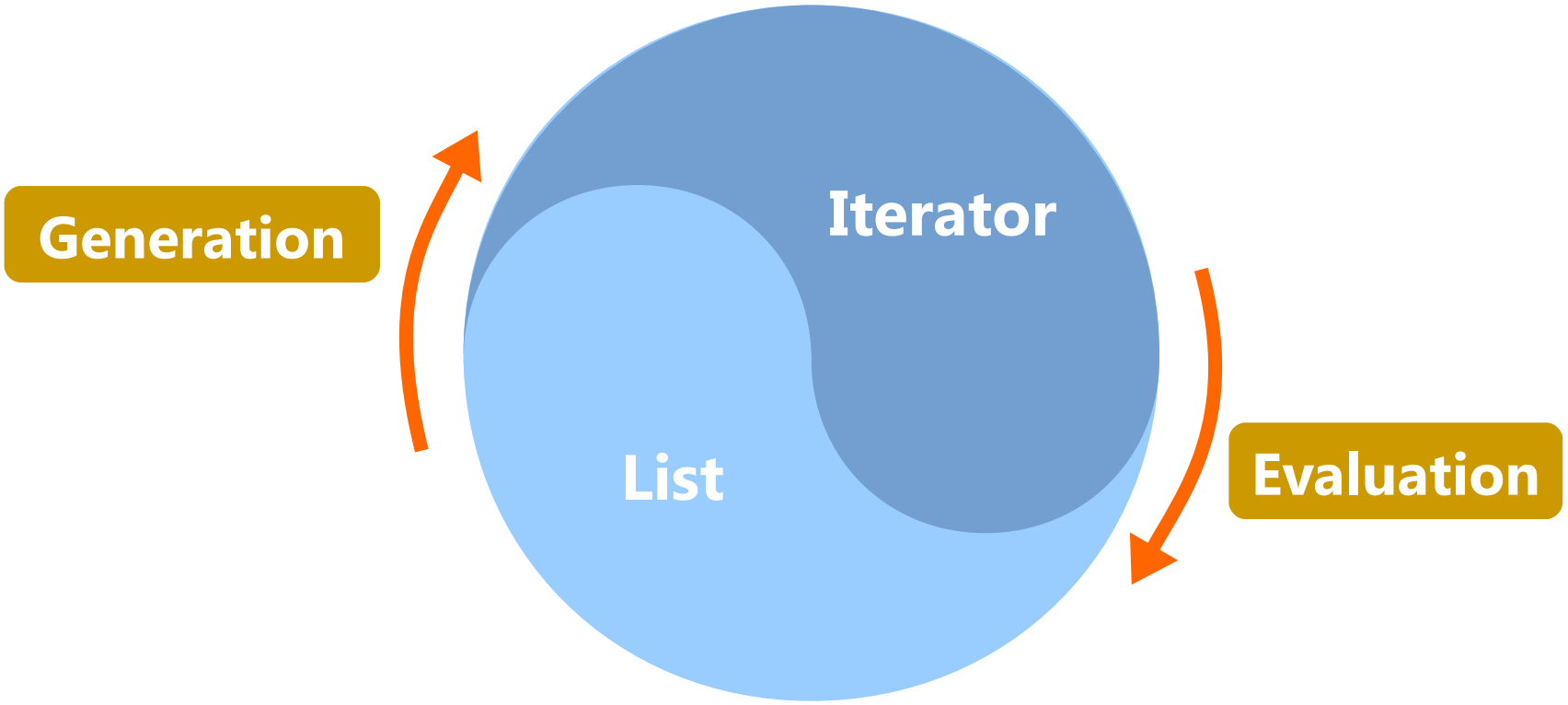
Capable of random access

Iterator Each element would be generated.

```
('apple', 'orange', 'grape')
```

Only evaluation could make next element available

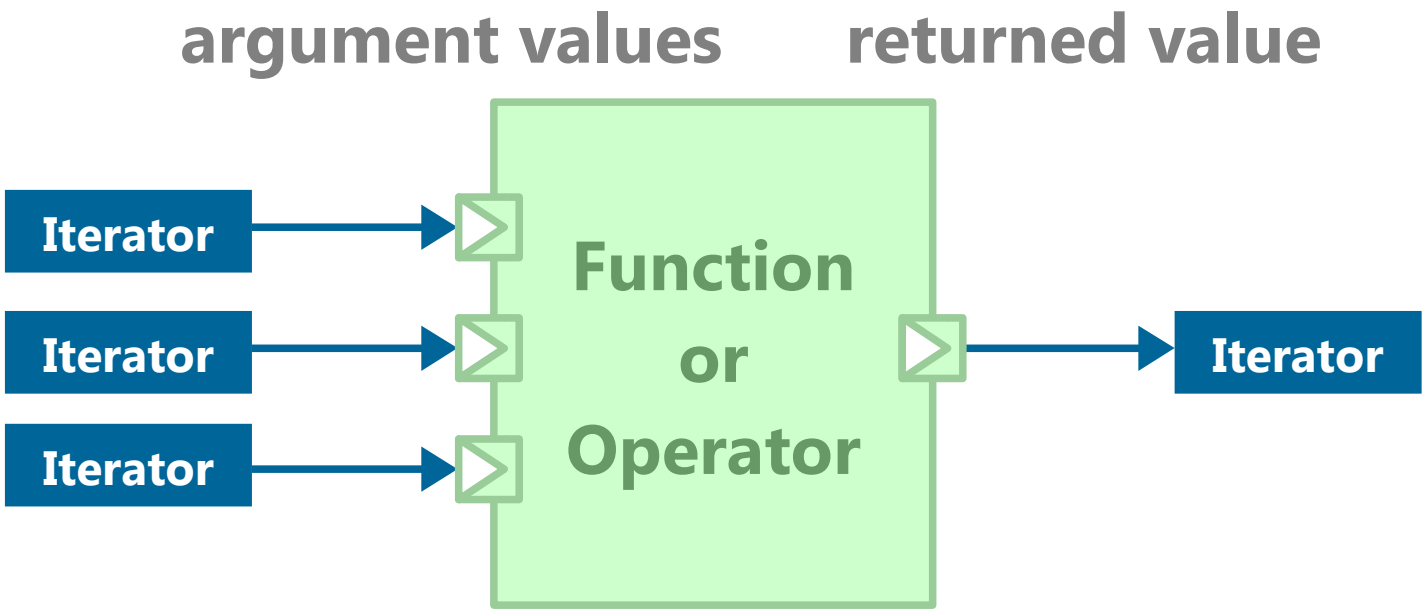
Gura's List and Iterator



Iterator Operation(1) Implicit Mapping

Implicit Mapping

Generates iterator embedding function or operator



Iterator Operation(1) Implicit Mapping

Usual Function

```
f(a:number, b:number) = {
    a * b
}
```

Mappable Function

Specifies attribute `:map`

```
f(a:number, b:number):map = {
    a * b
}
```

Iterator Operation(1) Implicit Mapping

Number

```
f(3, 4)
```

Answer: `12`

List

```
f([2, 3, 4], [3, 4, 5])
```

Answer: `[6, 12, 20]`

Iterator

```
f((2, 3, 4), (3, 4, 5))
```

Answer: `(6, 12, 20)`

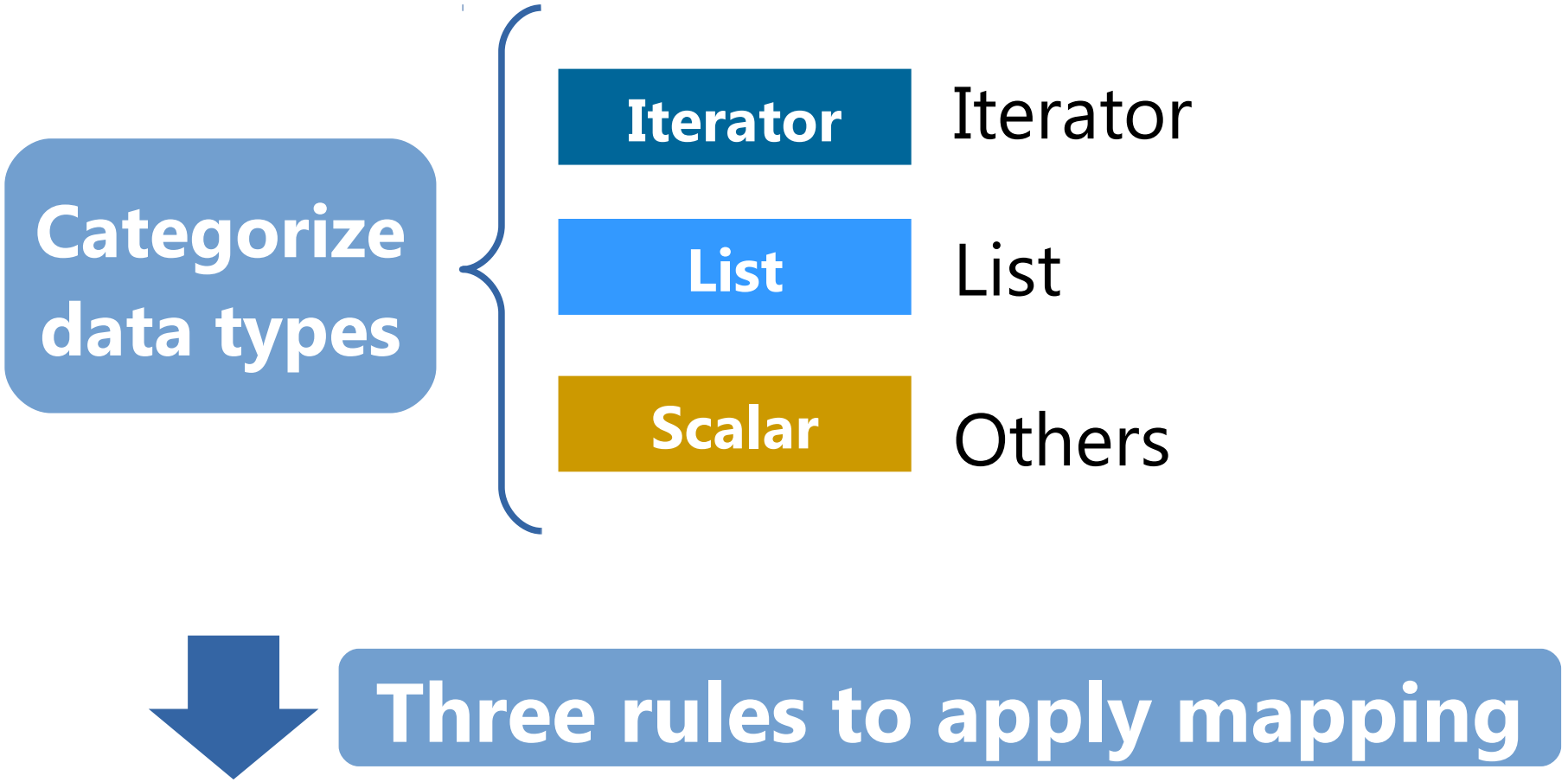
Number and Iterator

```
f(5, (3, 4, 5))
```

Answer: `(15, 20, 25)`

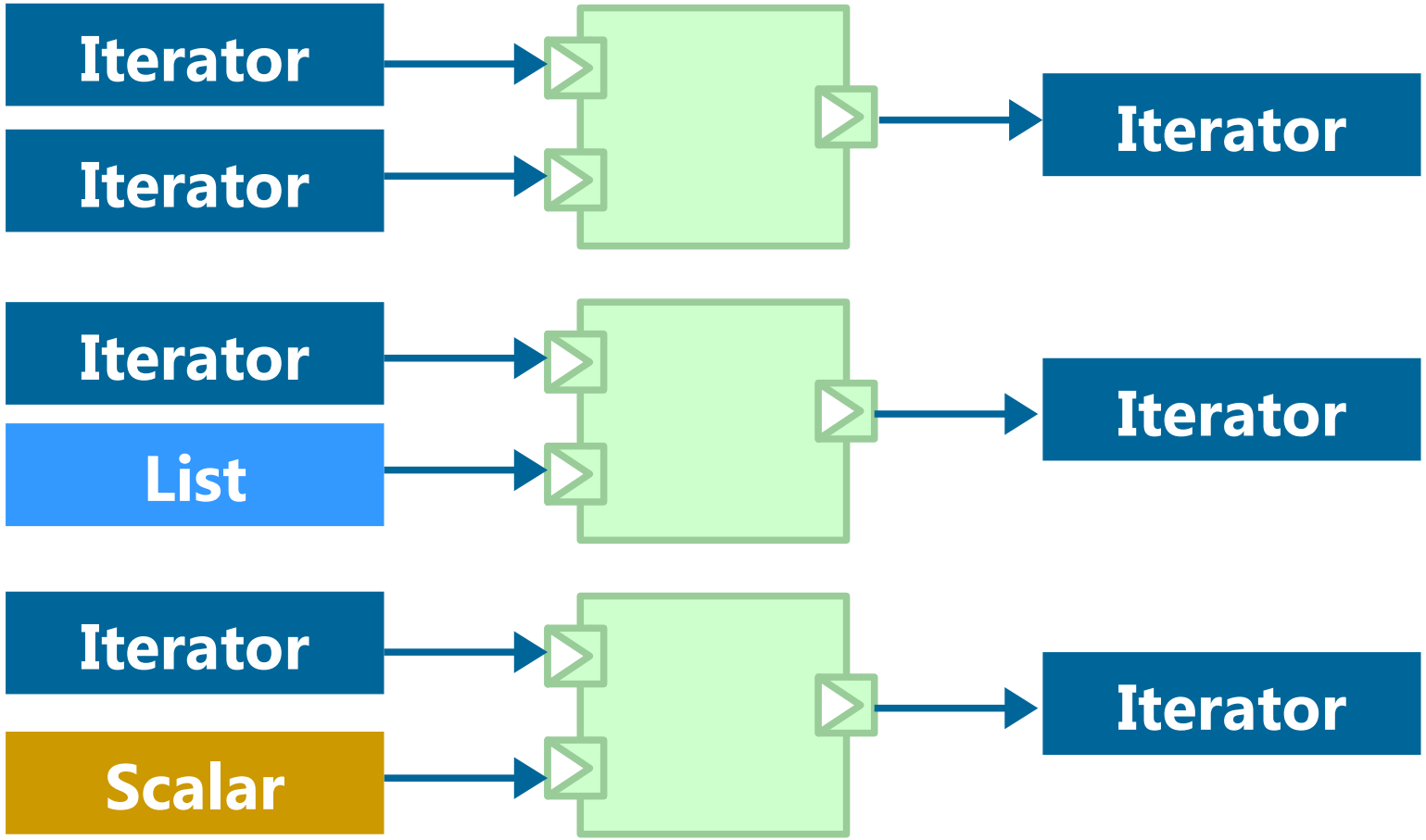
Iterator Operation(1) Implicit Mapping

Mapping process depends arguments' data type



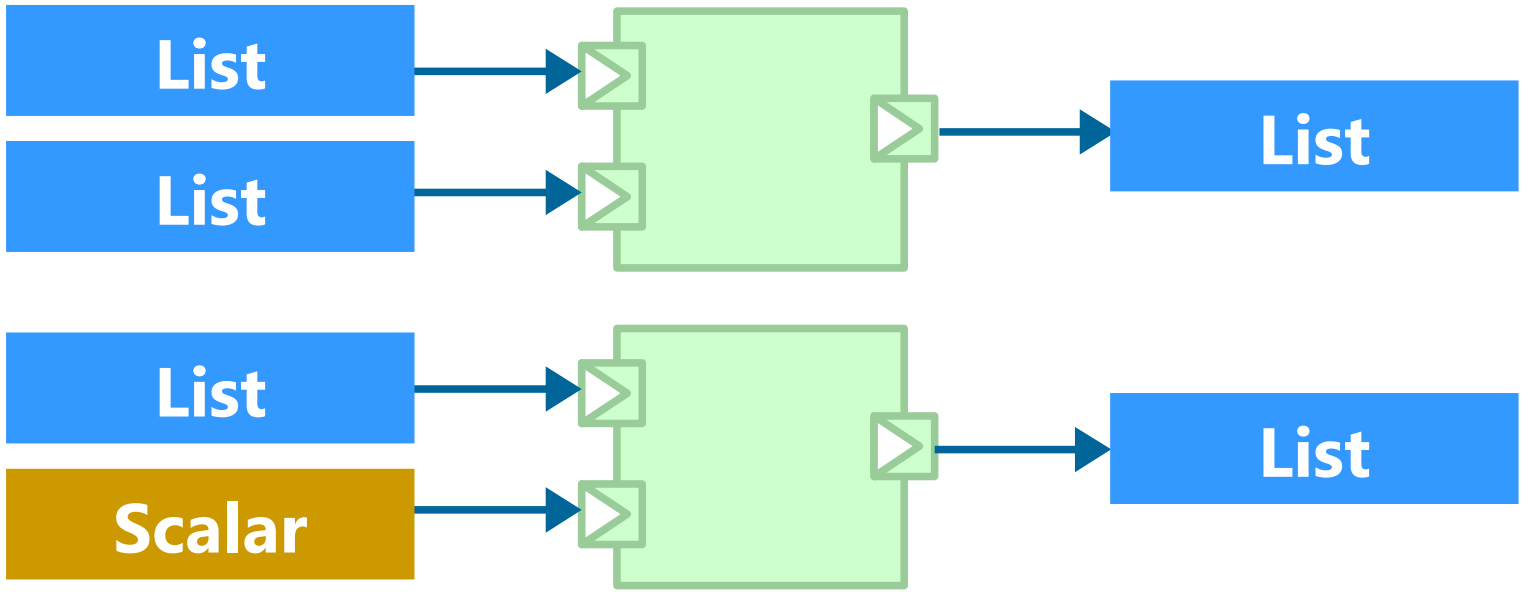
Iterator Operation(1) Implicit Mapping

Rule 1 Generates an iterator
if at least one iterator exists in arguments.



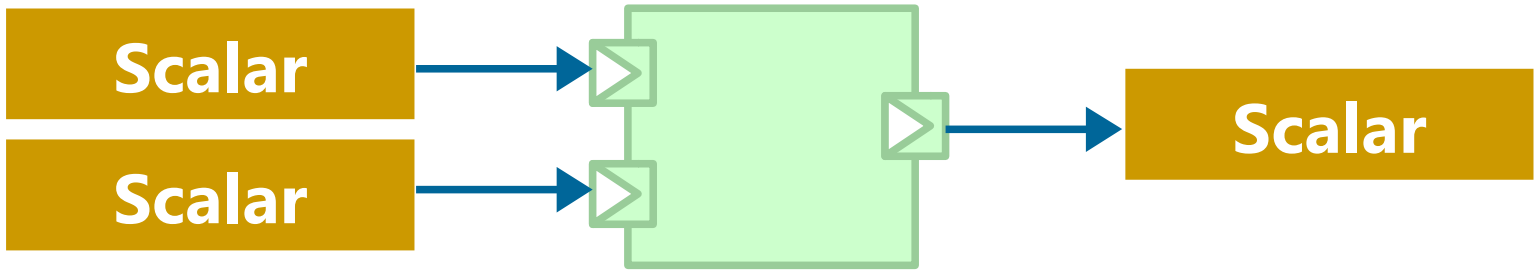
Iterator Operation(1) Implicit Mapping

Rule 2 Generates a list if there's no iterator and at least one list exists in arguments.



Iterator Operation(1) Implicit Mapping

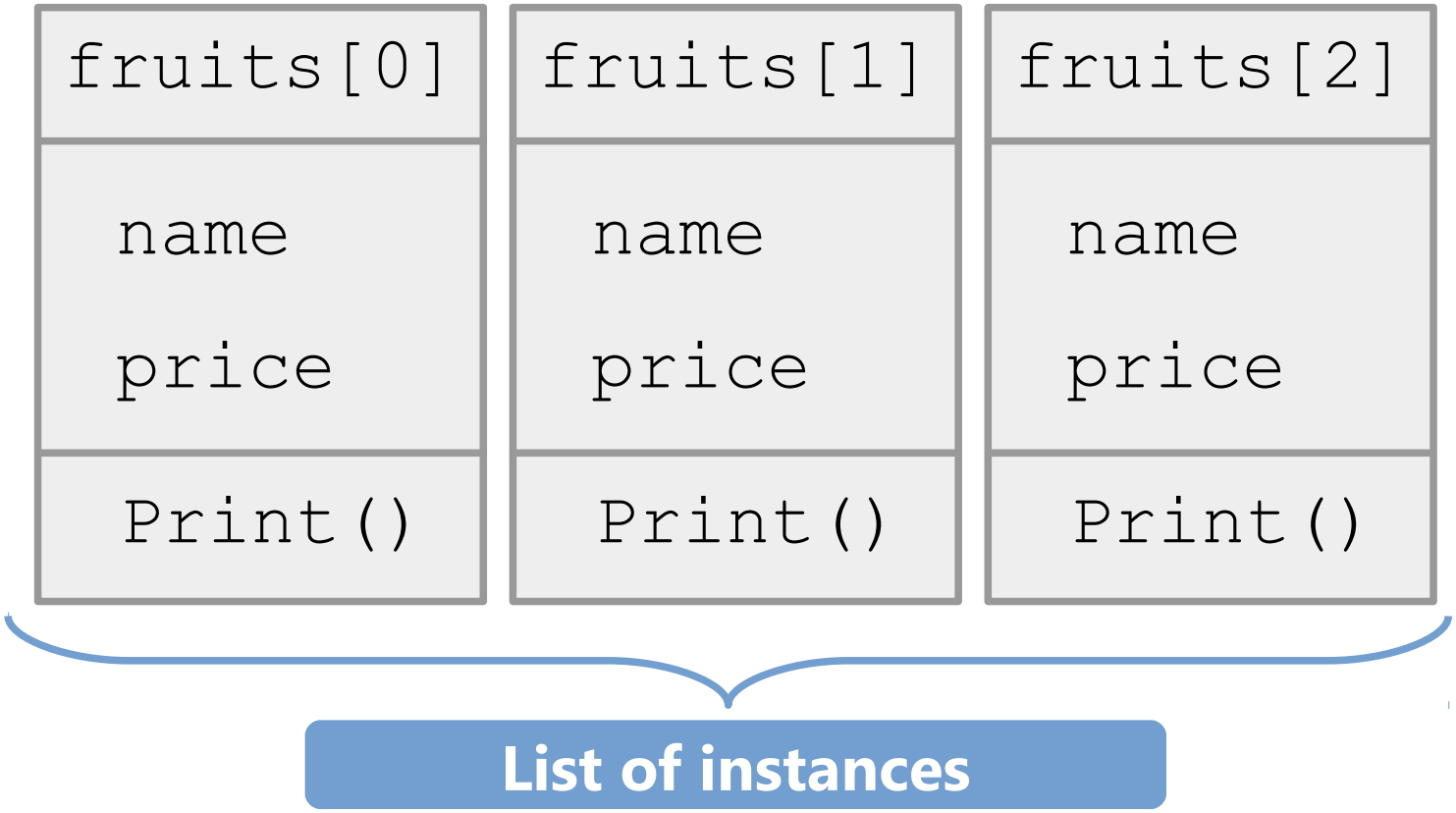
Rule 3 Generates a scalar
if only scalars exist in arguments.



Iterator Operation(2) Member Mapping

Member Mapping

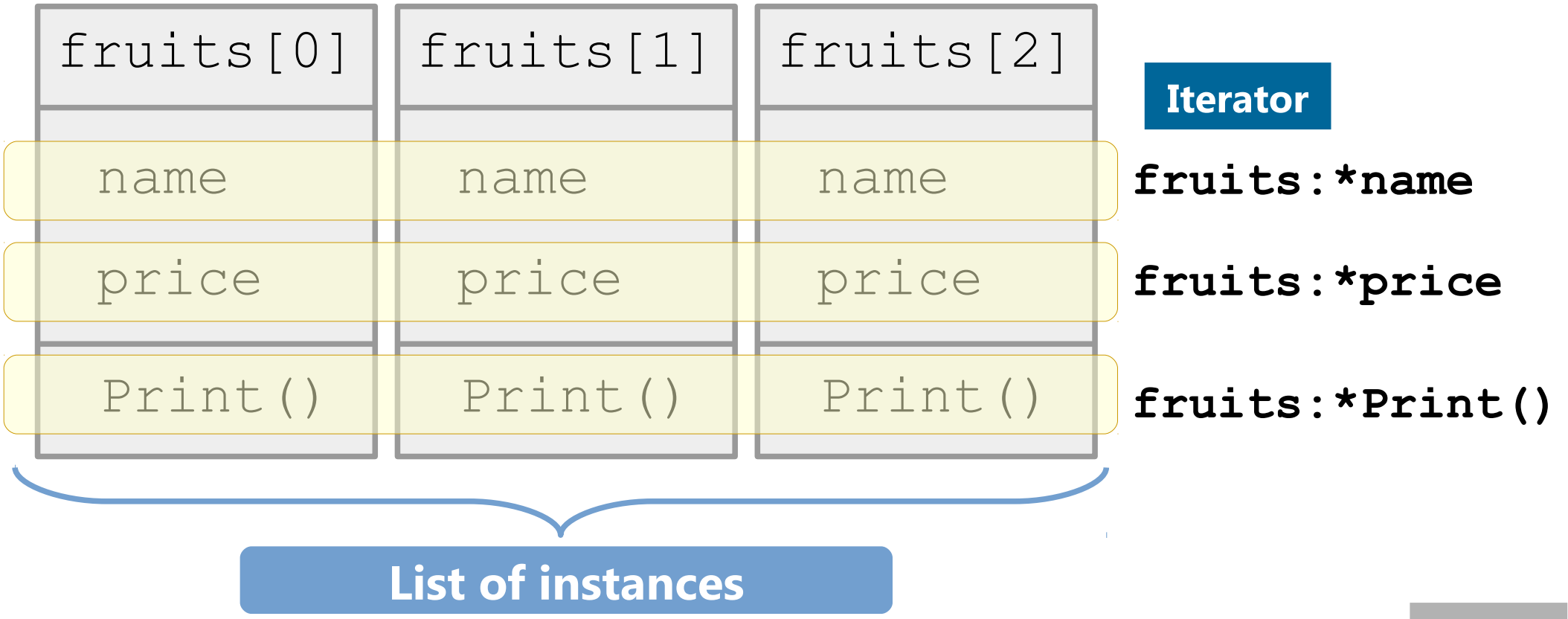
Generates an iterator to access instance members.



Iterator Operation(2) Member Mapping

Member Mapping

Generates an iterator to access instance members.



Iterator Operation(2) Member Mapping

Task Prints summation of `Fruit` instance's member `price`.

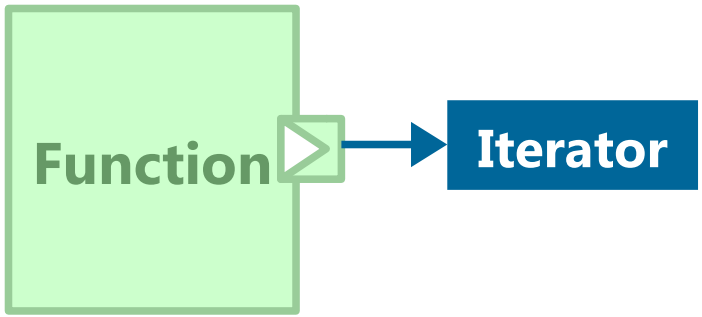
Solution1 Using repeat control

```
sum = 0
for (fruit in fruits) {
    sum += fruit.price
}
println(sum)
```

Solution2 Using Member Mapping

```
println(fruits:*price.sum())
```

Iterator Operation(3) Function



Design Policy

A function should return data sequence by an iterator, not a list.

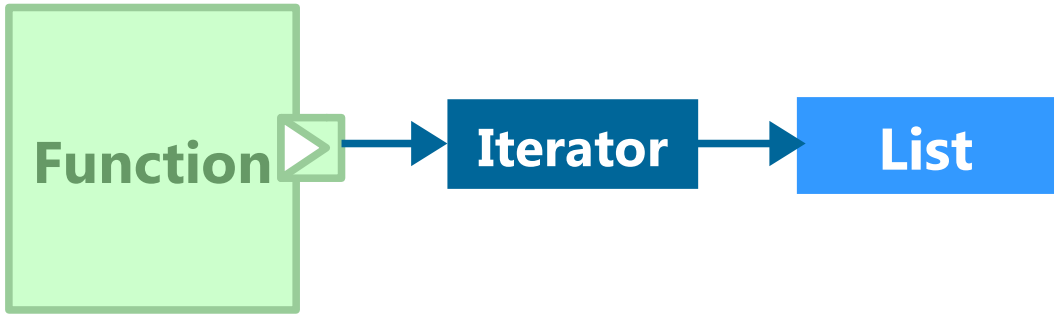
```
rtn = readlines('hello.c')
```

Iterator to generate strings of each line

```
rtn = range(10)
```

Iterator to generate numbers from 0 to 9

Iterator Operation(3) Function



Call with attribute `:list` in order to get a list

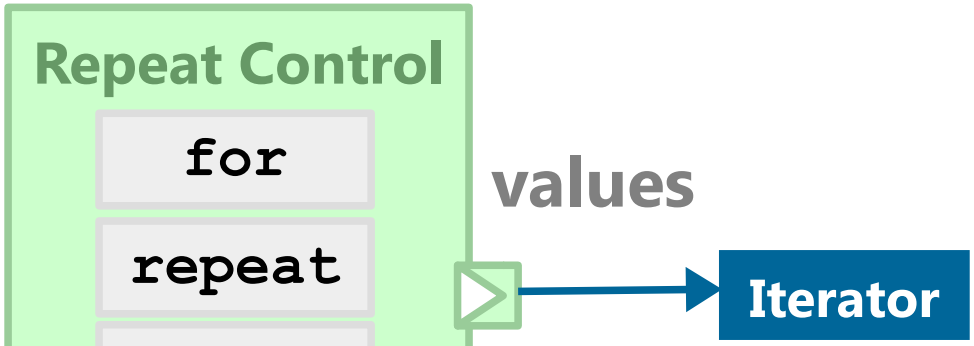
```
rtn = readlines('hello.c'):list
```

List containing strings of each line

```
rtn = range(10):list
```

List containing numbers from 0 to 9

Iterator Operation(4) Repeat Control



Generate an iterator from repeat control

Specify attribute `:iter`

```
x = for (...):iter {
    repeating job
}
```

Evaluation result of repeating job comes to be the iterator's element

Iterator Operation(4) Repeat Control

Example of repeat control iterator

```
n = 0
x = for (i in 0..5):iter {
    n += i
}
```

Nothing happens at this time

```
println(x)
```

Prints result: `0 1 3 6 10 15`

Iterator Operation(4) Repeat Control

Iterator to generate prime numbers

```
prime() = {
    p = []
    for (n in 2..):xiter {
        if (!(n % p.each() == 0).or()) {
            p.add(n)
            n
        }
    }
}
```

```
primes = prime()
```

Iterator to generate numbers (2, 3, 5, 7..)

Agenda

What's Gura?

Basic Specification

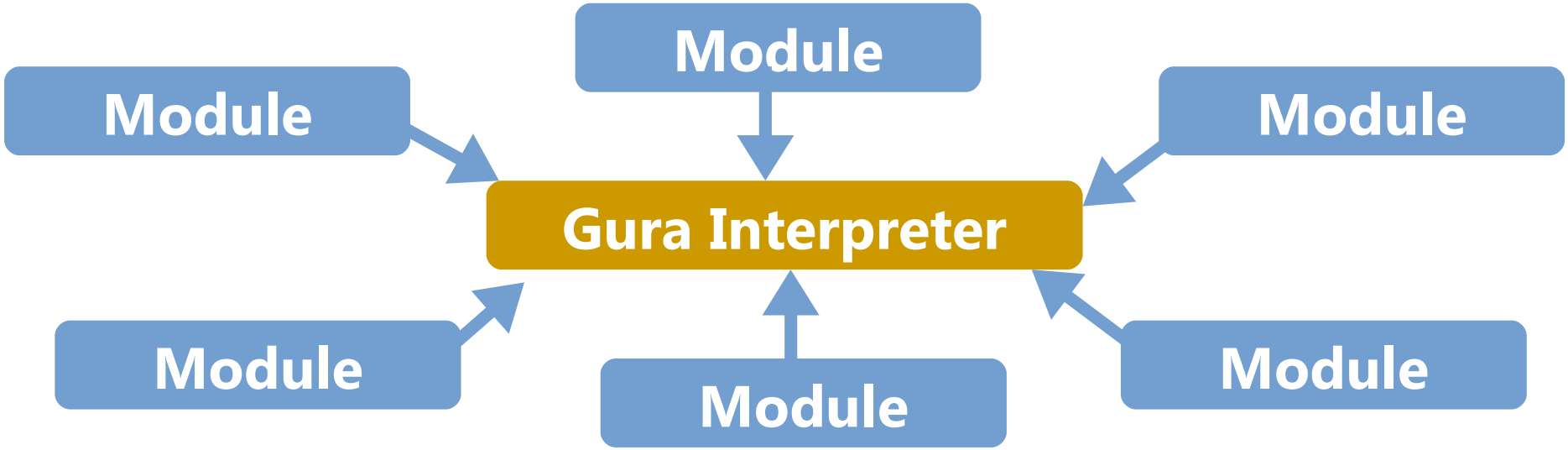
Iterator Operation

Extension Module

Extension Module

Design Policy

- Gura Interpreter itself should have as little dependency on OS-specific functions and libraries as possible.
- It should extend capabilities by importing modules.



Bundled Modules

GUI

wxWidgets | Tk | SDL

Graphic Drawing

Cairo | OpenGL | FreeType

Image Data

JPEG | PNG | GIF | BMP

ICO | XPM | PPM | TIFF

Text Processing

CSV | XML | yaml

Regular Exp | markdown

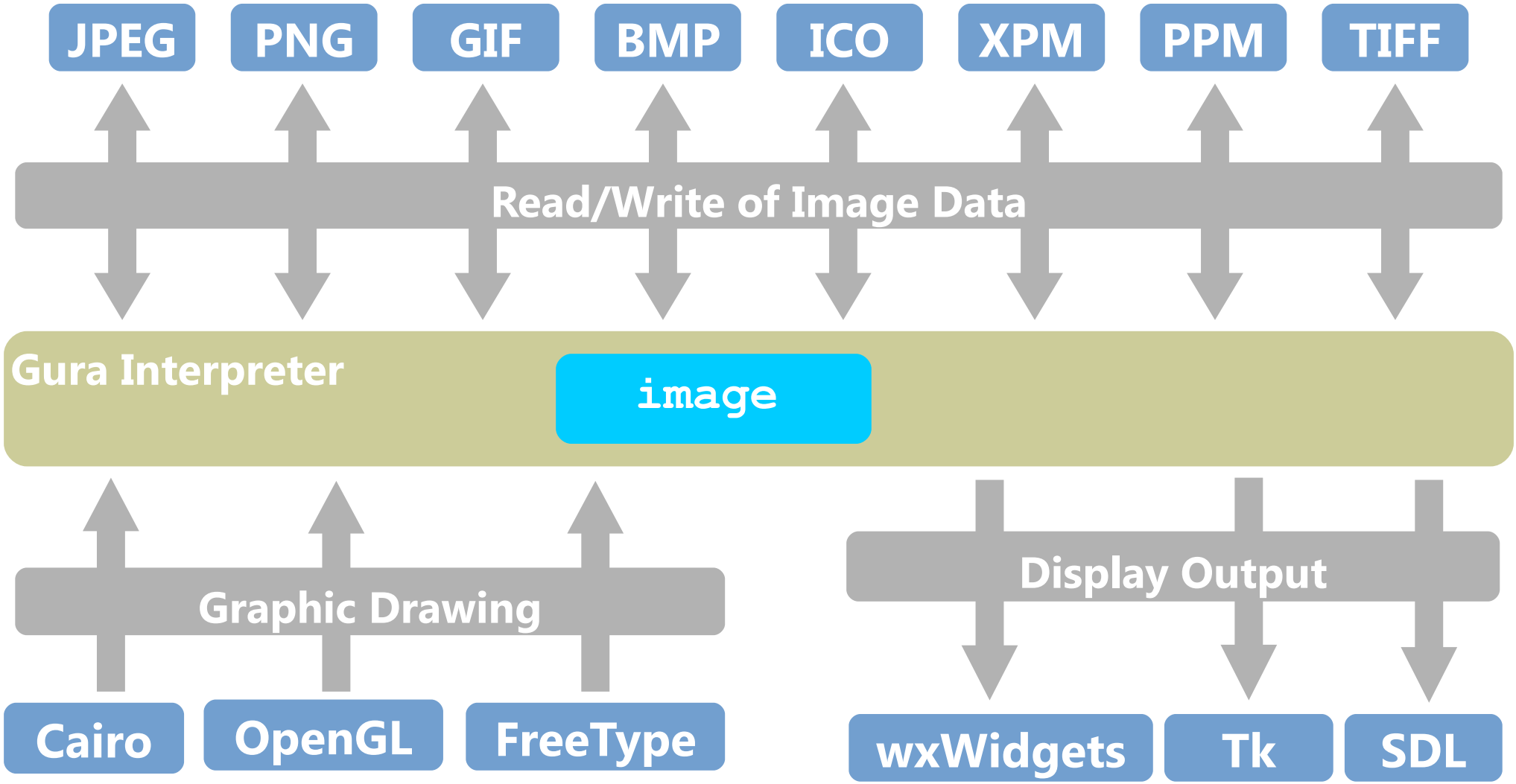
Archive and Compression

TAR | ZIP | GZIP | BZIP

Network

cURL | Server

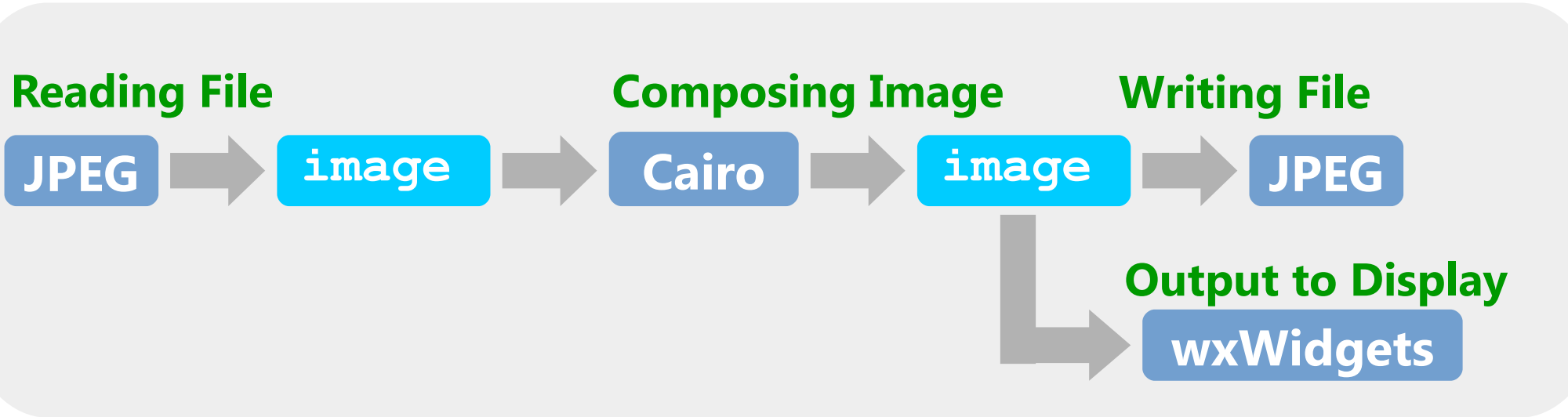
Cooperation between Modules



Application Example

[ID Photo Made at Home - Gura Shot]

- Creates ID photos by extracted face image from a file of digital camera.
- Outputs results in PDF and JPEG.



Thank you

www.gura-lang.org