常量：
字符常量：一定是单引号 ‘’ 括起来的一个数据

数据最小存储单位：字节（byte）
一个二进制数为：位 （bit）
八位为：字节 （byte）

1个二进制数据被称为位。bit
1字节 byte = 8bit
1k = 1024byte
1M = 1024k
1G = 1024M
1T = 1024G

在 JDK7 之后二进制是：0b 开头
八进制是：0 开头
十六进制是：0x、0X 开头

计算机是以补码形式存在，以补码形式计算的

正数的补码是本身
负数的补码是符号位不变，其它位取反后+1

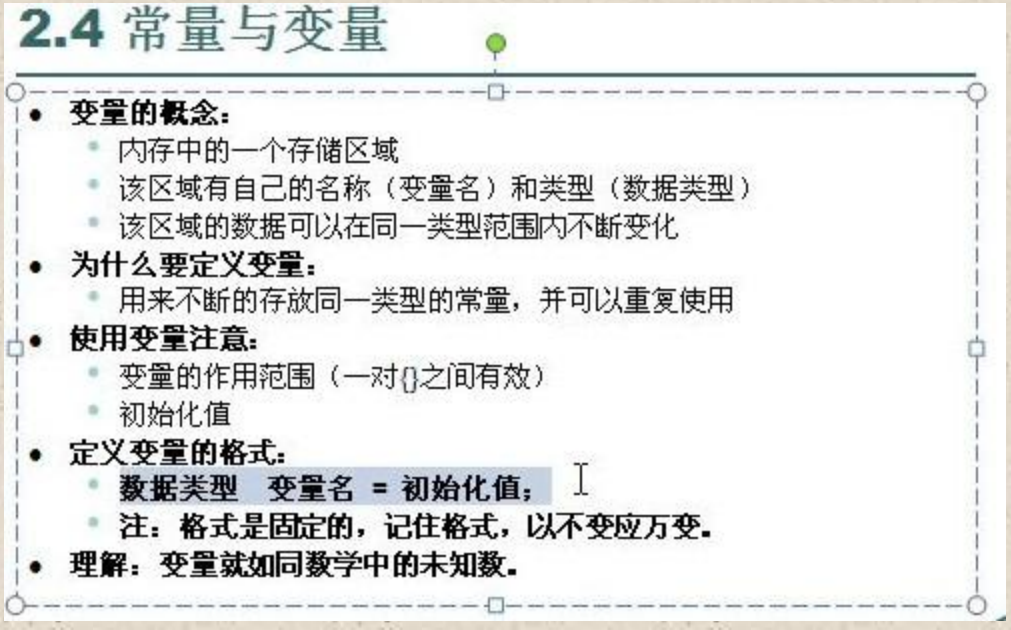
源码：本身
反码：正数反码是本身，负数反码——符号位不变，其他取反
补码：正数补码是本身，负数补码——反码+1

作业：1-1的结果是？
注意：
1+(-1)

第一步：获取数据的原码
1 00000001
-1 10000001
第二步：获取数据的补码
1 00000001
反 -1 11111110
补 -1 11111111
第三步：用补码计算
0 0000001
1 1111111
---
10 0000000
符号位也会参与运算，如果数据超出范围(8bit)，那么舍弃超出的。

变量

是内存中的一小块区域



boolean	byte	char	short	int	float	long	double
1	1	2	2	4	4	8	8

在一对大括号{}类内不能重复定义

使用前赋值，没有默认值

关于初始化的问题：

long 与 float 的初始化是必须要加 l、f

其他可以直接初始化，但不能超过精度

类型转换：

隐式转换：小类型转到大类型
显式转换：大类型转到小类型

Eg：byte b = 127;　//可以通过
　　b = b +1;　　//不可以通过
第一个式子中没有错，直接初始化只要不超过范围就可以了
第二个例子会出错，因为 3 默认是 int
　　byte b = 127;　　b = ++b;　//正确，值为-128
　　byte b = 127;　　b += 1;　//正确，值为-128　+=隐含了强制转换

```
面试题：下面的语句有没有问题，如果有，请解释。
    byte b1 = 3;
    byte b2 = 4;
    byte b3 = b1 + b2;
    byte b4 = 3 + 4;

    有问题，在byte b3 = b1 + b2;报错。
    解释：
        b1 + b2在参与运算的时候，首先会自动转换成int类型。
        在这个时候，b1和b2其实都被提升了类型。
        所以，他们的计算结果也应该是int类型。
        最终，就相当于直接把int类型赋值给byte类型，所以，会有精度的损失。

        如果参与运算的是常量，编译器会先计算值，在看该值是否是左边能够表示的范围。
        如果是，就不报错。
```

boolean	byte	char	short	int	float	long	double
1	1	2	2	4	4	8	8

几个基本数据类型的范围从小到大分别为：

byte、short、char<int<long<float<double

char、byte、short 之间的转换也需要强制转换

boolean 不参与转换

在 Java 语言中，short 类型的取值范围是-32768～32767，而 char 字符类型的编码值的取值范围是 0～65535。

因为 char 就是用十六位,而 short 用的是十五位

ASCLL 码表：

运算符：

++（前）、++（后）

在单独使用则效果一致

参与运算则++（前）的先自增后运算

++（后）的先运算后自增

“+”可用于算数加，也可以做字符串连接符

在字符串连接符时，从左往右

Eg：String s = “hello” + 2 + 3;　//输出为：hello23

String s = 2 + 3 ＋“hello”；　//输出为：5hello

String s = hello;　s = 4 + 5 ＋ s + 6; //输出为：9hello6

+=、-=、*=、/=、%=　会自动进行类型转换

```
int i = 1;
short s = 1;
s += i+1.2;
System.out.println(s);  //输出为：3
```

```
int a = 1; int b = 2;
System.out.println(a=b);  //输出为：2
```

可以理解：　a=b; //b 先将 2 赋值给 a，赋值完 b 就消失。所以只显示 a 的内容

逻辑运算：

&(与)　|（或）　!（非）　^（异或）

```
int a = 7;  // 0111

int b = 4;  // 0100

System.out.println(a & b);          //输出为:4

System.out.println(a<8 & b<9);  //输出为：true
```

&&(短路与)　　||（短路或）

它的左右两边都必须是 boolean 类型

&& 和 & 的区别

1. && 左右两边必须是 boolean 类型

   & 左右两边可以是整形，也可以是 boolean 类型

2. &无论如何，两边都参与运算

   &&若左边为 false，则不进行右边运算