Say Hi ! Golang

By : Kevin Setiawan Tanzil

Source : "Code Ref" by : GO (Kis), PHP (Mawaddi & Asep)

This is the simple handbook for learning Golang from PHP, JS and PYTHON user

Official Website
or most popular site if there's none

PHP	JS	Python 3	GO
php.net	developer.mozilla.org/en-US/docs/Web/JavaScript	python.org	golang.org

Key Person (Designer)

PHP	JS	Python 3	GO
Rasmus Lerdorf	Brendan Eich	Guido van Rossum	Robert Griesemer Rob Pike Ken Thompson

Backer Company

PHP	JS	Python 3	GO
most of Zend Engine under Zend Engine License & The TSRM License	Mozilla Foundation	Python Software Foundation, and Community-backed. Used strongly in Google.	Google

Playground/Fiddle/REPL

PHP	JS	Python 3	GO
phpfiddle.org	Javascript Console on browsers	pythonfiddle.com	play.golang.org

Compiler/Interpreter Implementations (if too many >10, choose the popular ones)

PHP	JS	Python 3	GO
	v8 (C++, Google, used in nodejs) jscore (C++) mozjs/spidermonkey (C/C++, Mozilla) rhino (Java, Mozilla) Nashorn (Java, Oracle) ChakraCore (Microsoft)	CPython (C) IronPython (.NET) Jython (Java) PyPy (RPython)	gc (C, replaced with Go) gccgo (C++) llgo (C++)

First Appeared (Year)

PHP	JS	Python 3	GO
1995	1995	1991	2009

Age for this year

PHP	JS	Python 3	GO
24	24	28	10

Major Implementation installation

PHP	JS	Python 3	GO
Windows: easy install for using wamp Mac: easy install for using mamp Linux: depends on the distribution, for example: apt-get install php			windows/mac: download, install linux: depends on the distribution, for example: apt-get install go # debian/ubuntu-like pacman -S go # archlinux/manjaro # don't forget to set GOPATH environment variable

Compile Syntax on Major Implementation

PHP	JS	Python 3	GO
			go build test.go go install test.go# also copy to bin directory

Run Syntax on Major Implementation (add compile syntax && if it must be compiled first)

PHP	JS	Python 3	GO

CLI SAPI with PHP code to be executed:
`$ php hello.php`
 	
Using browser with Apache:
`http://localhost/hello.php`
	`node test.js`
	`js test.js`
	`jsc test.js`
	`rhino test.js`
		`python test.py`	`go run test.go`
			`go build test.go && ./test`

Famous IDE that has at least: autocomplete and jump-to-definition/find-references

PHP	JS	Python 3	GO
- PhpStorm (Jetbrains) - Eclispe - Zend	Vscode	• PyCharm • Eclipse + PyDev plugin	IntelliJ + go-lang-idea-plugin SublimeText + gosublime wide VSCode + vscode-go LiteIDE

Library Installation

PHP	JS	Python 3	GO

`composer install <package-name>`	`npm install <package-name>`		`go get`
			`Go mod`
			`dep`

IMPORT MODULE/LIBRARY

PHP	JS	Python 3	GO

`require('foo.php');`
`require_once('foo.php');`
		• `import json`
		• `import fibo1, fibo2`
		• `import time as time_module`
		• `import sound.effects.echo`
		
		• `from datetime import datetime`
		• `from fibo import *`
		• `from sound.effects import echo`
		
		• `from . import abc`
		• `from .. import xyz`
		• `from ..filters import equalizer`
			`import "strconv"`
			`import (`
			`"strings"`
			`"sort"`
			`)`

IF, CONDITIONAL EXPRESSION

PHP	JS	Python 3	GO

<?php
 	
if ($i > 0) {
$num = 1;
} else if ($i == 0) {
$num = 0;
} else {
$num = -1;
}
?>
	if (i > 0) {
	num = 1
	} else if (i == 0) {
	num = 0
	} else {
	num = -1
	}
		if i > 0:
		num = 1
		elif i == 0:
		num = 0
		else:
		num = -1
			if i > 0 { // error if you move the curly brace below the if
			num = 1
			} else if i == 0 {
			num = 0
			} else {
			num = -1
			}

FOR-LOOP, EXECUTION CONTROL

PHP	JS	Python 3	GO

`<?php`
`$n = 0;`
`for($i=2;$i<=10;++$i) {`
`$n += $i;`
`} ?>`
	`var n = 0`
	`for(var i=2; i<=10;`
	`++i) {`
	`n += i`
	`}`
	`// use let instead of`
	`var for ES6+`
		`n = 0`
		`for i in range(2, 11):`
		`n += i`
			`n := 0`
			`for i:=2; i <= 10; i++`
			`{`
			`n += i`
			`}`

SWITCH-STATEMENT, EXECUTION CONTROL

PHP	JS	Python 3	GO
`<?php`	`let grade:string = "A";`	`grade = 'A'`	``grade := `A` ``
`$grade = "A";`	`switch(grade) {`		`switch grade {`
`switch($grade) {`	`case "A":`	`if grade == 'A':`	``case `A`:``
`case "A":`	`console.log("Excellent"`	`print('Excellent')`	``fmt.Println(`Excellent` ``
`echo "Excellent";`	`);`	`elif grade == 'B':`	`)`
`break;`	`break;`	`print('Good')`	``case `B`:``
`case "B":`	`case "B":`	`elif grade == 'C':`	``fmt.Println(`Good`)``
`echo "Good";`	`console.log("Good");`	`print('Fair')`	``case `C`:``
`break;`	`break;`	`elif grade == 'D':`	``fmt.Println(`Fair`)``
`case "C":`	`case "C":`	`print('Poor')`	``case `D`:``
`echo "Fair";`	`console.log("Fair");`	`elif grade == 'X':`	``fmt.Println(`Poor`)``
`break;`	`break;`	`print('X?')`	``case `X`:``
`case "D":`	`case "D":`	`elif grade == 'Z':`	``fmt.Println(`X?`)``
`echo "Poor";`	`console.log("Poor");`	`print('YZ?')`	`fallthrough`
`break;`	`break;`	`else:`	``case `Y`,`Z`:``
`case "X":`	`case "X":`	`print('Invalid choice')`	``fmt.Println(`YZ?`)``
`echo "X";`	`console.log("X?")`		`default:`
`case "Y":`	`// fallthrough here`		``fmt.Println(`Invalid``
`case "Z":`	`case "Y","Z":`		``Choice`)``
`echo "YZ";`	`console.log("YZ?")`		`}`
`break;`	`break;`		
`default:`	`default:`		
`echo "Invalid choice";`	`console.log("Invalid`		

BOTTOM-TESTED LOOP, EXECUTION CONTROL

PHP	JS	Python 3	GO

`<?php`
`$n = 10;`
`do {`
`echo $n . ' <br/>';`
`$n--;`
`} while ($n >= 0);`
`?>`
	`let n:number = 10;`
	`do {`
	`console.log(n);`
	`n--;`
	`} while(n>=0);`
		`n = 10`
		`while n >= 0:`
		`print(n)`
		`n -= 1`
			`n := 10`
			`for {`
			`fmt.Println(n)`
			`n -= 1`
			`if n<0 { break; }`
			`}`

TOP-TESTED LOOP, SKIP 3, BREAK IN THE MIDDLE

PHP	JS	Python 3	GO

<?php
$n = 1;
while($n<10) {
$n += 2;
if($n==3) continue;
if($n==7) break;
echo $n. '<br />';
}
?>
	let n:number = 1;
	while(n<10) {
	n += 2
	if(n==3) continue;
	if(n==7) break;
	console.log(n);
	}
		n = 1
		while n < 10:
		n += 2
		if n == 3:
		continue
		elif n == 7:
		break
		print(n)
			n := 1
			for n<10 {
			n += 2
			if n == 3 {
			continue
			}
			if n == 7 {
			break
			}
			fmt.Println(n)
			}

Operators Precedence (use double newline for next precedence)

PHP	JS	Python 3	GO

Same like most C based languages	Same like most C based languages	Same like most C based languages	`* / %`
			`<< >> // shift`
			`& &^ // bitwise-and, bitwise-clear (and-not)`
			
			`+ - // also concat`
			`| ^ // bitwise-or, bitwise-xor`
			
			`== != < <= > >=`
			
			`&& // logical-and`
			
			`|| // logical-or`

Special Operators (operators without precedence)

PHP	JS	Python 3	GO

Same like most C based languages	Same like most C based languages	Same like most C based languages	`, // comma, separating variables or values`
			`; // separating statements`
			`... // splatting array or variadic parameter`
			`. // calling method or data member of a struct`
			`: // getting a slice from an array or another slice`

DATA TYPES

PHP	JS	Python 3	GO

	TypeScript has boolean, number, string, array, tuple, enumeration, any, void, null, undefined and never.
	
	`let isDone: boolean = false;`
	
	`let decimal: number = 6;`
	`let hex: number = 0xf00d;`
	`let binary: number = 0b1010;`
	`let octal: number =`
			Go has boolean, numeric-non-decimal (int, int8, int16, int32, int64, uint, uint8, uint16, uint32, uint64, rune), non-decimal (float32, float64), string, interface, array, slice, maps
			
			`// using manifest type`
			`var positiveNumber uint8 = 89`
			`var negativeNumber = -1243423644`

SYMBOL

PHP	JS	Python 3	GO

	`let sym2 = Symbol("key"); let sym3 = Symbol("key");`
	
	`sym2 === sym3; // false, symbols are unique`		

Pointer, Array to pointer

PHP	JS	Python 3	GO
			var i int = 10; var p *int = &i; log.Printf("value_i %v, address_i %v,value_of_value_p %v, value_p %v, address_p %v\n",i,&i,*p,p,&p) //will print value_i 10, address_i 0xc420090000,value_of_value_p 10, value_p 0xc420090000, address_p 0xc42007c018 //attention to addres, may be address will print different value

Iterators and Generators, for-each, for..of, for..in

PHP	JS	Python 3	GO
	An object is deemed iterable if it has an implementation for the Symbol.iterator property. Some built-in types like Array, Map, Set, String, Int32Array, Uint32Array, etc. have their Symbol.iterator property already implemented. `let someArray = [1, "string", false];`	`std::string fruits[] = {"apple","grape","banana","melon"}; for (unsigned int i=0;i< sizeof (fruits)/sizeof(fruits[0]); i++){ std::cout<<"num "<<i<<" name "<<fruits[i]<<"\n"; } for (const auto& fruit : fruits) { std::cout<<fruit; } for (const auto& fruit`	`fruits := [4]string{"apple", "grape", "banana", "melon"} // take array and value for num, name := range fruits{ fmt.Printf("number : %d fruit : %s \n", num, name) // number : 0 Name : apple } // take value for _, name := range`

Array append

PHP	JS	Python 3	GO

	`let list = [4,5,6]`
	`list.push(7,8)`
		`[4,5,6] + [7,8]`
		
		`x = [4,5,6]`
		`x.extend([7,8])`
			`list := []int{4,5,6}`
			`list = append(list,7,8)`

Sub-array slicing

PHP	JS	Python 3	GO

	`let list = [4,5,6,7,8]`
	`list.slice(1,2) // [5]`
	`list.slice(2) //`
	`[6,7,8]`
	`list.slice(-2) // [7,8]`
	`list.slice(0,2) //`
	`[4,5]`
		`x = [4,5,6,7,8]`
		`x[1:2] # [5]`
		`x[2:] # [6,7,8]`
		`x[-2:] # [7,8]`
		`x[:2] # [4,5]`
			`list :=`
			`[]int{4,5,6,7,8}`
			`list[1:2] // []int{5}`
			`list[2:] //`
			`[]int{6,7,8}`
			
			`list[:3] // []int{4,5}`

Associative array

PHP	JS	Python 3	GO
	let someMap = {a: 1, b: 2, c: 3}; someMap['d'] = 4 someMap.e = 5 if(someMap['f'] == undefined) { // confirm ??? console.log('does not exists') // there's a better way } for(let k in someMap) { console.log("key",	someDict = {'a': 1, 'b': 2, 'c': 3} someDict['d'] = 4 someDict['f'] # KeyError for k in someDict: print(k, someDict[k]) d 4 b 2 c 3 a 1 someDict.get('f') #	someMap := map[string]int{\`a\`:1,\`b\`:2,\`c\`:3} someMap[\`d\`] = 4 if _, ok := someMap[\`f\`]; !ok { fmt.Println(\`does not exists\`) } for k, v := range someMap {// random order fmt.Printf(\`key %s have

Namespace

PHP	JS	Python 3	GO

			// every package is a new namespace
			// package must be on the same folder, for example:
			// Yay/bla.go Yay/foo.go
			// both bla.go and foo.go should declare package Yay (or anything else, but must be the same) at the first line
			// to export things, you must capitalize the first letter of the

Functions

PHP	JS	Python 3	GO
	function buildName(firstName: string, lastName?: string) { if (lastName) return firstName + " " + lastName; else return firstName; } let result1 = buildName("Bob"); // works correctly now let result2 = buildName("Bob",		func buildName(firstName, lastName string) { if lastName != `` { return firstName + ` ` + lastName } return firstName } result1 := buildName(`Bob`,``) // no default param in Go, except if you use ... result2 :=

Lambda expression, anonymous functions

PHP	JS	Python 3	GO
	`var res = function(a:number,b:number) { return a*b; }; console.log(res(12,2))` `var foo = (x:number)=> { x = 10 + x console.log(x) } foo(100)`		`res := func(a, b int) int { return a*b } fmt.Println(res(12,2))` `foo := func(x int) { x = 10 + x fmt.Println(x) } foo(100)`

CLASS DECLARATION

PHP	JS	Python 3	GO

			`type Car struct {`
			`Engine string`
			`}`
			
			`func (c Car) Display()`
			`{`
			``fmt.Println(`Engine is:``
			`` `+c.Engine)``
			`}`
			
			`func NewCar(engine`
			`string) (*Car) {`
			`return`
			`&Car{Engine:engine}`
			`}`

OBJECT CREATION AND METHOD CALL (BASED ON CLASS DECLARATION)

PHP	JS	Python 3	GO
	`let c = new Car('V8');` `c.disp();`		``c := NewCar(`v8`)`` `c.Display()`

INTERFACE DECLARATION

PHP	JS	Python 3	GO

			`type IPerson interface`
			`{`
			`// no data member allowed on interface`
			`SayHi() string`
			`}`

EXCEPTION HANDLING

PHP	JS	Python 3	GO

	If you're sure the code will be running synchronously then you can use try..catch otherwise you have to use the provided async library to catch the exception for instance in Promise you have to call catch() also in rxjs
	try{
	throw new Error('Dummy error');
	}
	catch(err){
			// using panic and recover is not idiomatic Go
			func ExceptionHandling() {
			defer (func(){
			if err := recover(); err != nil {
			fmt.Println(\`ERROR:\` + err.String())
			}
			})()
			panic(\`Dummy error\`)
			}
			
			// correct way in

STRING OPERATIONS

PHP	JS	Python 3	GO
	var str = new String("This is string"); console.log("str.charCodeAt(0) is:" + str.charCodeAt(0)); //str.charAt(0) is:84 console.log("str.charAt(0) is:" + str.charAt(0)); str.charAt(0) is:T var str1 = new String("This is string one");		str := `This is string` fmt.Println(`str[0] is ` + strconv.Itoa(int(str[0])) + "\n") fmt.Println(`str[0] is ` + str[0] + "\n") // not utf8-safe, must use for-range for utf8 encoding str1 := `This is string one` str2 := `This is string two`

OBJECT-ORIENTED CONSTRUCTORS

PHP	JS	Python 3	GO

`<?php`
`class BaseClass {`
`function __construct()`
`{`
`echo "In BaseClass constructor\n";`
`}`
`}`
 	
`class SubClass extends BaseClass {`
`function __construct()`
`{`
`parent::__construct();`
`echo "In SubClass constructor\n";`
			`// no constructor syntax on Go, you can initialize or use a function, see "class declaration" section`

PRIMITIVE TYPE CASTING

PHP	JS	Python 3	GO

	`let n:number = parseInt('123');`
	`let j:number = parseFloat('123.45');`
			`x := float64(125125125125)`
			`y := int(12857295.24)`

STRING TO INTEGER CONVERSION
INTEGER TO STRING CONVERSION

PHP	JS	Python 3	GO

	`let n:number = parseInt('123');`
	`let s:string = n.toString();`
			``i, err := strconv.Atoi(`125125125`)``
			
			`str := strconv.Itoa(1251251251)`

Character Literal

PHP	JS	Python 3	GO
			`single quote or rune(int)`

MULTILINE STRING SUPPORT

PHP	JS	Python 3	GO
	\`You can write multilines using the template string\`		yes, double quote or backquote

ESCAPE SEQUENCE STRING LITERAL AND STRING INTERPOLATION

PHP	JS	Python 3	GO
	`backquote ${1+1} \n` "double quote 2 \n" 'single quote 2 \n'		"double quote 2 \n"

NON-ESCAPE SEQUENCE STRING LITERAL

PHP	JS	Python 3	GO
			`backquote`

EXECUTING EXTERNAL COMMAND

PHP	JS	Python 3	GO

			`// use "os/exec" library`
			`out, err := exec.Command(`echo`, `123`).Output()`

FUNCTION OVERLOADING: DECLARING FUNCTION WITH SAME NAME BUT DIFFERENT TYPE OR NUMBER OF ARGUMENTS;
SUPPORT FOR DEFAULT ARGUMENT
SUPPORT FOR VARIADIC-LENGTH ARGUMENT

PHP	JS	Python 3	GO

	Not supported		no, only variadic argument
			
			`func test(a int, b ...string) {`
			`// len(b)`
			`}`
			
			`test(1)`
			``test(2,`a`)``
			``test(3,`a`,`b`)``

OPERATOR OVERLOADING

PHP	JS	Python 3	GO
Supported	Not supported		Not supported

MONKEY-PATCHING

PHP	JS	Python 3	GO
	Everything is open, dynamic, and assignable in JavaScript world unless for some restrictions defined using Object.defineProperty or something		Not supported

THREADING, CONCURRENCY MODEL

PHP	JS	Python 3	GO

	All .ts files will be converted or transpiled to .js files so it will use JavaScript threading model that is single thread. The JavaScript engine will act as a dispatcher that will put every function and callbacks into a queue. Concurrency can be handled using Promise or Reactive extensions like Rxjs.		message-passing (channel-based/CSP)
			
			go (func(){
			// run on another light-thread
			})()
			
			// use "chan" to create a queue

METAPROGRAMMING

PHP	JS	Python 3	GO

	Everything is transparent in JavaScript world but it still lacking no_method_found exception like what Ruby have		`// using built-in module:`
			`import "reflect"`
			
			`// then access the program data using:`
			`reflect.ValueOf(&anything)`

Garbage Collection

PHP	JS	Python 3	GO
			YES