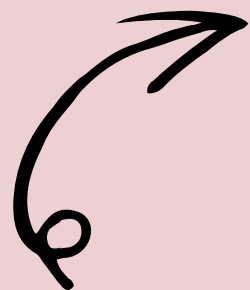


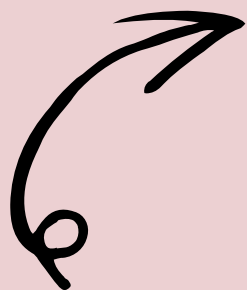
Python程序设计

主讲人：刘潇潇

单位：山东电子职业技术学院



第二章 Python基础语法



主要内容



0

1

基本语法

0

2

变量和
数据类型

0

3

标识符和
关键字

0

4

运算符

0

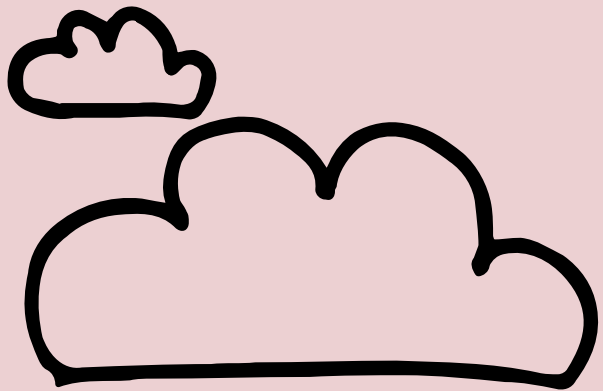
5

运算符
优先级

0

6

位运算



基本语法



注释

行与缩进

语句换行

注释

Python中的单行注释以#开头！



```
# 第一个注释
```

```
print('Hello, World') #第二个注释
```



练习

使用打印语句打印一段话、几个字符等，在打印语句上一行，使用注释，解释此句的作用,例如：



```
#告诉大家一个秘密
```

```
print('我的年龄是：'+str(18))
```



注释

多行注释可以使用**三引号（'''）**作为开头和结束符号



```
'''
```

```
这是第三个注释  
可以同时注释很多行
```

```
'''
```



练习

写一段简单的小程序，并在程序开头使用多行注释，对小程序进行解释，例如：

...

接下来的代码是一段非常厉害的代码

它用一个1

加上另外一个1

得出了2这个结果

...


a = 1

b = 1


c = a + b

行与缩进



python最具特色的就是使用**缩进**来表示**代码块**



```
if True:
    print ('TRUE')
else:
    print ('The answer is: ')
    print ('FALSE')
```




```
if True:
    print ('TRUE')
else:
    print ('The answer is: ')
    print ('FALSE')
```




练习

实现下图中的两种不同缩进的代码，在运行时将报错截图。




```
if True:
    print ('TRUE')
else:
    print ('The answer is: ')
    print ('FALSE')
```

```
if True:
    print ('TRUE')
else:
    print ('The answer is: ')
    print ('FALSE')
```




语句换行

Python 通常是一行写完一条语句，但如果语句很长，我们需要换行，这时可以使用**圆括号**来实现。




```
string=( 'Python是一种面向对象、解释型计算机程序设计语言，'  
        '由Guido van Rossum于1989年底发明。'  
        '第一个公开发行人版发行于1991年，'  
        '源代码同样遵循 GPL(GNU General Public License)协议。')
```




语句换行

需要注意的是，在 [], {}, 或 () 中的语句，不需要使用圆括号进行换行。



```
total = ['item_one', 'item_two', 'item_three',  
         'item_four', 'item_five']
```



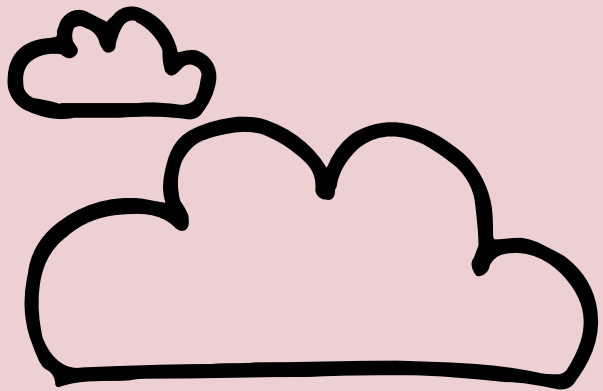
练习

使用 `MyRoomate = ['姓名1', '姓名2'...]` 这条语句创建一个数组，用于存放你所住的宿舍的所有人的姓名，使用语句换行。



```
MyRoomate = ['刘潇潇', '范甜甜', '孟然然',  
              '牛萌萌', '魏芳芳', '陈飞飞']
```





变量和数据类型




变量与赋值


变量的类型

变量和赋值

Python中的变量用来存储数据，其类型和值在赋值的那一刻被初始化。




```
number = 612  
letter = 'B'  
string = '小王子来自'  
  
print(string + letter + str(number) + '星球')
```




练习

编辑并运行如下程序，查看两条打印的结果



```
number = 612
letter = 'B'
string = '小王子来自'

print(string + letter + str(number) + '星球')
print('小王子来自%s%d星球'%(letter,number))
```



变量的类型

1. 数字类型

2. 布尔类型

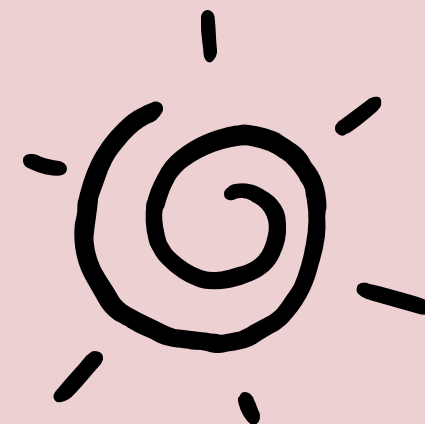
3. 字符串类型

4. 列表类型

5. 元组类型


6. 字典类型

- 整型
- 浮点型
- 复数



数字类型变量

Python中的数字类型包含整型、浮点型和复数类型




```
Integer = 100 #赋值整型变量  
Float = 3.1415923 #浮点型  
Complex = 3.12+1.23j #复数
```



练习

使用`print(type(变量))`语句打印以下三个变量的数据类型



```
Integer = 100 #赋值整型变量  
Float = 3.1415923 #浮点型  
Complex = 3.12+1.23j #复数
```



整型

整数类型（int）简称整型，它用于表示整数。


二进制：0b10100

八进制：010111


十进制：64

十六进制：0x14

整型



Python的整型可以表示的**范围是有限的**，它和**系统的最大整型一致**，例如，32位上的整型是32位，可以表示的数的范围是 $-2^{31} \sim 2^{31}-1$ 。在64位机器上的整型是64位的，可以表示的数的范围是 $-2^{63} \sim 2^{63}-1$ 。



浮点型



浮点型（Float）用于表示实数。


浮点型字面值可以用十进制或科学计数法表示。

<实数>E或者e<整数>


E或e表示基是10，后面的整数表示指数，指数的正负使用+或-表示。



浮点型



Python的浮点型遵循的是IEEE754双精度标准，每个浮点数占8个字节，能表示的数的范围是-1.8308~1.8308




复数

复数类型，用于表示数学中的复数，例如， $5+3j$ 。



两大特点

- (1) 复数由实数部分和虚数部分构成，表示为：
 $\text{real}+\text{imag}j$ 或 $\text{real}+\text{imag}J$
 - (2) 复数的实数 real 和虚数 imag 都是浮点型
- 

数字类型转换




函数	说明
<code>int(x [,base])</code>	将x转换为一个整数
<code>float(x)</code>	将x转换到一个浮点数
<code>complex(real [,imag])</code>	创建一个复数



布尔类型变量

布尔类型是特殊的整型，它的值只有两个，分别是
True和False。



```
bool1 = true  
bool2 = True  
bool3 = TRUE
```




布尔类型变量

以下对象的布尔值都是False:

- None
- False (布尔型)
- 0 (整型0)
- 0L (长整型0)
- 0.0 (浮点型0)
- 0.0+0.0j (复数0)
- "" (空字符串)
- [] (空列表)
- () (空元组)
- {} (空字典)


练习

编写并运行如下程序，查看打印结果。




```
bool1 = True
bool2 = (1 + 1 == 2)

if bool1 == bool2:
    print('Women are always right!')
```




列表和元组类型

列表和元组类似普通“数组”，可以保存任意数量、类型的值（元素）。




```
list_name = [1, 2, 'Hey'] #列表  
tuple_name = (1, 2, 'Hey') #元组
```



列表

列表中的元素使用[]包含，元素的个数和值可以随意修改。




```
list_name = [1, 2, 'Hey'] #列表  
list_name[1] = 'Hi'  
print(list_name)
```




元祖

元祖中的元素使用（）包含，元素不可以被更改。




```
tuple_name = (1, 2, 'Hey') #元祖
tuple_name[0] = 3
print(tuple_name)
```

```
File "/Users/liuxiaoxiao/PycharmProjects/Python程序设计/第二章.py", line 75, in <module>
    tuple_name[0] = 3
TypeError: 'tuple' object does not support item assignment
```




字典类型

字典是python中的映射数据类型，由键-值对组成。字典可以存储不同类型的元素，元素使用大括号{}来包含。




```
dict_name1 = {'name': '张三', 'class': 'J18018', 'age': 18}
dict_name2 = {'name': '李四', 'class': 'J18019', 'age': 17}
dict_name3 = {'name': '王五', 'class': 'J18020', 'age': 16}
```




练习

打印例题中的数据，查看结果。




```
dict_name1 = {'name': '张三', 'class': 'J18018', 'age': 18}  
dict_name2 = {'name': '李四', 'class': 'J18019', 'age': 17}  
dict_name3 = {'name': '王五', 'class': 'J18020', 'age': 16}  
print(dict_name1)  
print(dict_name2)  
print(dict_name3)
```




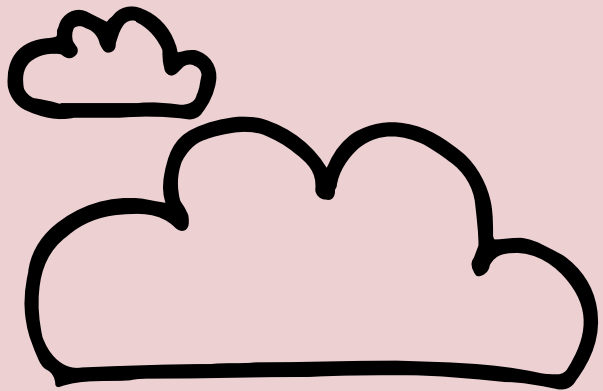
练习

将例题中的数据改写一下，按照姓名、班级、年龄，分类别统计。



```
dict_name = {'name1': '张三', 'name2': '李四', 'name3': '王五'}  
dict_class = {}  
dict_age = {}
```





标识符和关键字



标识符

关键字

标识符

定义：标识符（Identifier）是指用来标识某个实体的一个符号。在不同的应用环境下有不同的含义。



标识符

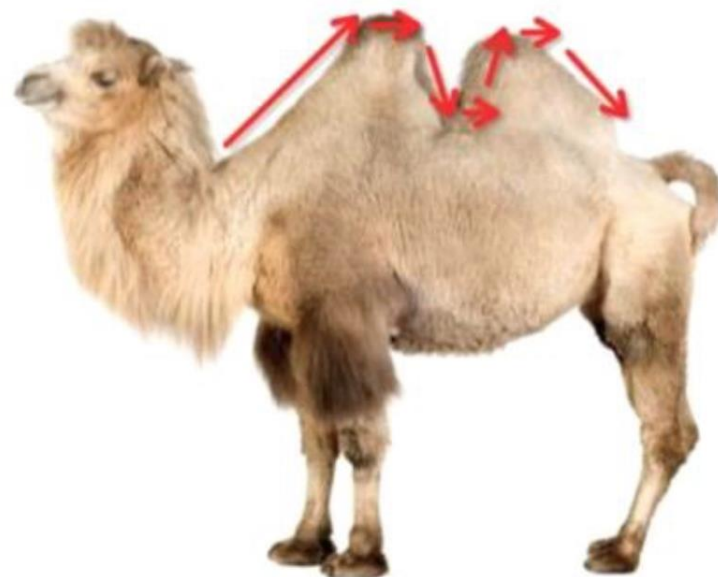
- 标识符由字母、下划线和数字组成，且数字不能开头。
- Python中的标识符是区分大小写的。
- python中的标识符不能使用关键字

标识符命名规则

- 见名之意



- 不建议使用驼峰式



如：

userName

userLoginFlag

关键字

关键字指的是具有特殊功能的标识符。



关键字含义

关键字	含义
False	布尔类型的值，表示假，与 True 相反
None	None 比较特殊，表示什么也没有，它有自己的数据类型 - NoneType。
True	布尔类型的值，表示真，与 False 相反
and	用于表达式运算，逻辑与操作
as	用于类型转换
assert	断言，用于判断变量或者条件表达式的值是否为真
break	中断循环语句的执行
class	用于定义类
continue	跳出本次循环，继续执行下一次循环
def	用于定义函数或方法

<code>del</code>	删除变量或序列的值
<code>elif</code>	条件语句，与 <code>if</code> 、 <code>else</code> 结合使用
<code>else</code>	条件语句，与 <code>if</code> 、 <code>elif</code> 结合使用。也可用于异常和循环语句
<code>except</code>	<code>except</code> 包含捕获异常后的操作代码块，与 <code>try</code> 、 <code>finally</code> 结合使用
<code>finally</code>	用于异常语句，出现异常后，始终要执行 <code>finally</code> 包含的代码块。与 <code>try</code> 、 <code>except</code> 结合使用
<code>for</code>	<code>for</code> 循环语句
<code>from</code>	用于导入模块，与 <code>import</code> 结合使用
<code>global</code>	定义全局变量
<code>if</code>	条件语句，与 <code>else</code> 、 <code>elif</code> 结合使用
<code>import</code>	用于导入模块，与 <code>from</code> 结合使用
<code>in</code>	判断变量是否在序列中
<code>is</code>	判断变量是否为某个类的实例
<code>lambda</code>	定义匿名函数



<code>nonlocal</code>	用于标识外部作用域的变量
<code>not</code>	用于表达式运算，逻辑非操作
<code>or</code>	用于表达式运算，逻辑或操作
<code>pass</code>	空的类、方法或函数的占位符
<code>raise</code>	异常抛出操作
<code>return</code>	用于从函数返回计算结果
<code>try</code>	<code>try</code> 包含可能会出现异常的语句，与 <code>except</code> 、 <code>finally</code> 结合使用
<code>while</code>	<code>while</code> 循环语句
<code>with</code>	简化 Python 的语句
<code>yield</code>	用于从函数依次返回值



运算符



0

1

算术运算符

0

2

赋值运算符

0

3

复合赋值
运算符

0

4

比较运算符

0

5

逻辑运算符

0

6

成员运算符

算术运算符

运算符	相关说明
+	加：两个对象相加
-	减：得到负数或一个数减去另一个数
*	乘：两个数相乘或是返回一个被重复若干次的字符串
/	除：x除以y
%	取余：返回除法的余数
**	幂：返回x的y次幂
//	取整除：返回商的整数部分


练习

请移步课本P35，例2-1



赋值运算符

赋值运算符只有一个，即=，它的作用是把等号右边的值赋给左边。例如， $x=1$



为多个变量赋同一个值： $x=y=z=1$

将多个值赋值给多个变量 $a, b = 1, 2$



复合赋值运算符

运算符	相关说明	实例
+=	加法赋值运算符	$c+=a$ 等效于 $c=c+a$
-=	减法赋值运算符	$c-=a$ 等效于 $c=c-a$
=	乘法赋值运算符	$c=a$ 等效于 $c=c*a$
/=	除法赋值运算符	$c/=a$ 等效于 $c=c/a$
%=	取模赋值运算符	$c\%=a$ 等效于 $c=c\%a$
=	幂赋值运算符	$c=a$ 等效于 $c=c**a$
//=	取整除赋值运算符	$c//=a$ 等效于 $c=c//a$

练习

请移步课本P37，例2-2



比较运算符

运算符	相关说明
<code>==</code>	检查两个操作数的值是否相当
<code>!=</code>	检查两个操作数的值是否相等
<code>></code>	检查左操作数的值是否大于右操作数的值
<code><</code>	检查左操作数的值是都小于右操作数的值
<code>>=</code>	检查左操作数的值是否大于或等于右操作数的值
<code><=</code>	检查左操作数的值是否小于或等于右操作数的值

练习

请移步课本P38，例2-3



逻辑运算符


运算符	逻辑表达式	描述
and	x and y	布尔“与”，如果x为False，x and y返回False，否则它返回y的计算值
or	x or y	布尔“或”，如果x为True，它返回True，否则返回y的计算值
not	not x	布尔“非”，如果x为True，返回False，如果x为False，它返回True

练习

请移步课本P39，例2-4



成员运算符



运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中，如果 x 不在 y 序列中返回 True。

练习

请移步课本P41，例2-5



运算符 优先级

➔ 从高到低
优先级的
所有运算符

运算符	描述
**	指数（最高优先级）
~ + -	按位翻转，一元加号和减号（最后两个的方法名为+@ 和 -@）
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^	位运算符
<= < > >=	比较运算符
<> <u>==</u> <u>!=</u>	等于运算符
= %= /= //=-= += *= **=	赋值运算符
<u>is</u> <u>is not</u>	身份运算符
<u>not</u> or and	逻辑运算符



练习

假设：

$$a = 20, b = 10, c = 15$$

$$d = 5, e = 0$$

那么

$$e = (a + b) * c / d$$

$$e = ((a + b) * c) / d$$

$$e = (a + b) * (c / d)$$

$$e = a + (b * c) / d$$

位运算

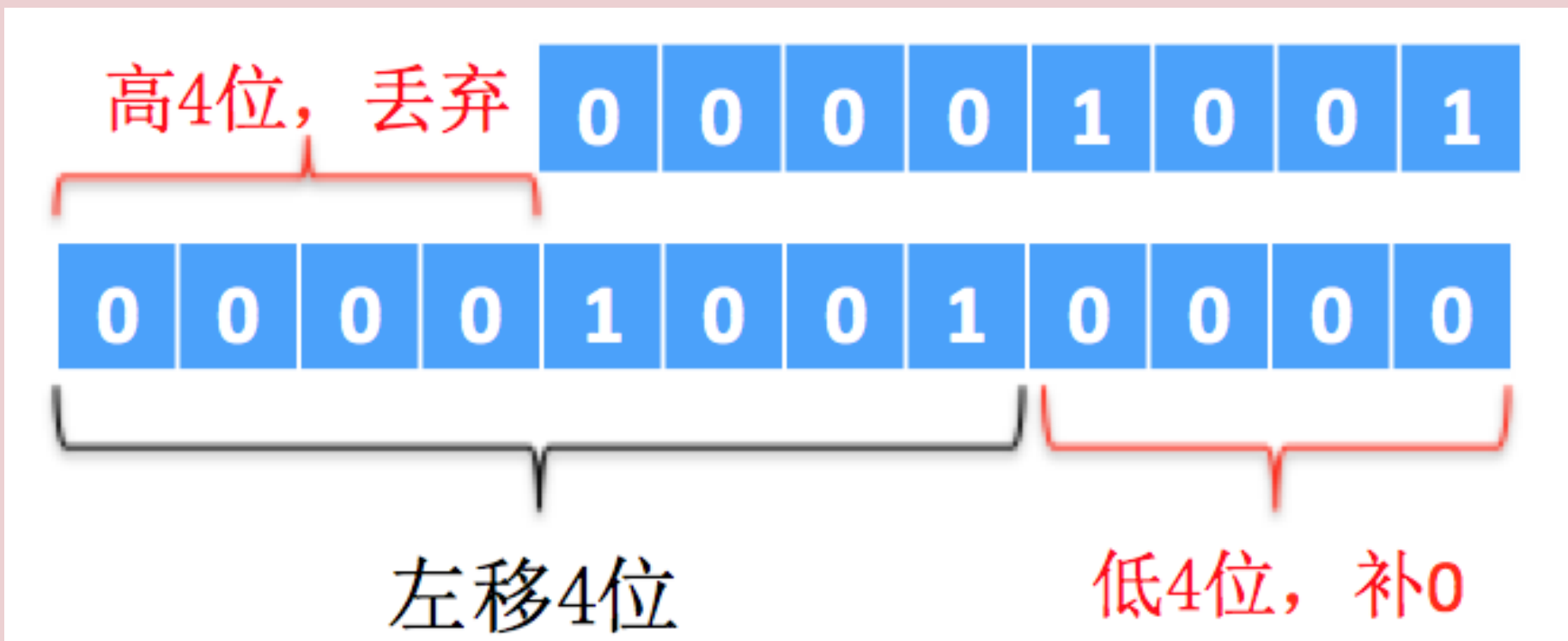
直接对整数在内存中的二进制位进行操作





按位左移

按位左移指的是二进制全部左移 n 位，高位丢弃，低位补0。





按位右移

按位右移指的是将二进制全部右移 n 位，移出的位丢弃，移进的位补符号位。按位右移的符号位保持不变





按位与

- 位与指的是参与运算的两个数各对应的二进位进行“与”的操作。只有对应的两个二进位都是1时，结果位就为1，否则结果位为0



	0	0	0	0	1	0	0	1
按位与 (&)	0	0	0	0	0	0	1	1
运算结果	0	0	0	0	0	0	0	1





按位或

按位或指的是参与运算的两个数各对应的二进位进行“或”的操作。只要对应的两个二进位有一个为1时，结果位就为1



按位或 (|)

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

运算结果

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---





按位异或

- 按位异或就是将参与运算的两个数对应的二进制位进行比较，如果一个位为1，另一个位为0，则结果为1，否则，结果位为0。



	0	0	0	0	1	0	0	0
按位异或 (^)	0	0	0	0	0	1	0	0
运算结果	0	0	0	0	1	1	0	0





按位取反

- 按位取反就是将二进制的每一位进行取反；
- 0取反为1，1取反为0



9的二进制	0	0	0	0	1	0	0	1
按位取反	1	1	1	1	0	1	1	0
+1 (原码)	1	0	0	0	1	0	1	0



Thank you

