第三章 内置数据类型与运算符

int()
float()
bool()
str()

卷一：数学数据类型与其运算符

一、最基本内置数据类型介绍（p15）

1.1 python中内置的基本数据类型

整数型（int），浮点数型（float），布尔型（bool），字符串型（str）

1.2 数字（整数和浮点数）的运算符

运算符	说明	运算符	说明
+	加法	//	整数除法（整除）
-	减法	%	取余
*	乘法	**	次方
/	浮点数除法	注意：0不能做除数	error

使用divmod()可以同时得到1两个数的商和余数

```
divmod(num1, num2)
```

1.3 增强型赋值运算符

运算符+、-、*、/、//、**、%和赋值运算符 = 结合构成 增强型赋值运算符

例如：下面两个赋值式子作用等价

```
a = a + 1
a += 1  # 注意：运算符中间不能加空格
```

二、整数（p16）

2.1 整数的进制

二进制：0b　　八进制：0o　　十进制　十六进制：0x

2.2 使用int()实现类型转换

```
int('other-type')
```

注意：整数转为小数是截断

注意：转化类型后，生成了新的对象（旧的对象还存在）（拥有独立的id）

解释：在python中，每一个新值，都是一个新的对象

自动转换：在运算时，不同的数据类型向大的方向扩展

2.3 python中的整数大小

在python3中，int可以存储无限大的整数，而不会造成整数溢出（符合科学计算）

三、浮点数（p17）

3.1 浮点数

浮点数，称为float

浮点数在内存中是按照计算机科学计数法存储的（3.14 -> 0.314e1），分为小数部分和指数部分

3.2 类型转化float()：

```
float('other type')
```

自动转化：整数和浮点数的运算，数据类型转换为浮点数

3.3 四舍五入：

num1四舍五入num2位

```
round(num1, num2)
```

四、bool类型与逻辑性运算符（p20）

4.1bool值

true，false，但是本质还是1和0，甚至可以和数据相加

```
a = true  # bool类型
b = 3
c = a + b  # 结果为4
```

4.2 比较运算符

a = 15，b = 30：

运算符	描述	实例
==	用来比较两边的值是否相等	a == b，返回false

!=	是否不相等	a != b，返回true
>		
<		
>=		
<=		

4.3 逻辑运算符

运算符	格式	说明	原理
or 或	x or y	若x为true，则返回true 若x为false，则返回y的值	总结：从左往右，遇到第一个能出来的就出来，遇不到就返回下一个的值
and 与	x and y	若x为true，则返回y的值 若x为false，则返回false	
not 非	not x	颠倒真假	

4.4 同一运算符（p21）

运算符	描述	
is	判断是否（引用）同一对象	即：id相不相同
is not	不同一对象	

注意：同一运算符（is，is not）比较id，比较运算符（==等）比较的是value值

==默认调用对象的__eq__()方法（了解）

整数缓存问题

观察下面的程序：

```
a = 3
b = 3
print(a is b)  # true

c = 1000
d = 1000
print(c is d)  # true
```

但是在原理部分，python（命令行）仅仅对整数值比较小的数进行缓存（-5,256），但是解释器（pycharm，vscode）会进行一些优化，使任意整数都能保存并使用同一个对象

卷二：字符串类型与其运算符

六、字符串的操作

（p23）

6.1 特殊的字符串：转义字符

转义字符	描述
\（在行尾时）	续行符
\\	反斜杠符号
\' 、\"	单/双引号符
\b	退格
\n	换行
\t	横向制表
\r	回车

6.2 字符串拼接: +

1. 可以使用+号将字符串拼接（运算符按对象重载）

两边都是数字：加法；两边都是字符串：拼接；两边数据类型不同：报错。

2. 直接放在一起：'abc''def'

6.3 字符串复制：*

str * num能够复制num次字符串

```
a = "num "
print(a * 3)  # num num num
```

6.4 print()不换行打印

```
print("", end='')  # 注意：没有空格
```

' '中能够修改默认的结尾，添加什么都可以

6.5 控制台上输入字符串

```
a = input("请输入：")  # 可以输入提示字符串
print(a)
```

（p24）

6.6 数字转换成字符串型：str()

```
str(3.33)
str(True)
```

6.7 使用[]提取字符

字符串的本质是字符序列，用[]可以像序列那样提取单个的成员。（类似于c中的数组，但是操作方法更加多样）

正向搜索：最左侧的第一个成员的偏移量是0，到len(str - 1)为止。

反向搜索：最右侧的第一个成员的偏移量是-1，到-len(str)为止。

序列	a	b	c
正向编号	0	1	2
反向编号	-3	-2	-1

下面演示了正反编号的使用方法

```
a = 'abcdefghijklmnopqrstuvwxyz'
print("输出a的值\n" + a)
print("a[0] " + a[0])
print("a[3] " + a[3])
length = len(a)
print("a[len(a) - 1] " + a[length - 1])
print("a[-1] " + a[-1])
print("a[-len(a)] " + a[-length])

'''
输出a的值
abcdefghijklmnopqrstuvwxyz
a[0] a
a[3] d
a[len(a) - 1] z
a[-1] z
a[-len(a)] a
 '''
```

6.8 replace()实现字符串替换

字符串是不可变的。当试图通过[]方法修改字符串中的字符时发现报错了。但是我们有时确实需要修改字符串中的某些值。这时候就用到了.replace(被替换的字符串，用来替换的字符串)进行修改。

```
a = 'cbadefghijklmnopqrstuvwxyz'
print(a)
b = a.replace('cba', 'abc')    # 返回一个新的字符串，原字符串不改变
print(b)
```

'''
cbadefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
'''

解释：

a -> 'cbadefghijklmnopqrstuvwxyz'

a. replace('cba', 'abc') 创建了新的对象 'abcdefghijklmnopqrstuvwxyz'

b = a. replace('cba', 'abc')

所以b - > 'abcdefghijklmnopqrstuvwxyz'

6.9 字符串切片slice操作

切片（slice）操作可以让我们快速的提取字符串，标准格式为

str[起始偏移量start: 终止偏移量end: 步长step]

省略的情况下：

start：默认为第一个

end：默认为最后一个

step：默认为1

第二个:用不到可以省略

负数的情况下：

start：倒数

end：倒数

step：倒着数的步长

典型操作如下：

```
a[:] # 提取整个字符串
a[start:] # 从start开始到结尾（正序0开始，倒序-1开始）
a[:end] # 从头到end-1（end会忽略）
a[start:end] # 从start到end-1
a[start:end:step] # 从start到end-1，步长为step
```

6.10 split分割和join合并

.split()

作用基于指定的分割符将字符串分割成多个子字符串

操作值：字符串

参数：默认为空白，可以自己设计字符串用于分割

返回值：返回字符串的列表

```
a = 'to be or not to be'
a.split()
```

```
a.split(" ")
# ['to', 'be', 'or', 'not', 'to', 'be']
```

`.join()`

作用：按指定的方式合并字符串

操作值：分隔符

参数：字符串的列表

返回值：一个完整的字符串

```
a = ['to', 'be', 'or', 'not', 'to', 'be']
' '.join(a)
# to be or not to be
```