一、语言基础

1、数据类型

1.1、类型概述

a、Java

在java语言中，变量分为两种：基本类型和引用类型。

基本数据类型是CPU可以直接进行运算的类型。Java定义了以下几种基本数据类型：

- 整数类型：byte、short、int、long
- 浮点数类型：float、double
- 字符类型：char
- 布尔类型：bool

b、Go

在Go语言中，数据类型用于声明函数和变量。

数据类型的出现是为了把数据分成所需内存大小不同的类型，编程的时候需要用大数据的时候才需要申请大内存，就可以充分利用内存。

所有类型都能赋值给interface{}空接口；GO语言中没有类的概念，只能通过struct和method来模拟；

Go语言按类别有以下几种数据类型：

1. 布尔型：bool
2. 数字类型：
    - 无符号整数：uint8、uint16、uint32、uint64

- 有符号整数：int8、int16、int32、int64
- 浮点数：float32、float64
- 其它类型：
  - byte、uint8 ： 两种类型可以互换，是一回事
  - rune、int32：两种类型可以互换，是一回事
  - int、uint：长度由CPU的位数决定(32或64位)、
  - uintptr：无符号整型，用于存放一个指针
- 复数：complex64、conplex128
- 字符：byte/uint8、rune
  - uint8/byte代表ASCII的一个字符
  - rune代表一个UTF-8字符，代表中文、韩文等字符。

3. 字符串类型：string
4. 派生类型：
   - 指针类型（pointer）
   - 数组类型：
   - 结构体：struct
   - channel类型
   - 函数类型
   - 切片类型
   - 接口类型（interface）
   - Map类型

c、Python

变量就是变量，它没有类型；Python使用对象模型来存储数据，构造任何类型的值都是对象。对象都拥有三个特性：身份、类型、值。

身份：每个对象都有一个唯一的身份标识自己，任何对象的身份都可以通过内建函数id()来得到。

类型：对象的类型决定了对象可以保存什么类型的值，可以进行什么样的操作，以及遵循什么样的规则。可以通过type()查看对象类型。返回的是对象而不是简单的字符串。

值：对象表示的数据

Python3中有六个标准数据类型：

1. Number（数字）：int、float、bool、complex(复数)
2. String（字符串）
3. List（列表）
4. Tuple（元组）
5. Set（集合）
6. Dictionary（字典）

- 不可变数据类型：Number、String、Tuple
- 可变数据类型：List、Dictionary、Set

d、JavaScript

JavaScript变量均为对象。当声明一个变量时，就创建了一个新的对象。

JavaScript不区分整数和浮点数，统一使用Number表示。

值类型（基本类型）：String、Number、Boolean、Null、Undefined、Symbol

引用类型（对象类型）：Object、Array、Function、RegExp、Date

1.2、数据类型对比

类型	Java	Go	Python	JavaScript	备注
byte	byte	byte			
bool	bool	bool	<class,bool>		
有符号整数	short/int/long	int8/int16/int32/int64/int/rune	<class,int>		Python bool继承int,可以当成int的0或1
无符号整数	--	uint8/uint16/uint32/uint64/uint/uintptr	--		
浮点数	float/double	float32/float64	<class,float>		
复数	--	complex	<class,complex>	--	
字符	char	byte/uint8/rune	--		Go中字符有2种，uint8/byte代表ASCII的一个字符，rune代表一个UTF-8字符
指针	--	pointer	--	--	
字符串	String	string	<class,str>	String	
空	null	nil	None	null、undefined	
元组	--	--	<class,tuple>	--	
数组	Array	array	--	Array	
List	ArrayList/LinkedList等	slice	<class,list>	List	
Map	HashMap/TreeMap等	map	<class,dict>	Map	
Set	HashSet/TreeSet等	--	<class,set>	Set	
结构体	--	struct	--	--	
函数	成员函数	func（函数和方法）	def	Function	
接口	Interface	interface		--	
类	Class	--	Class	Class	
Channel	--	channel	--	--	

2、变量

2.1、变量声明(定义)

a、Java

```
// 方式一：  数据类型 变量名 = 赋值;
// 方式二：  var 变量名 = 赋值;  // JDK10 局部变量类型推断，可以使用var代替实际类型

String name = "skwqy";
int age = 100;
var name2 = "skwqy";
```

b、GO

```
// 方式一：  var 变量名 数据类型 = 赋值      // 完整形式
// 方式二：  var 变量名 = 赋值              // 类型推断，省略数据类型
// 方式三：  变量名 := 赋值                 // 简短声明，省略var关键字

var name string = "skwqy"
var name = "skwqy"
name := "skwqy"
```

c、Python

```
# Python中变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建
name = 'skwqy'
name2 = "sk"
x = 10
```

d、JavaScript

```
// 方式一： var 变量名 = 赋值
// 方式二： let 变量名 = 赋值 （推荐）
// let 声明的变量仅在块级作用域中有效。注意：let不允许在相同的作用域内，重复声明同一个变量。
// --------------------------------------------------
{
    let a = 10;
    var b = 1;
}
a // ReferenceError: a is not defined.
b //1
// --------------------------------------------------
var a = [];
for (var i = 0; i < 10; i++) {
  var c = i;
  a[i] = function () {
    console.log(c);
  };
}
a[6](); // 9
// --------------------------------------------------
var a = [];
for (let i = 0; i < 10; i++) {
  let c = i;
  a[i] = function () {
    console.log(c);
  };
}
a[6](); // 6
```

2.2、常量

常量一旦声明，常量的值就不能改变。

a、Java

```
// Java 常量使用final来修饰
public static final float PI = 3.14;
```

b、Go

```
// Go 常量使用const来修饰
const PI float32 = 3.14
const PI2 = 3.14
```

c、Python

```
# Python 中没有定义常量的关键字，也不存在受编译器保护的常量
# 常量只是程序员之间的默契，不收解析器的保护：使用大写字母和下划线来表示常量
PI = 3.14
```

d、JavaScript

```
const PI = 3.14
```

2.2、常用数据结构

2.2.1、字符串

a、Java

```
方式一：  String name = "名字";              // 指向String常量池中的字符串
方式二：  String name2 = new String(...);  // 指向堆上的对象
```

b、Go

```
方式一：  var name string = "名字"      // 通过双引号("")来创建
方式二：  var name2 string = `名字`     // 通过反引号(``)来创建，也称为原始文本。不支持转义字符，可以跨越多行，可以包含除反引号之外的任何字符。通常，它用于在正则表达式或HTML中编写多行消息。
```

c、Python

Python中没有字符类型，一个字符也是字符串。

```
方式一：  name = '姓名'           # 单引号，单行
方式二：  name2 = "姓名"          # 双引号，单行
方式三：  name3 = '''姓名'''      # 三个单引号，多行
方式四：  name4 = """姓名"""      # 三个双引号，多行

原始字符串
方式一：  name = r'test\t123'     # 字符串不会转义，原样输出
方式一：  name2 = r"test\t123"    # 字符串不会转义，原样输出
方式三：  name3 = r'''姓名'''      # 字符串不会转义，原样输出
方式四：  name4 = r"""姓名"""      # 字符串不会转义，原样输出
```

2.2.2、数组

2.2.3、List 或 切片

2.2.3、Map 字典

2.2.4、Set 集合

2.2.5、元组

3、字符串和编码

Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。现代操作系统和大多数编程语言都直接支持Unicode。

ASCII编码和Unicode编码的区别：ASCII编码是1个字节，而Unicode编码通常是2个字节。

如果统一成Unicode编码，乱码问题从此消失了。但是，如果你写的文本基本上全部是英文的话，用Unicode编码比ASCII编码需要多一倍的存储空间，在存储和传输上就十分不划算。

所以，本着节约的精神，又出现了把Unicode编码转化为“可变长编码”的 `UTF-8` 编码。UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。如果你要传输的文本包含大量英文字符，用UTF-8编码就能节省空间：

字符	ASCII	Unicode	UTF-8
A	01000001	00000000 01000001	01000001
中	x	01001110 00101101	11100100 10111000 10101101

在计算机内存中，统一使用Unicode编码，当需要保存到硬盘或者需要传输的时候，就转换为UTF-8编码。

参考： https://www.liaoxuefeng.com/wiki/1016959663602400/1017075323632896

4、程序结构

4.1、选择结构

4.2、循环结构

二、函数

1、参数

2、返回值

三、面向对象

四、并发编程