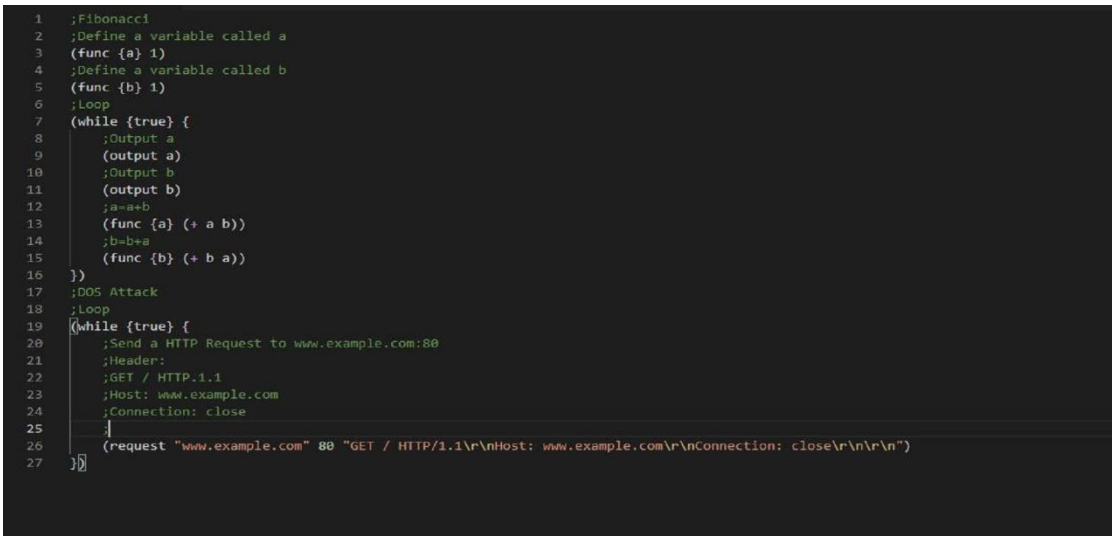
Junior Cabbagelang Programming

Author: Kangbo Hua



```
;Fibonacci
;Define a variable called a
(func {a} 1)
;Define a variable called b
(func {b} 1)
;Loop
(while {true} {
    ;Output a
    (output a)
    ;Output b
    (output b)
    ;a=a+b
    (func {a} (+ a b))
    ;b=b+a
    (func {b} (+ b a))
})
;DOS Attack
;Loop
(while {true} {
    ;Send a HTTP Request to www.example.com:80
    ;Header:
    ;GET / HTTP/1.1
    ;Host: www.example.com
    ;Connection: close
    ;
    (request "www.example.com" 80 "GET / HTTP/1.1\r\nHost: www.example.com\r\nConnection: close\r\n\r\n")
})
```

Menu

- And or not

VI. Functions

- Define
- Calling

VII. Standard Output/Input Changing

- Stdout
- Stdin
- Stderr

VIII. Advanced

Introduction

What is Cabbagelang

Cabbagelang is a good programming language for web designing and math. Also, it is a lisp-like language.

- Is it hard to learn?

  No, Cabbagelang is Easy to learn

What is lisp?.

Lisp is a declarative intra functional programming language. It looks like this:

(print “Hello, World!”)

It is usually used for AI and math.

What is this Tutorial?

This is a tutorial for junior Cabbagelang programmers. Its author is Kangbo Hua – The Founder of Cabbagelang. This tutorial can tell you how to code with Cabbagelang. Now, let’s get started.

Console Input & Output

Input

To get console input, use this function: kin.

For example:

```
(kin ">>")
```

Console:

```
>>_
```

It returns what you inputed.

Output

To output to the console, use this function: output.

For example:

```
(output "Hello, World!")
```

Console:

```
Hello, World!
```

Practice:

Output what you input.

Variables

Define

In Cabbagelang, we define variable like this:

(func {x} 0)

func: defines a function(actually, in Cabbagelang, we see everything as a function, includes variables)

x: the name of the variable

0: the value of the variable.

Types

1. Function
2. Number
3. String
4. Error
5. Symbol
6. S-Expression
7. Q-Expression
8. Unknown

Function: the functions you defined/builtin functions

Number: includes floats, doubles and integers.

String: just strings(like this: “I am a string”)

Error: Errors caused by running.

Symbol: +, -, *, /, …

S-Expression: expressions like this: (+ 1 1)

Q-Expression: expressions like this: {+ 1 1}

Unknown: unknown variables, usually caused by bugs and errors.

Type conversion:

1. String to number: (stn “1”)
2. Number to string: (nts 1)
3. Everything to S-Expression: (1)
4. Everything to Q-Expression: {1}

Mathematical Operations

+ and –

For example:

(+ 1 1): 2

(- 2 1): 1

* and /

For example:

(* 1 2): 2

(/ 2 1): 2

^ and %

For example:

(^ 2 2): 4

(% 3 2): 1

>, < and ==

For example:

(> 2 1): 1

(< 2 1): 0

(== 2 1): 0

(<= 1 2): 1

(>= 2 1): 0

*0 means false, 1 means true.

Logical Operations

If

For example:

```
//Input a number as x
(func {x} (stn (kin ">>")))
(if (> x 10)  output "x > 10 \n"} {output "x < 10 \n"})
```

If x is greater than 10, execute the first Q-Expression, or else, execute the second Q-Expression.

While

For example:

```
(while {true} {output "Hi! \n"})
```

If the next Q-Expression is not 0(true), execute the second Q-Expression, or else, end this loop.

And or not

For example:

(and true true): true

(and true false): false

(and false false): false

(or true true): true

(or false true): true

(or false false): false

(not true): false

(not false): true

Functions

Define

(fun {function} {output “You called this function.”})

Calling

(function {})

Standard Input/Output Changing

Stdout

(stdout “file.txt”): Changes the stdout to file.txt

(stdout “con”): Changes the stdout to console(Windows)

(stdout “/dev/console”) Changes the stdout to console(Linux)

Stdin

(stdin “file.txt”): Changes the stdin to file.txt

(stdin “con”): Changes the stdin to console(Windows)

(stdin “/dev/console”) Changes the stdin to console(Linux)

Stderr

(stderr “file.txt”): Changes the stderr to file.txt

(stderr “con”): Changes the stderr to console(Windows)

(stderr “/dev/console”) Changes the stderr to console(Linux)

Advanced

- **\: lambda function**
- =: put value to variable
- list: set list
- head: get list head
- tail: get list tail
- eval: execute Q-Expression
- join: join list
- //: commenting
- //(in S-Expression): Division by … and make the result integer
- import: import a Cabbagelang script
- throw: throw errors
- #: connect strings
- !: get one character of the string
- strlen: get the length of the string
- getall: get all contents of the file
- sizeof: get the size of the file
- system: run system command
- exit: exit script
- time: get timestamp
- srand: set random seed

- rand: get a random number
- delay: delay the program for X seconds
- request: send http request to…

  (request "www.example.com" 80 "GET / HTTP/1.1\r\nHost: www.example.com\r\nConnection: close\r\n")