浦东新区2023年Python教师培训项目

第3课-Python程序基础（二）

课程概要

本节课主要内容学习运算符、函数、面向对象、类和对象，学习定义类和实例化对象，了解面向对象的三大特征。

知识点

Python算术运算符

算术运算符主要是对两个对象进行算术计算的符号，其运算逻辑与数学的概念相似，因此比较好理解，常见的算术运算符有：

+：加，对两个对象进行相加运算;

-：减，一个数减去另一个数，或者得到负数;

/：除，一个数除以另外一个数;

*：乘，两个数相乘，或者返回一个被重复若干次字符串;

%：取模除，返回两个数相除的余数;

//：取整数，返回两个数相除所得商数的整数部分;

**：幂运算，返回X的Y次幂。

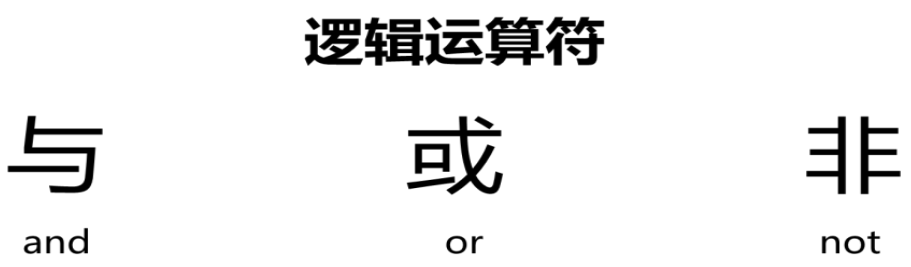
Python比较运算符

对于两个对象进行比较，其运算对象可以是数值也可以是字符串。

==：等于，判断两个对象是否相等，这里的相等是指两个变量的值相等而两个变量却不相同;

此外，其他比较运算符还包括：!=不等于，判断两个对象不相等，>大于，<小于，>=大于等于，<=小于等于。

Python逻辑运算符



and 和

英 /ənd; ən; n; ænd/
美 /ənd,ən,n,ænd/

逻辑表达式

示例用法: 条件1 and 条件2

与(and): 逻辑运算符，只有当两个条件都为真(True)时才为真(True)，否则为假(False)

试一试：

(1==1) and (2>3) False

True False

(1 == 1) and (3 != 4) True

True True

(1 == 2) and (3 != 4) False

False True

(2>3) and (3 != 3) False

False False

总结一下：

逻辑运算符	条件A	条件B	表达式	结果
and	True	True	A and B	True
	True	False		False
	False	True		False
	False	False		False

or 或者
英 /ɔ:(r)/ 美 /ɔ:r/

逻辑表达式

示例用法: 条件1 or 条件2

或(or): 逻辑运算符，当or两边的条件中只要有一个为真(True)，逻辑表达式就为真(True)，否则为假(False)

试一试：

(1==1) or (2>3) True

True False

(1 == 1) or (3 != 4) True

True True

(1 == 2) or (3 != 4) True

False True

(2>3) or (3 != 3) False

False False

总结一下：

逻辑运算符	条件A	条件B	表达式	结果
or	True	True	A or B	True
	True	False		True
	False	True		True
	False	False		False

not 不

英 /nɒt/
美 /nɑ:t/

逻辑表达式

示例用法: not 条件

非(not)： 逻辑运算符，对条件取反。
条件为真(True)，not之后表达式为假(False)
条件为假(False)，not之后表达式为真(True)

not (1==1) False
True

not (3 > 4) True
False

与(and) 同真为真 否则为假

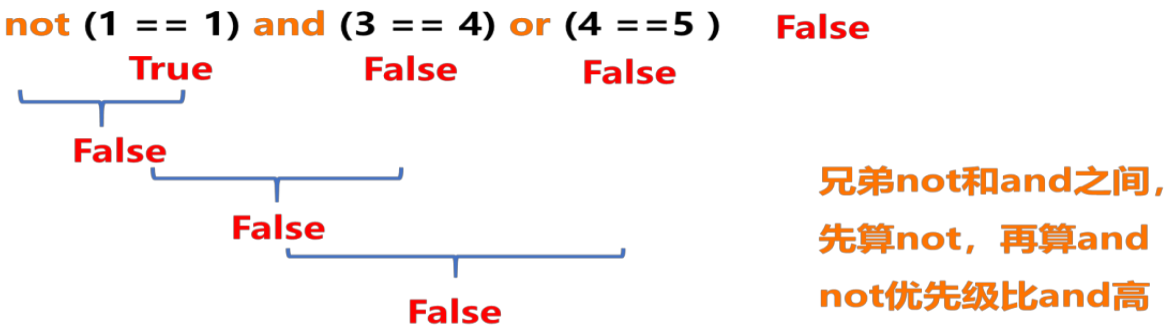
或(or) 有真为真 否则为假

非(not) 结果取反

逻辑运算符的优先级

(2<= 3) and (1==1) or (2 > 3) and (4 !=4) True
True True False False
True False
True

兄弟or和and之间，
先算and，再算or
and优先级比or高



函数

我们在定义函数的时候除了函数名字和函数功能，我们在创建自己的函数的时候，还需要有一个关键字。得让人一看就知道，这是一个函数。

函数（方法）的好处：

函数是封装好的用来实现特定逻辑功能的代码段

函数在程序中是可以重复调用的

函数能提高代码的重复利用率

程序运行的过程中，首先看到要调用什么函数，然后通过函数名，去找对应的函数，接下来才是去执行这个函数内部的代码。

我们可以通过定义能传参的函数来实现我们想要的功能：在需要改变的地方，放上变量；同时需要在方法名的后面的小括号里面，传递形参，用来告诉使用这个方法的用户，要想使用这个方法，需要传递哪些参数；在最后调用方法的时候，传递和形参对应的实参，来调用方法。

关键字：

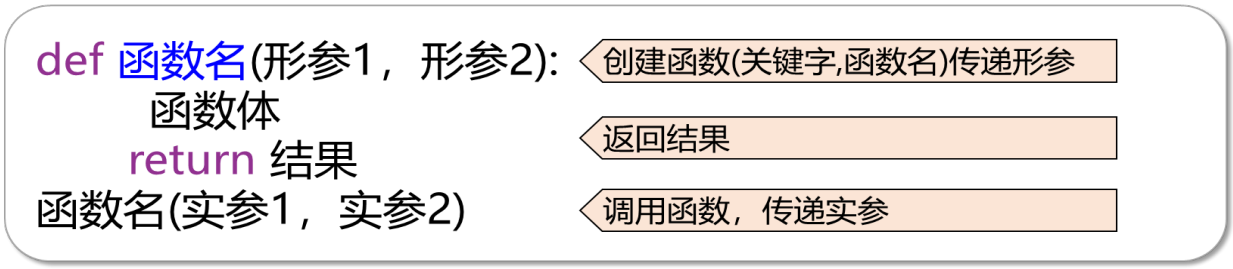
definition 定义
英 /ˌdefɪˈnɪʃn/ 美 /ˌdefɪˈnɪʃn/

function 函数
英 /ˈfʌŋkʃn/ 美 /ˈfʌŋkʃn/

示例用法： def 函数名( ):
函数功能

方法，也叫函数

函数代码块以 def 关键词开头，后接函数名称和小括号 ()
任何形参必须放在小括号里面。括号后面跟冒号，函数体部分缩进
形参：在定义函数的时候给出的变量叫形参(名字任意)
实参：在调用函数的时候传递的实际数据叫实参
return 结果 结束函数



面向对象基本特征

类(Class)：用来描述具有相同的属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。对象是类的实例。

方法：类中定义的函数。

类变量：类变量在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。

数据成员：类变量或者实例变量用于处理类及其实例对象的相关的数据。

方法重写：如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖（override），也称为方法的重写。

局部变量：定义在方法中的变量，只作用于当前实例的类。

实例变量：在类的声明中，属性是用变量来表示的，这种变量就称为实例变量，实例变量就是一个用 self 修饰的变量。

继承：即一个派生类（derived class）继承基类（base class）的字段和方法。继承也允许把一个派生类的对象作为一个基类对象对待。例如，有这样一个设计：一个Dog类型的对象派生自Animal类，这是模拟"是一个（is-a）"关系（例图，Dog是一个Animal）。

实例化：创建一个类的实例，类的具体对象。

对象：通过类定义的数据结构实例。对象包括两个数据成员（类变量和实例变量）和方法。

类定义

```
#语法格式如下：
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
#类实例化后，可以使用其属性，实际上，创建一个类之后，可以通过类名访问其属性。
```

类对象

类对象支持两种操作：属性引用和实例化。

属性引用使用和 Python 中所有的属性引用一样的标准语法：**obj.name。**

类对象创建后，类命名空间中所有的命名都是有效属性名。所以如果类定义是这样:

```
class MyClass:
    """一个简单的类实例"""
    i = 12345
    def f(self):
        return 'hello world'

# 实例化类
x = MyClass()

# 访问类的属性和方法
print("MyClass 类的属性 i 为: ", x.i)
print("MyClass 类的方法 f 输出为: ", x.f())
```

类的方法

在类的内部，使用 **def** 关键字来定义一个方法，与一般函数定义不同，类方法必须包含参数 self, 且为第一个参数，self 代表的是类的实例。

```
#类定义
class people:
    #定义基本属性
    name = ''
    age = 0
    #定义私有属性,私有属性在类外部无法直接进行访问
    __weight = 0
    #定义构造方法
    def __init__(self,n,a,w):
        self.name = n
        self.age = a
        self.__weight = w
    def speak(self):
        print("%s 说: 我 %d 岁。" %(self.name,self.age))

# 实例化类
p = people('runoob',10,30)
p.speak()
```

类属性与方法

类的私有属性

__private_attrs：两个下划线开头，声明该属性为私有，不能在类的外部被使用或直接访问。在类内部的方法中使用时 **self.__private_attrs。**

类的方法

在类的内部，使用 def 关键字来定义一个方法，与一般函数定义不同，类方法必须包含参数 **self**，且为第一个参数，**self** 代表的是类的实例。

self 的名字并不是规定死的，也可以使用 **this**，但是最好还是按照约定是用 **self**。

类的私有方法

__private_method：两个下划线开头，声明该方法为私有方法，只能在类的内部调用 ，不能在类的外部调用。**self.__private_methods。**

```
class JustCounter:
    __secretCount = 0  # 私有变量
    publicCount = 0    # 公开变量

    def count(self):
        self.__secretCount += 1
        self.publicCount += 1
        print (self.__secretCount)

counter = JustCounter()
counter.count()
counter.count()
print (counter.publicCount)
print (counter.__secretCount)  # 报错，实例不能访问私有变量
```

相关学习网站

菜鸟教程：https://www.runoob.com/python3/python3-class.html

鱼C课堂：http://study.163.com/course/introduction/378003.htmutm_source=weChat&utm_medium=webShare&utm_campaign=share&utm_content=courseIntro