变量是程序的基本组成单位

变量注意事项

1️⃣ 变量表示内存中的一个存储区域，不同的变量，不同的类型，占用空间的大小不同

2️⃣ 该区域有自己的名称

3️⃣ 变量必须先声明，后使用

4️⃣ 该区域的数据可以在同一类型范围内不断变化

5️⃣ 变量在同一个作用域内不能重名

Java的数据类型

⭐基本数据类型:

- 数值型：整数类型，存放整数如：`byte[1]`，`short[2]`，`int[4]`，`long[8]`
- 浮点数类型，`float[4]`，`double[8]`
- 布尔型：`True`，`False`，`boolean[1]`
- 字符型: `char[2]`，存放当个字符

⭐ 引用数据类型:

- 类
- 接口
- 数组

数据类型使用

1️⃣ **整形:**

类型	占用空间	范围
byte[字节]	1字节	-128-127
short[短整型]	2字节	-2^15·2^15-1(-32768-32767)
int[整形]	4字节	-2^31-2^31-1(-2147483648-2147483647)
long[长整形]	8字节	-2^63-2^63-1

⚠️ int类型注意细节

- Java每个整形类型都有固定大小，且不受操作系统的影响
- java整形默认是int，声明long类型需要后面加上'L'或者'l'

```
public class IntDetal {
    //main方法
    public static void main(String[] args){
        int n1 = 2;
        // 错误 int n2 = 2L;
        long n3 = 2L;
    }
}
```

- java程序中变量声明为int型，出发不足以表示大数，才使用long
- bit计算机中最小的存储单位。byte是计算机中基本存储单元，1byte=8bit

2️⃣ 浮点型：

类型	占用	范围
float 单精度	4字节	-3.403E38-3.403E38
double 双精度	8字节	-1.798E308-1.798E308

浮点型在机器中存放形式为符号位+指数位+尾数位，尾数部分可能丢失，造成进度损失(小数都是一个近似值)

引用GPT的一句话来解释：

> 在高级语言编程中，计算得到的小数通常是近似值，而不是精确值。这是由于计算机的二进制表示方式和有限的存储能力导致的。
>
> 计算机使用二进制来表示数字，而许多十进制小数无法在有限的二进制位数内准确表示。例如，十进制中的1/3（0.3333...）在二进制中是一个无限循环的小数。因此，计算机在存储这样的小数时只能使用有限的位数，导致了近似值的产生。
>
> 此外，浮点数表示法（floating-point representation）在计算机中用于存储小数，它采用科学计数法的形式，包括一个尾数和一个指数。由于浮点数的尾数有限，无法精确表示所有的小数，因此在进行浮点数运算时，可能会出现舍入误差，导致计算结果是近似值。
>
> 总的来说，由于计算机的二进制表示和有限的存储能力，高级语言编程中得到的小数通常是近似值而非精确值。

⚠️ 浮点数使用细节

- 浮点数也有固定长度且不受操作系统影响
- Java浮点数默认是`double`，如果要声明`float`类型就是,后面必须加上`'f'`或者`'F'`

```
```

```
2
3  public class FloatDetal {
4      //主方法
5      public static void main(String[] args){
6
7          // 这里也会报错，因为默认是double float f1=0.12;
8          float f3 = 0.12F;
9
10         //错误double范围要比float大 double f2=0.12F;
11
12         double f4 = 0.12;
13
14         double f5 = 0.13F; //对的float范围在double范围之内
15     }
16 }
```

- 有两种表现形式一种是十进制表示比如`5.12，4.23f，.512`，另一种是科学计数法比如`5.12e2[5.12*10^2],5.12E-2[]`

```
1
2  public class FloatDetal {
3      //主方法
4      public static void main(String[] args){
5
6          .....
7
8          //科学计数法
9          double f7 = 5.12e2; //5.12e2 == 5.12*10^2
10         System.out.println(f7); //512.0
11         double f8 = 5.12E-2; //5.12e2 == 5.12/10^2
12         System.out.println(f8);// 0.0512
13     }
14 }
```

- 通常情况下使用double，比float更加精确
  - 浮点数使用陷阱❗不要使用计算后的浮点数来做布尔运算，因为计算过的浮点数点是一个无限接近于它本身的值，如果需要判断就需要判断两个数的相减得到的值的绝对值，判断这个值是否在某个精度的范围内

```
1
2  public class FloatDetal {
3      //主方法
4      public static void main(String[] args){
5          .......
6
7          //浮点数陷阱
8          double f9 = 2.7;
9          double f10 = 8.1/3;
```

```
        System.out.println(f9); //2.7
        System.out.println(f10); //2.6999999999999997

        if (f9 == f10){
            System.out.println("这两个数相等"); //不会输出
        }
        //判断两个数的相减得到的值的绝对值，在判断这个值是否在某个精度的范围内，我这里定义为0.00000001
        //表示如果小于0.00000001就认为它们相等
        if (Math.abs(f9-f10) < 0.00000001){
            System.out.println("差值很小，达到指定的精确度内"); //输出
        }

    }
}
```

3️⃣ 字符类型：

字符类型单个使用`char`,多个使用`string`

⚠️ 字符类型使用细节

- 字符常量使用单引号括起来的单字符
- 转义字符为 ""
- 在java中，char的本质是一个整数，在输出时，是输出`unicode`码对应的字符[Unicode编码转换 - 站长工具 (chinaz.com)]
- char是可以和计算的,使用`unicode`码计算

```

public class Char01 {
    public static void main(String[] agrs){
        char c1 = 'a';
        char c2 = 97;
        char c3 = 't'+ 1;
        System.out.println((int)c1); //输出97
        System.out.println(c2); //输出a
        System.out.println(c1 + 10); //输出107
        System.out.println((int)c3); //117
        System.out.println(c3); //输出ASCII的对应的字符u
    }
}
```

⚠️字符类型本质: 字符型存储到计算机中，需要先将字符的ASCII码值找出来，在转换为二进制然后在保存到计算机中，读取就是反着来

ASCII码

Unicode码：字母和汉字都统一两个字节

utf-8：字母一个字节，汉字3个字节

gbk：字母一个字节，汉字2个字节

gb2312：

big5：繁体中文

4️⃣ 布尔类型:

多用于使用判断和循环的逻辑运算中

⚠️ 这里的True和false不可以使用数字替代

基本数据类型转换

⭐动类型转换

当java程序在进行赋值或者运算时，精度小弟类型自动转换为精度大的数据类型

1️⃣自动转换路线

```
char --> int --> long --> float --> double

byte --> short --> int --> long --> float --> double
```

1️⃣ 测试

```
public class Transform {
    //主方法
    public static void main(String[] args) {
        //低精度可以转换成高精度
        int a = 'c';
        System.out.println(a); //输出99
    }
}
```

⚠️自动类型转换注意细节

- 多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后在进行计算
- byte,short不能和char自动转换
- byte，short，char是可以运算的，不过在运算的时候会转换为int类型，只要是参与运算了都会被转换为int

```
public class Transform {
```

```
    //主方法
    public static void main(String[] args) {
        //低精度可以转换成高精度
        // int a = 'c';
        // System.out.println(a); //输出99


        int a = 10;
        float b = 20.1f;
        double c = 22.1;

        float d = a+b; //浮点型
        double e = a+c; //double
        System.out.println(d);
        System.out.println(e);




        byte b2 = 1;
        byte b7 = 4
        short b3 = 2;
        char b5 = 'a';
        short b4 = b2 + b3; //报错 b2+b3变为int类型了
        short b6 = b2 + b5; //报错和上面一样
        byte b8 = b2 + b7 //报错和上面一样

    }
}
```

- `boolean`不参与转换

⭐ 强制类型转换

1️⃣ 比如要强制转换为`int`格式为`(int)a`，但是有可能会造成精度降低或溢出，格外要注意

⚠️ 强制类型转换细节:

- 当进行数据的大小从 大--->小，就需要使用到强制转换
- 强转符号只针对于一个或者说是最近的一个数有效
- char类型可以保存int的常量值，但是不能保存int的变量值，需要强转

```
//强制类型装换


public class CastType {
    //主函数
    public static void main(String[] agrs){
```

```
        .....


        char c1 = 100;
        int m = 100;
        // char c2 = m; //错误char类型不能直接保存int的变量值，这里是将m->100->c2
        char c3 = (char)m;//这里是将m->100->ASCII->c3
        // System.out.println(c2);
        System.out.println(c3);
    }
}
```

- byte和short会转换为int类型

基本数据类型和String类型的转换

⭐基本数据类型转`string`语法为，基本数据类型的值+`""`

```
public class StringTest { //注意千万不能将类命名为String!
    //主入口
    public static void main(String[] args){
        int n1 = 100;
        float n2 = 1.1f;
        double n3 = 2.2;
        boolean n4 = true;
        String str1 = n1 +"";
        String str2 = n2 +"";
        String str3 = n3 +"";
        String str4 = n4 + "";
        System.out.println(str1+" "+str2+" "+str3+" "+str4); //输出100 1.1 2.2 true
}
```

⭐`string`转基本类型

```
        //String转基本类型
        String n5 = "123"
        String n6 = "1.234"
        int num1 = Integer.parseInt(n5);
        double num2 = Double.parseDouble(n6);
```

⚠ 注意事项

- 将String类型转换成基本数据类型使，要确保String类型能够转换成有效的数据
- 格式不对就会抛出异常