1. 程序异常会显示异常的代码位置以及异常内容，缩进异常类型表示为IndentationError

2. 在Python中，变量的类型可以随时改变，可以使用内置函数**type()**返回变量类型。

3. Python中的函数可以分为2类：内置函数、自定义函数

4. print（” xxx”,end=”\n”）//默认\n

5. print（x,y,sep=“：”）//分割符

6. eval()函数可以解析和执行字符串表达式，并返回表达式的计算结果。

7. 模块是一种以“.py”为后缀的文件，其中可以包含变量、函数等各种代码形式。

8. 导入模块有两种方法

* import 模块名
* from 模块名 import··

使用

* 模块名.变量
* 模块名.函数（参数）

9. 内置模块random()

|  |  |
| --- | --- |
| **random()** | **返回一个0到1的随机浮点数n（0≤n≤1）** |
| randint(a,b) | 返回一个指定范围内的整数n（a≤n≤b） |

10.

可以使用**as关键字对模块进行重命名**，例如，在导入random时，为其起别名r

import random as r

11. Python中的模块分为3类：

**内置模块** **第三方模块** **自定义模块**

12. 为了避免作为模块的文件中的代码被执行，可以在study.py中添加

**if \_\_name\_\_ == "\_\_main\_\_":**

**第三章**

1. 数字类型有**整型（int）、浮点型（float）、复数型（complex）**，**布尔类型（bool）**。
2. **二进制整数 在数字前加上0b或者0B；**
3. **八进制整数在数字前加上0o或者0O。**
4. **十六进制整数在数字前加上0x或者0X**
5. Python中的浮点型的取值范围大约为**-1.8e308~1.8e308**，超出这个范围，Python会将其视为**无穷大（inf）**或者**无穷小（-inf）**
6. 浮点型最长可输出16个数字，当浮点数进行运算后，最长可输出17个数字。然而，**计算机系统只能提供15个数字的准确性**，最后一位由计算机系统根据二进制计算结果确定，存在一定误差
7. **复数由实部和虚部构成**，形如3+2j
8. a = 3.2 + 1.1j

print(a.real) #打印复数的实部

print(a.imag) #打印复数的虚部

1. 可以通过内置函数**complex(real,imag)传入实部和虚部**的方式，如果**没有传入虚部，则虚部默认为0j；**

a = complex(2,3) #传入实部2，虚部3

b = complex(5) #传入实部5，没有传入虚部

1. 算数运算符 + - \* / %取本身的意思 //取整除 \*\*幂运算2\*\*3 ==8
2. 在双引号定义的字符串中可以含有单引号，但是不能直接含有双引号，同理在单引号定义的字符串中也不能直接含有单引号。
3. 如果需要忽略字符串中的转义字符，可以在字符串的前面添加**r**或者**R**。

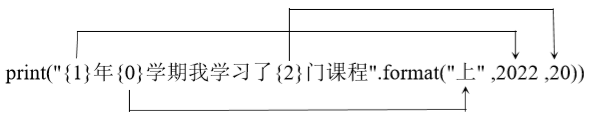
r'C:\Windows\tracing'

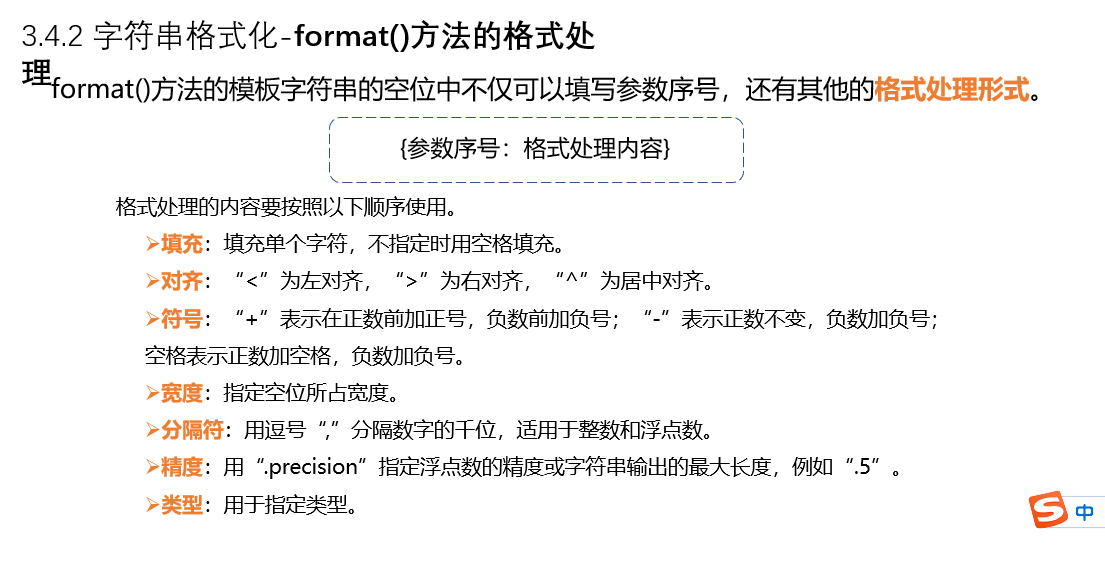
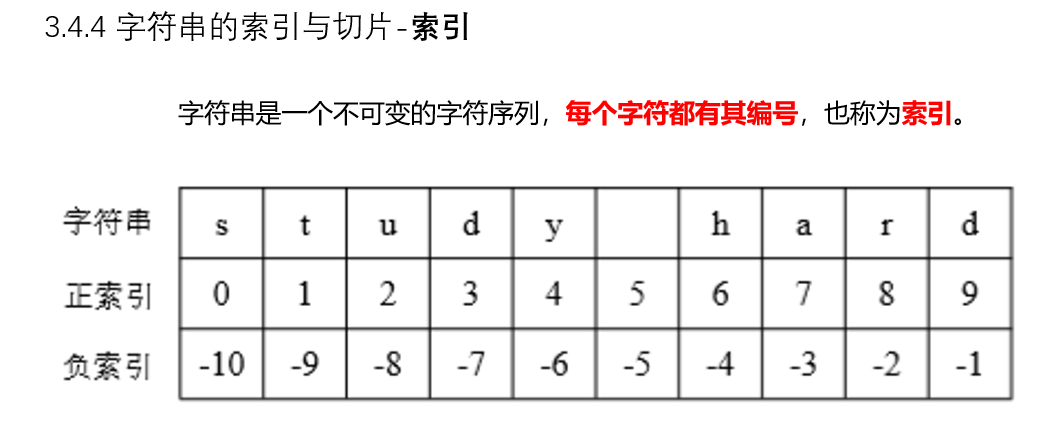
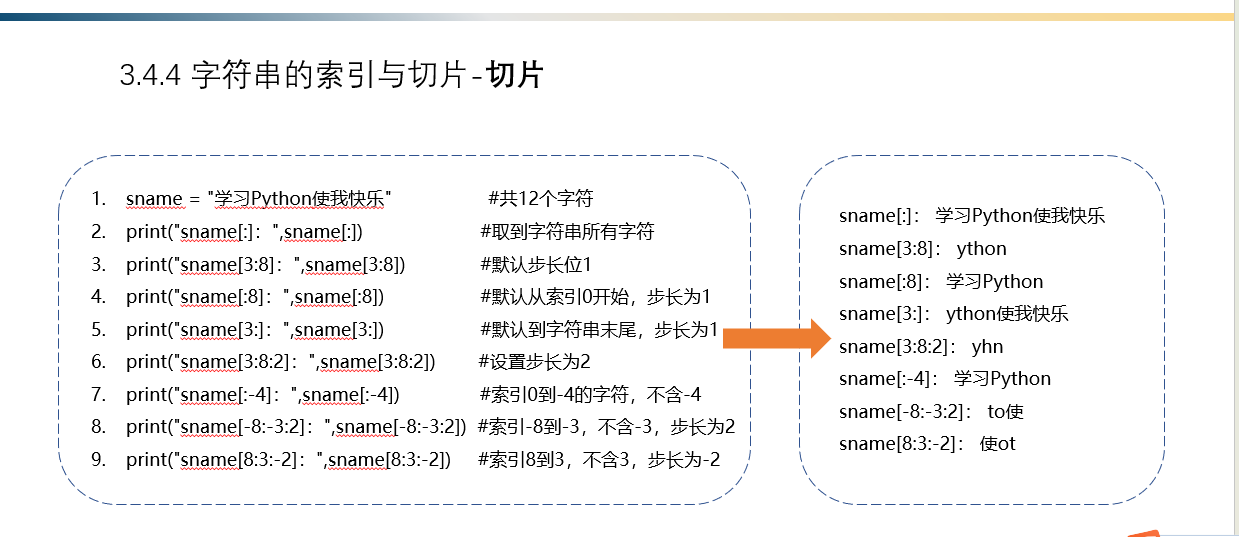
1. 三引号中（单引号或双引号）可以包含单引号、双引号，无需转义，**所有字符均以原始形态打印出来**
2. **字符串的格式化format（）**

print("{ }年{ }学期我学习了{ }门课程".format(2022,"上",20))

图示

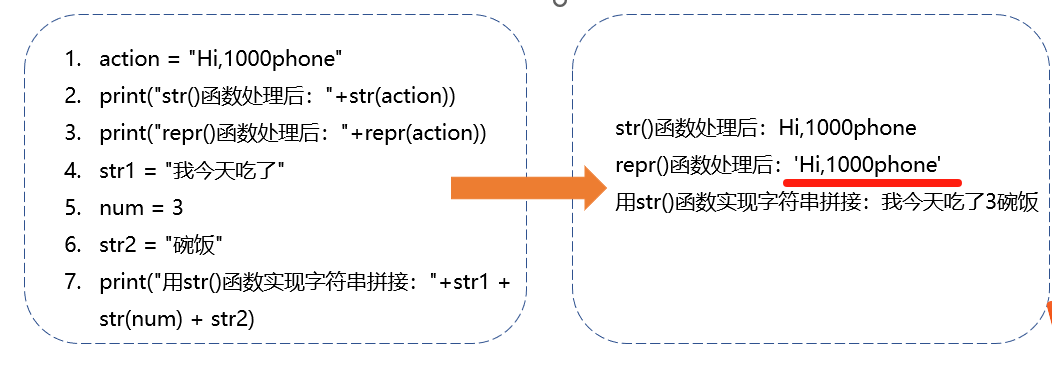
描述已自动生成



1. 
2. 
3. 
4. “+”只能用于相同类型之间的拼接
5. str()函数和repr()函数是将对象转换为字符串形式的两种机制

**str()函数**：会将对象转换为合理形式的字符串，以便用户理解

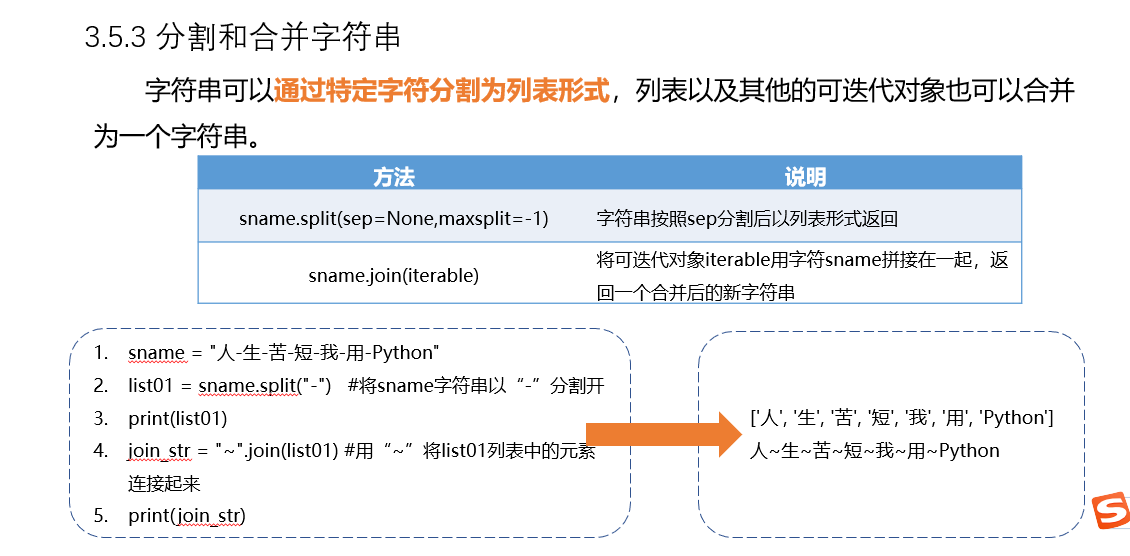
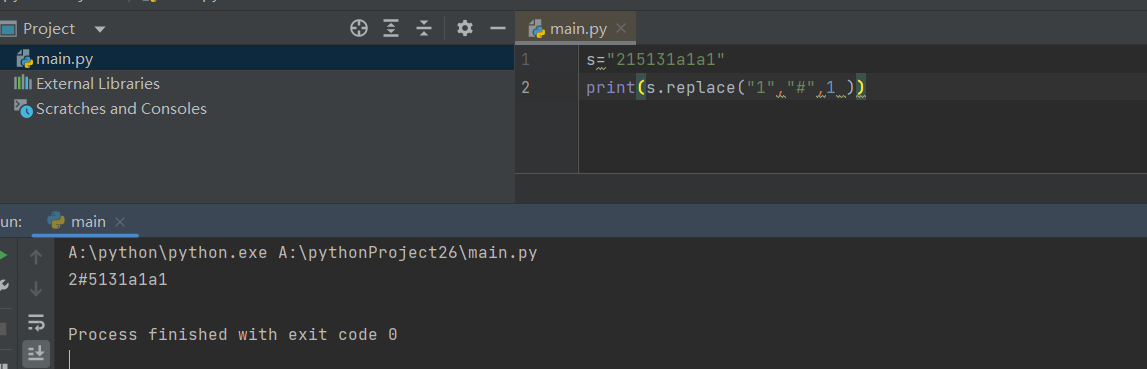
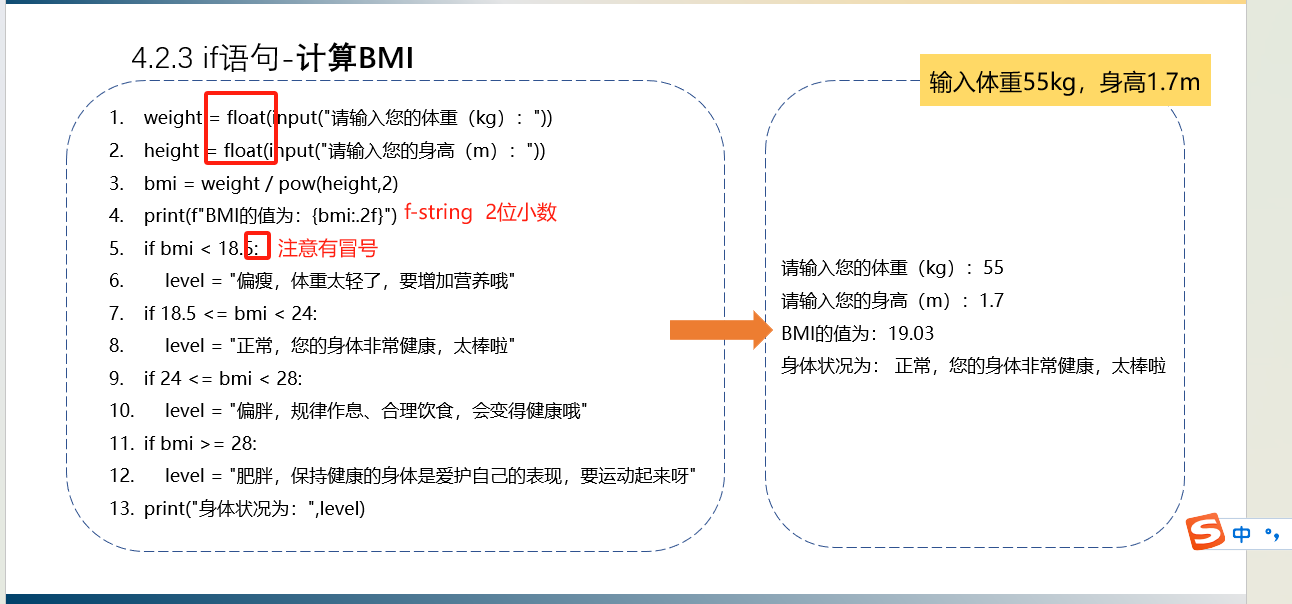
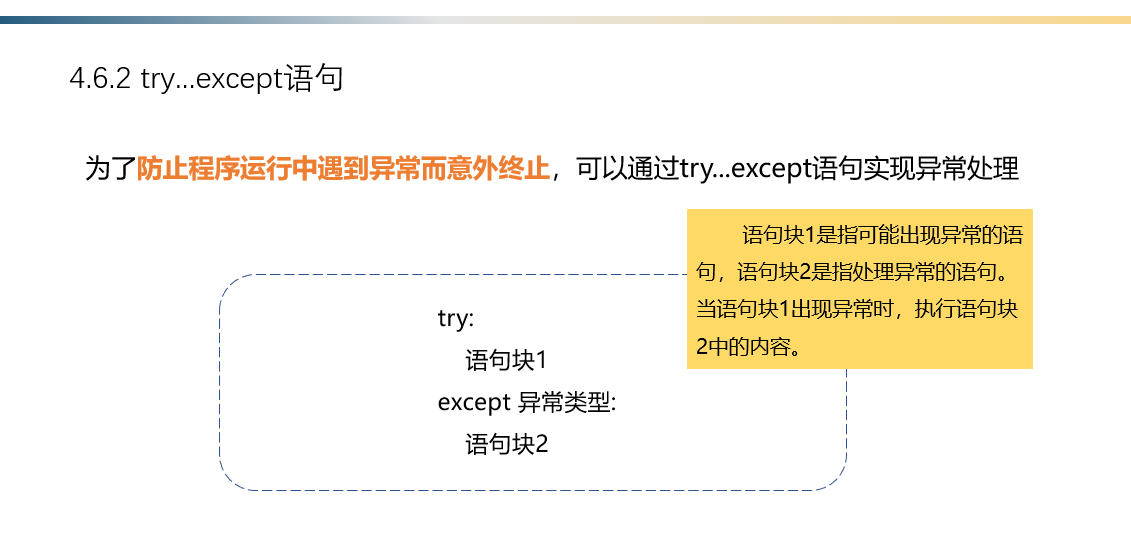
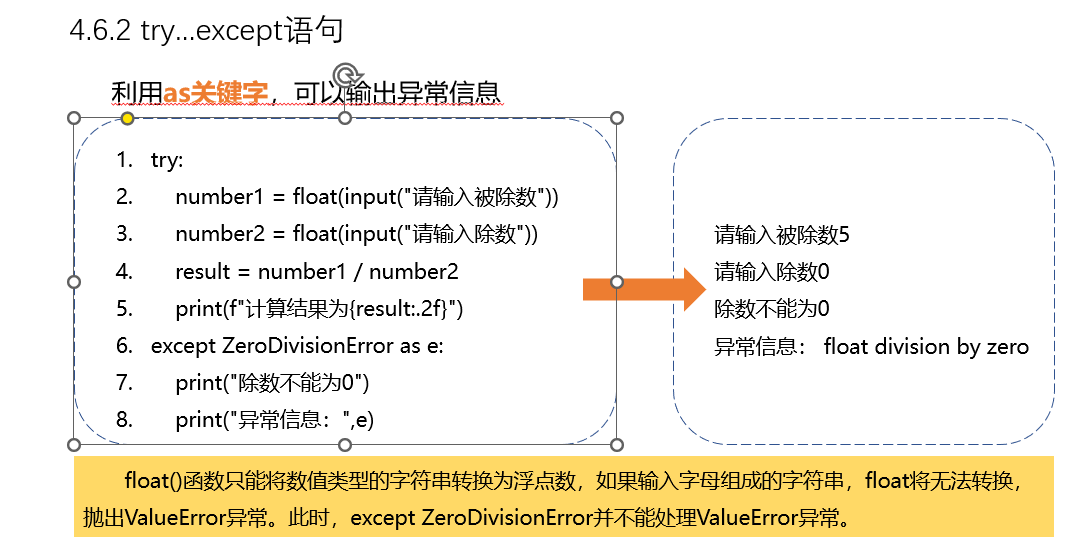
**repr()函数**：会创建一个字符串，用合法的Python表达式的形式来表示对象，以供Python解释器读取



1. 字符串的成员归属需要用到**成员运算符**，成员运算符能够**判断指定序列中是否包含某个值**。

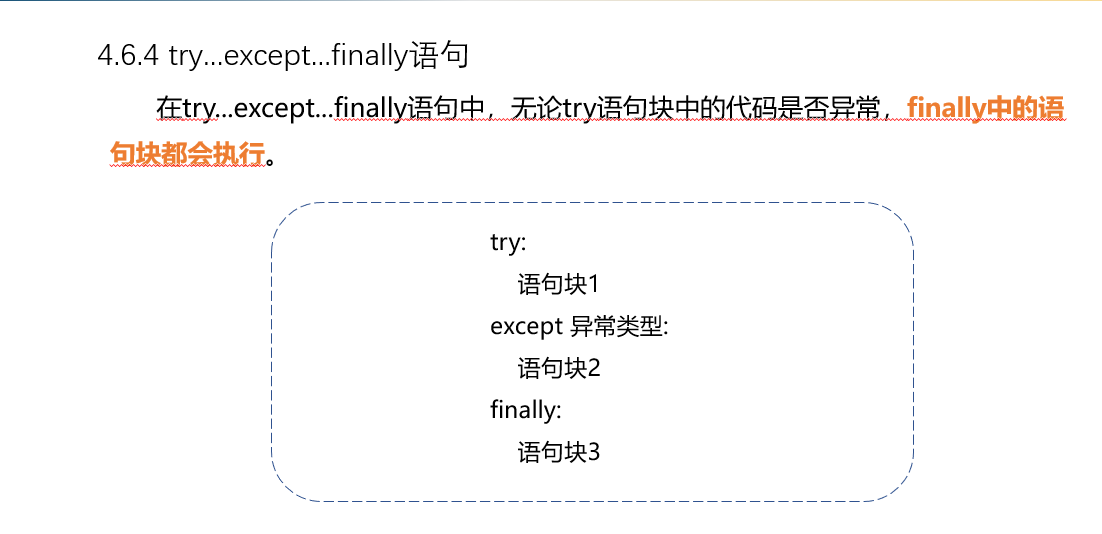
**in**：如果在指定序列中找到值，返回True，否则返回False。

**not in**：如果在指定序列中没有找到值，返回True，否则返回False。

1. 
2. 
3. index()方法在**没有检索到子串**时会报错
4. 
5. 程序的基本结构：顺序、分支、循环
6. 
7. Python可以使用这种语法格式
8. for语句和while语句都可以与else语句搭配使用，**else下的语句块只在循环正常结束时执行**，如果循环因为break语句等中断退出，else下的语句块不会执行。
9. 除法运算中，当除数为0时，会发生异常**ZeroDivisionError**
10. 
11. 
12. 图片包含 表格

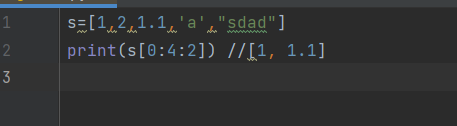
    描述已自动生成
13. 文本

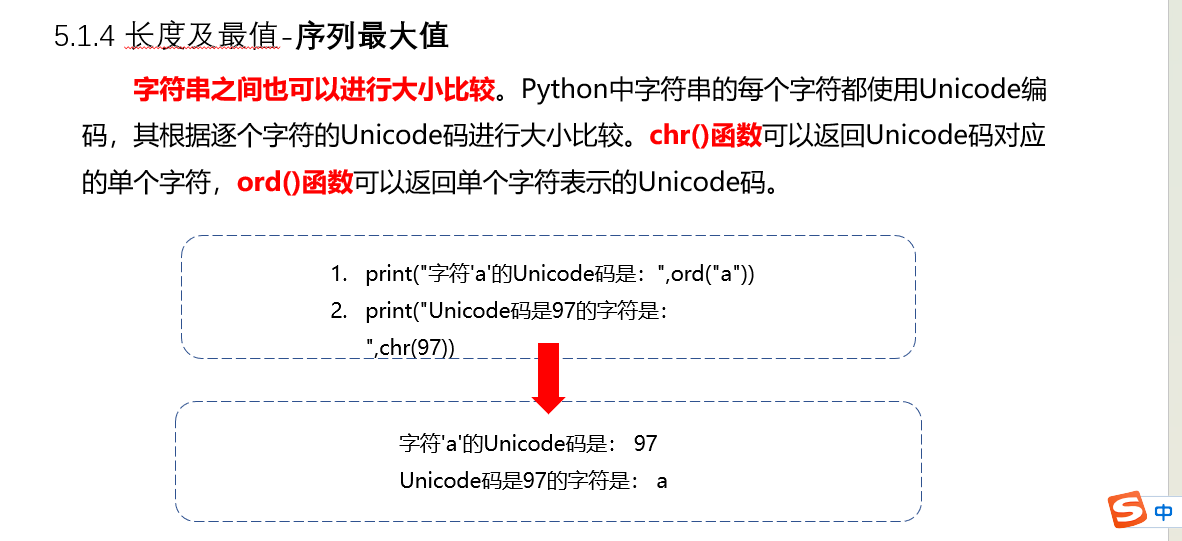
    低可信度描述已自动生成
14. 在try...except...finally语句中，无论try语句块中的代码是否异常，**finally中的语句块都会执行**。

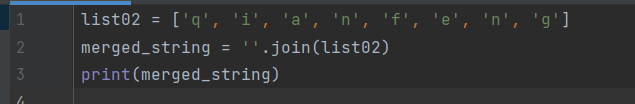


1. 图形用户界面

   中度可信度描述已自动生成
2. 切片是指**从序列中截取部分元素组成新的序列**，且不会使原序列产生变化。



1. **字符串之间也可以进行大小比较**。Python中字符串的每个字符都使用Unicode编码，其根据逐个字符的Unicode码进行大小比较。**chr()函数**可以返回Unicode码对应的单个字符，**ord()函数**可以返回单个字符表示的Unico de码。
2. 
3. 当序列中**存在类型不同的元素**时，无法进行比较，不能返回其中的最大元素
4. 将列表合并

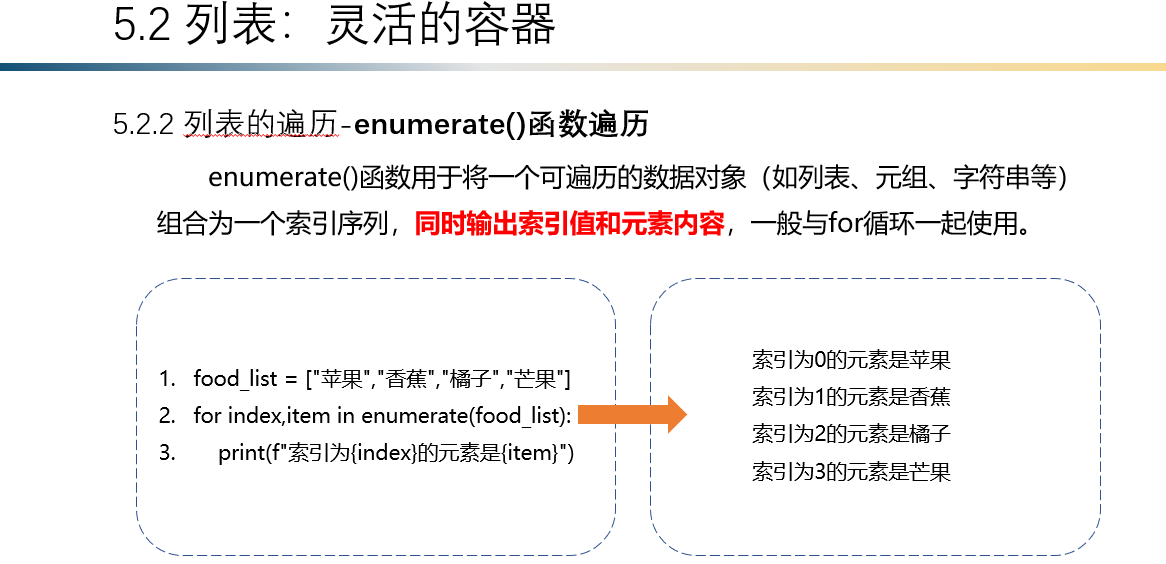


1. 列表的遍历即获取列表中每一个元素的值，常用的遍历方式有三种，分别为：

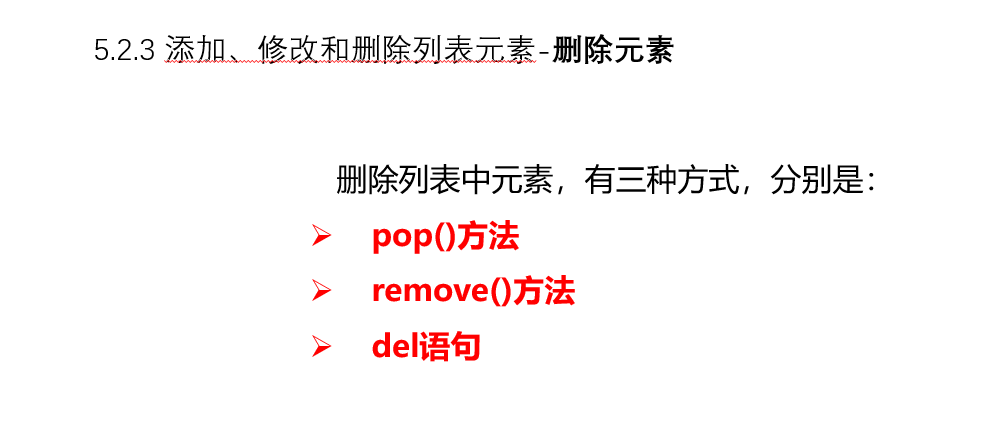
**for循环**直接遍历

**range()函数**索引遍历

**enumerate()函数**遍历



1. 图形用户界面, 应用程序

   描述已自动生成
2.   
   图示

   描述已自动生成  
   remove()方法用于删除列表list中出现的**第一个值为x的元素**，没有返回值，如果找不到x，**程序会发生ValueError异常**。

图示

低可信度描述已自动生成

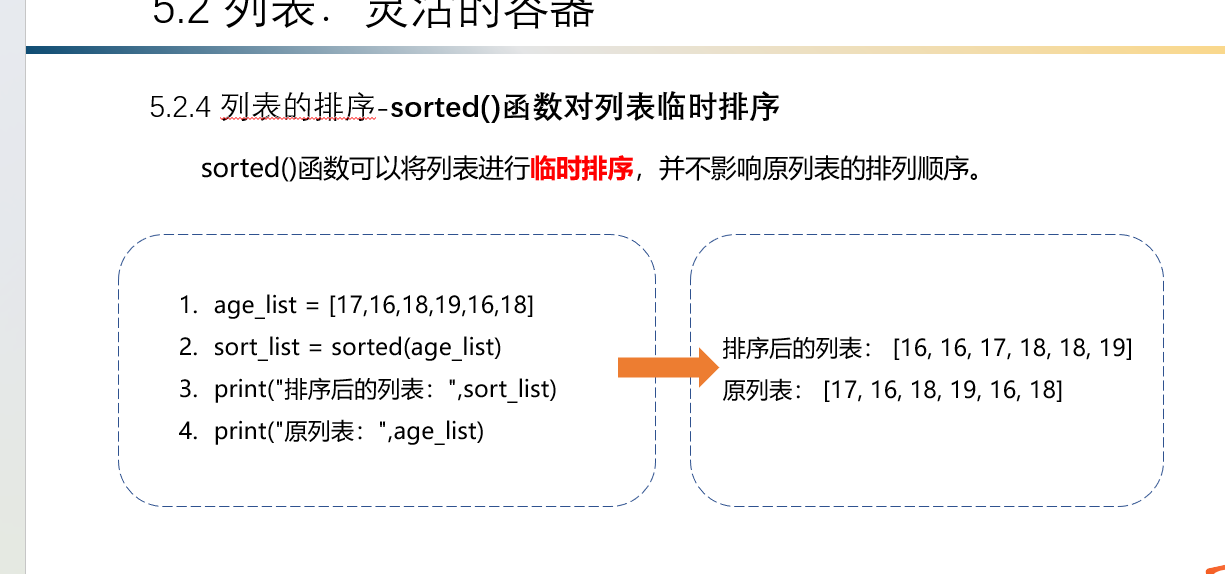
图片包含 图示

描述已自动生成

1. **sort()方法会永久改变列表元素的排列顺序**

图示

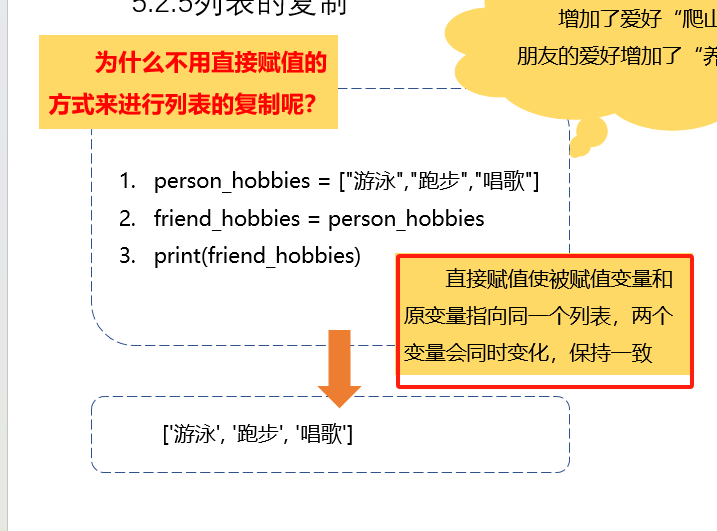
描述已自动生成



**注意 sort永久排序 sorted临时排序 也就是说sort改变序列本身 sorted不改变**

1. 列表的复制可以通过两种方式实现，一种是创建一个包含整个列表的切片，即同时省略起始索引和终止索引（[:]）；另一种方式是用copy()方法

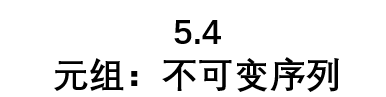
copy\_list = list.copy()

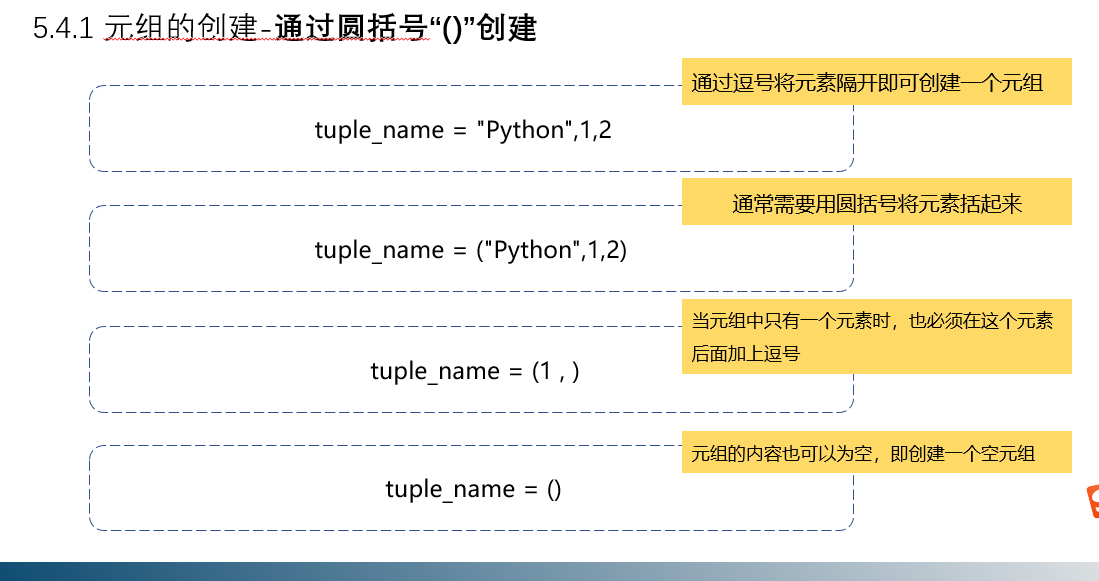
1. 
2. 列表推导式

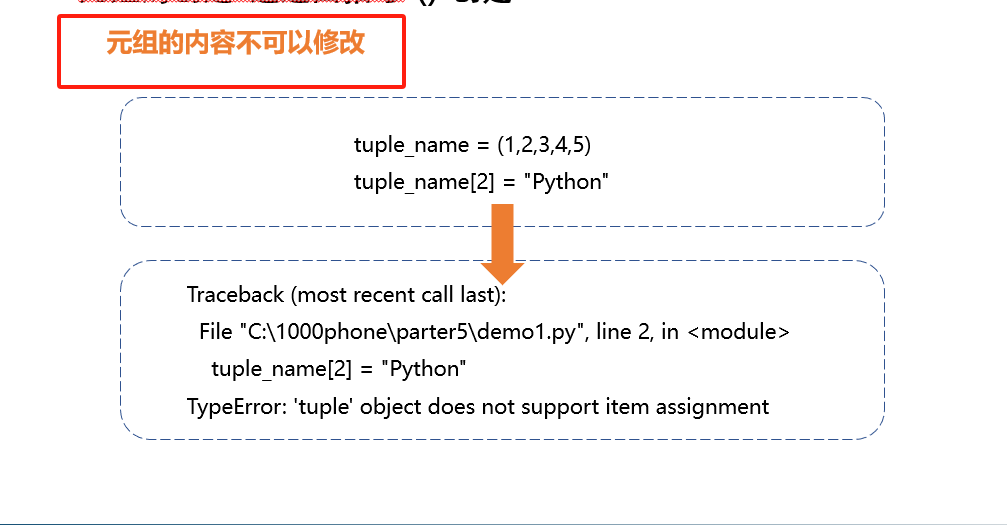
图示

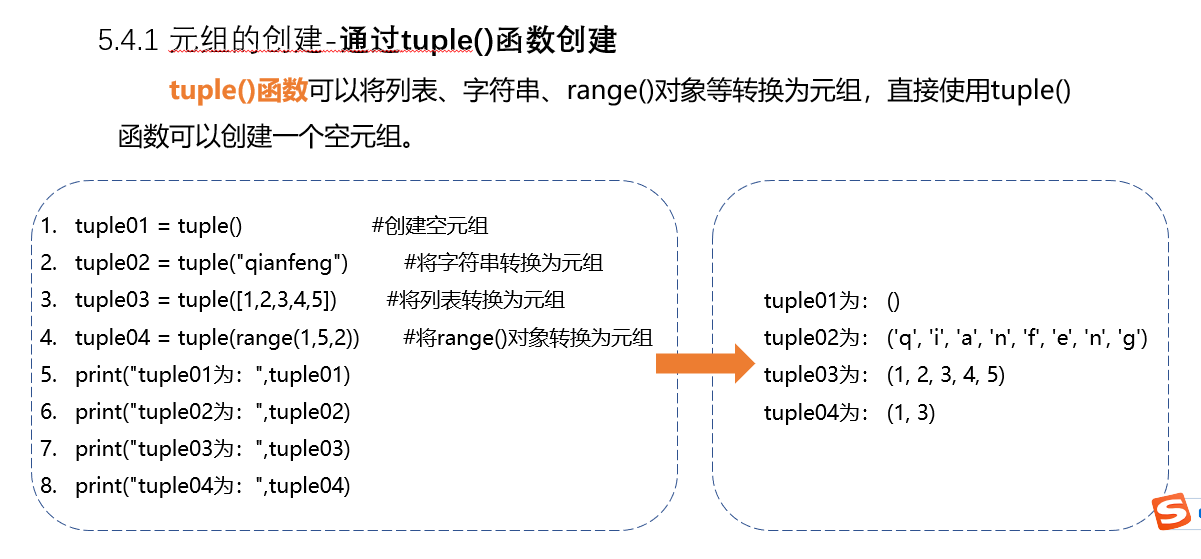
描述已自动生成

1. 列表最大值 最小值 max（list） min(list)



1. 元组是不可变的，创建元组可以通过两种方式：**直接通过圆括号“()”创建**以及**通过tuple()函数创建**。
2. 



1. 通过turtle（）创建元组
2. 
3. 元组的遍历和列表相同，可以通过for循环直接遍历，range()函数索引遍历以及enumerate()函数遍历。
4. 元组**可以在字典（后续讲解）中作为键使用**，而列表则不行。

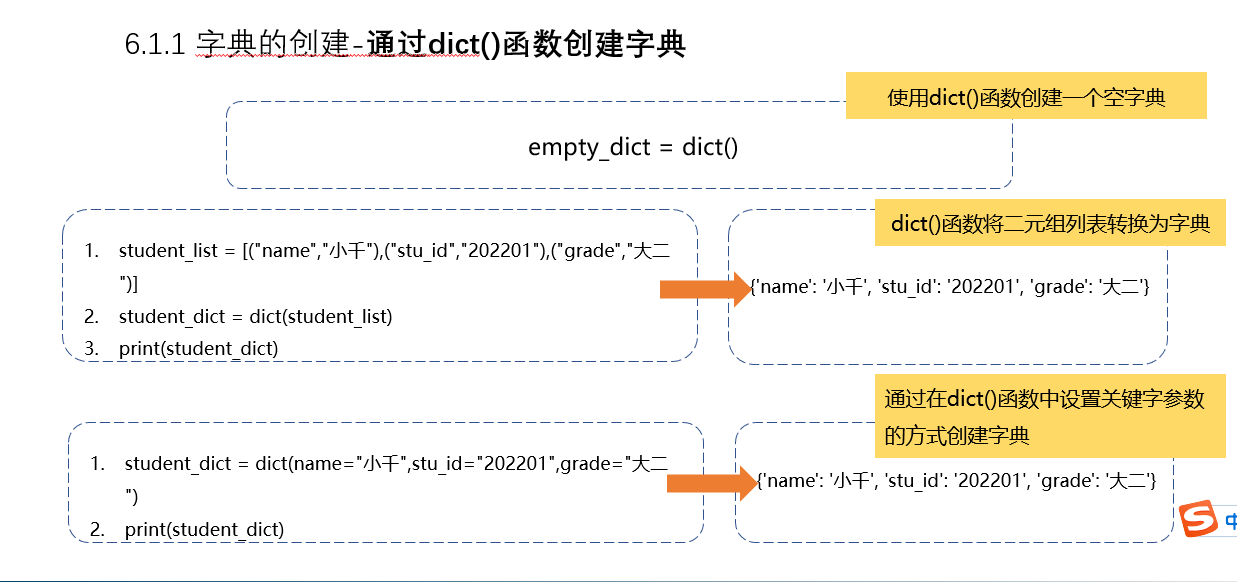
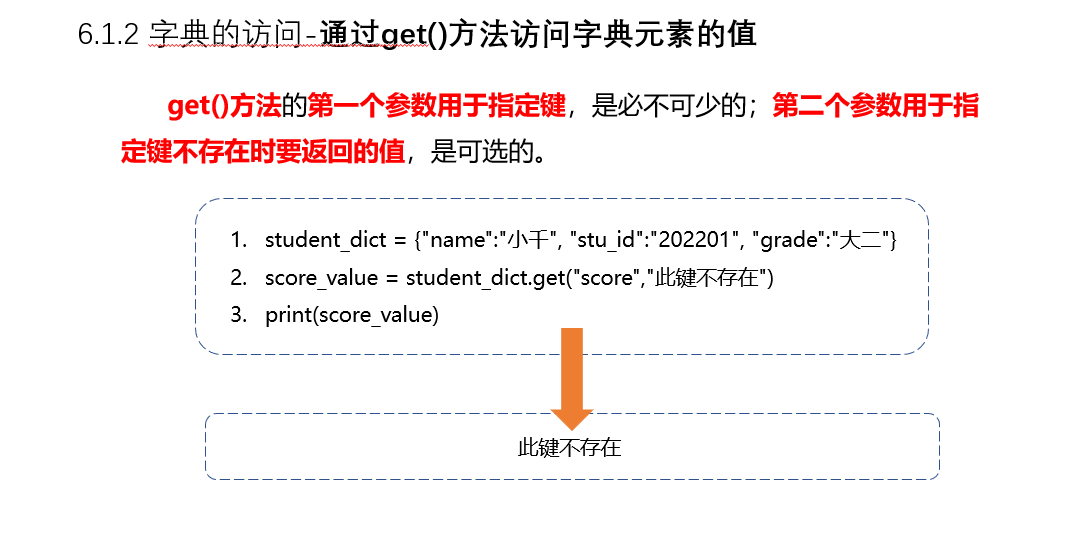
元组**比列表的访问和处理速度快**，如果只需要访问元素，而不需要修改的话，建议使用元组。

1. 元组**可以作为很多内置函数和方法的返回值**（后续讲解）
2. **列表以方括号为标志，其元素可以动态变化；元组以圆括号为标志，是不可变序列，本章以简易购物系统展现了其具体用法。**
3. **列表是 [ ] , 元组是（） , 字典是 { }**
4. 字典以**花括号“{}”**为标志，元素均为键值对形式，键值对形如“**key:value**”，**以英文冒号“:”为标志**，**元素之间以逗号“,”分隔**。字典中的**键需要是不可变的数据类型**，例如数字、字符串和元组，**且不能重复**；值可以是任意数据类型，可以重复。

字典有两种常用的创建方式：

**直接通过花括号“{}”创建**

**通过dict()函数创建**

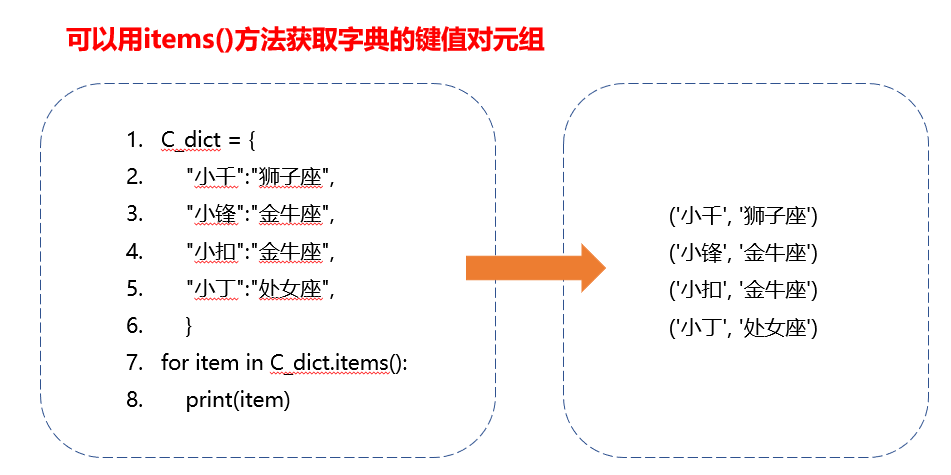
1. 
2. 
3. 字典的遍历方式：

**1遍历所有的键值对**

**2遍历所有的键**

**3 遍历所有的值**

1. **可以用items()方法获取字典的键值对元组**



1. 图示

   描述已自动生成
2. 图示

   低可信度描述已自动生成

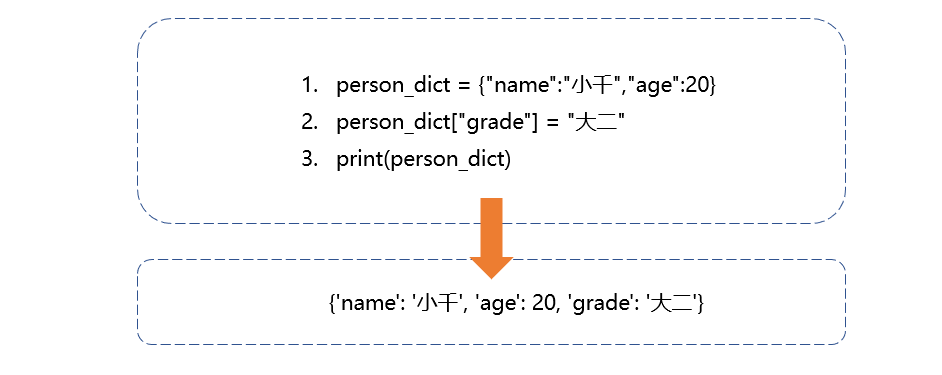
图示

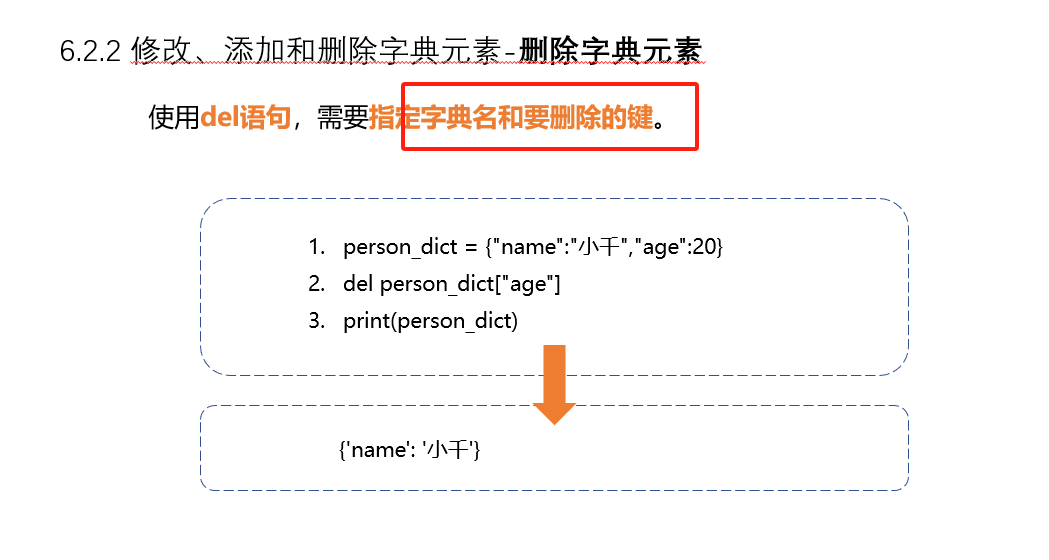
中度可信度描述已自动生成

图示

中度可信度描述已自动生成

1. **可以使用成员运算符（in、not in）来判断某键是否在字典中**
2. **字典元素的修改是通过键来完成的**
3. **如果通过键修改值时，键不存在，则会在字典中添加此键值对**



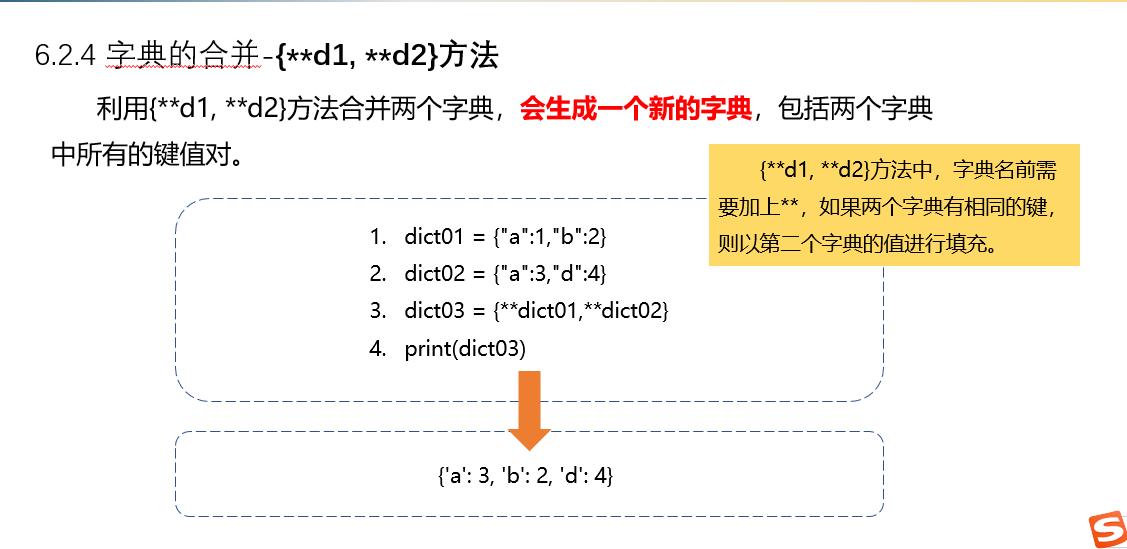
1. 可以通过setdefault()方法添加字典元素，该方法有两个参数，第1个参数表示键，第2个参数表示值。如果键在字典中不存在，那么setdefault()方法会向字典中添加该键，并以第2个参数作为该键的值，没有指定第2个参数的情况下，键的值默认是None。setdefault()方法会返回设置的键对应的值**。**
2. 
3. 图示

   描述已自动生成
4. 文本

   描述已自动生成
5. **Update()将dict02合并到dict01中，如果有相同的键，将会用dict02中键对应的值去更新dict01，如果dict02中有dict01没有的键，将会添加到dict01中。**

图表

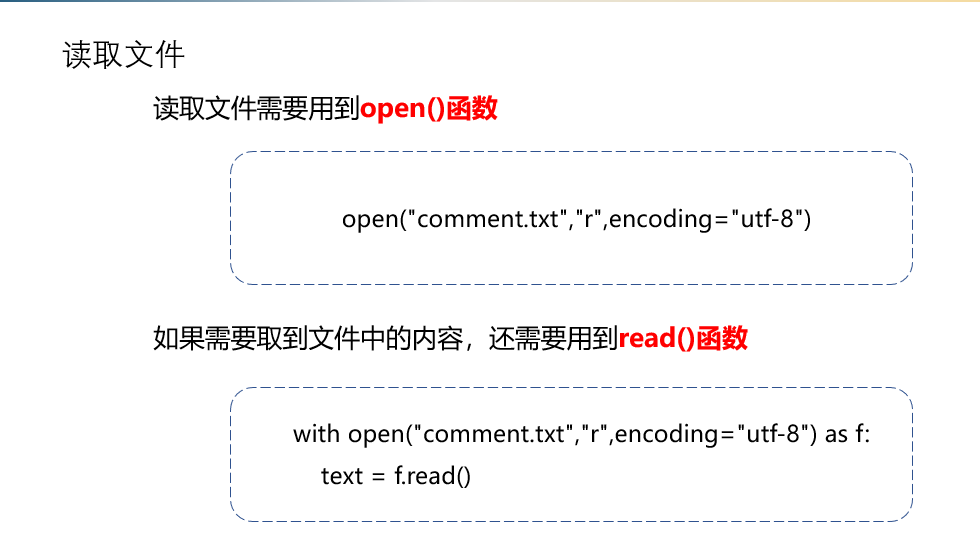
描述已自动生成

1. 
2. **“|”运算符与{\*\*d1, \*\*d2}方法产生的效果类似，会将两个字典合并成一个新的字典，两个字典有相同的键时，以第二个字典的值进行填充。**

图片包含 表格

描述已自动生成

1. 日程表

   低可信度描述已自动生成
2. **Jieba分词lcut形式的函数；通过add\_word()函数可以向分词词典中增加新的词；可以通过del\_word()函数删除分词词典中的词语**
3. **可以通过del\_word()函数删除分词词典中的词语**
4. 
5. **将字典comment\_dict中的键值对按照值的大小顺序进行排列**

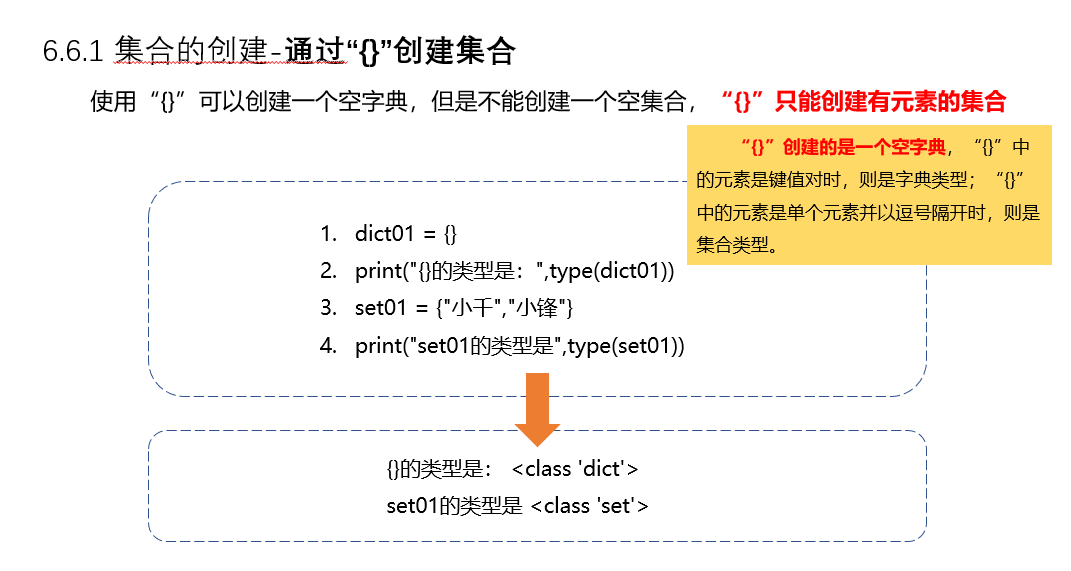
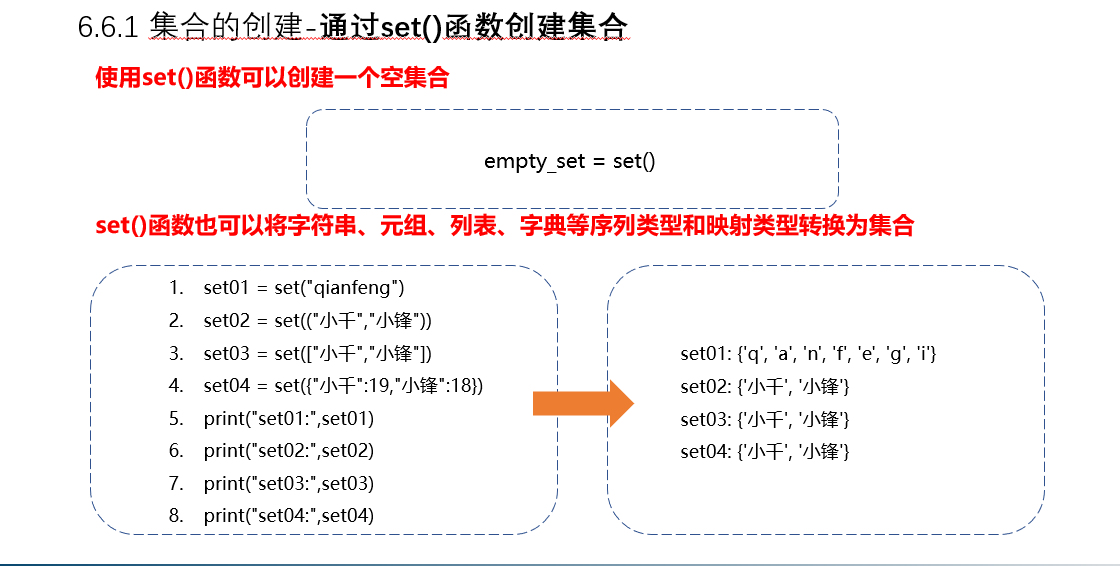
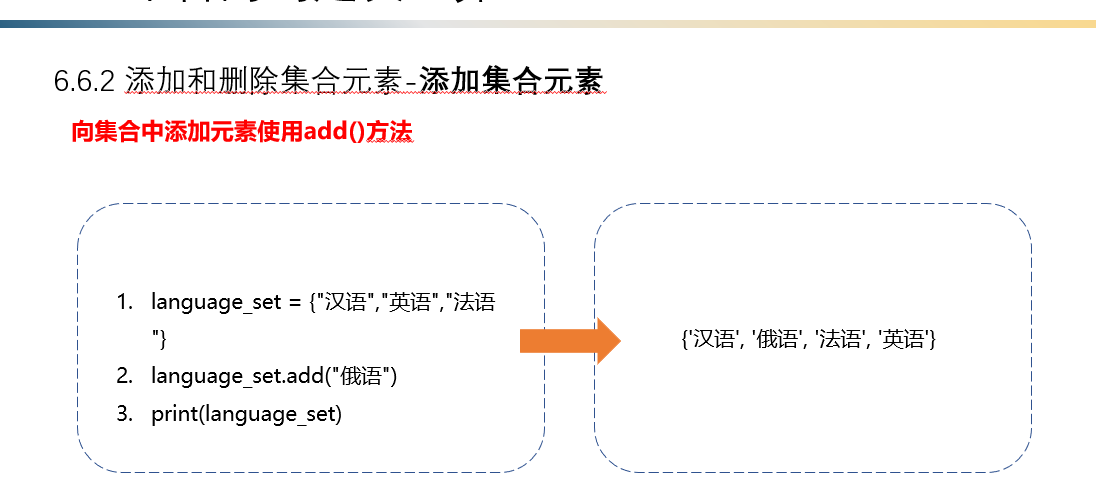
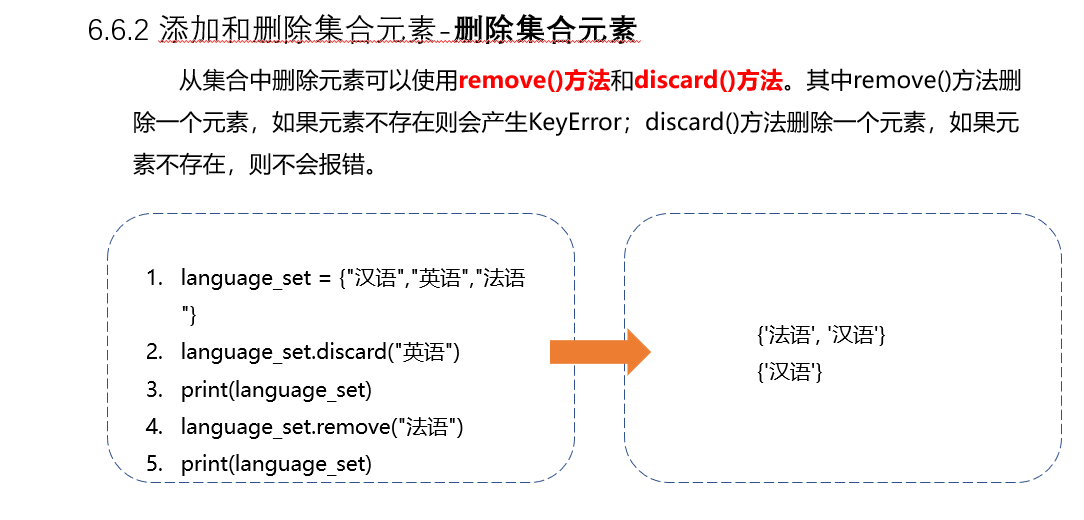
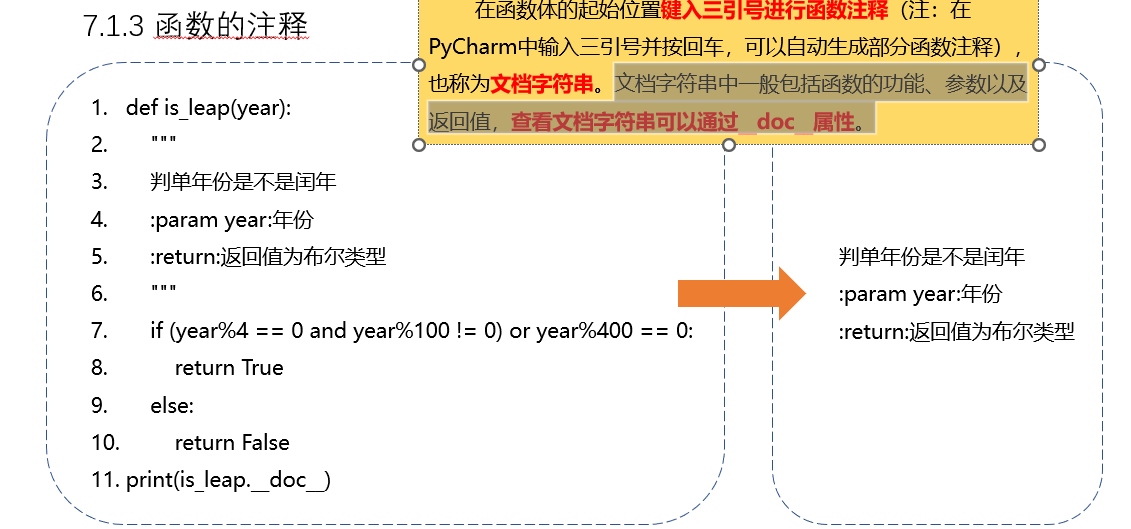
**sort\_list = sorted(comment\_dict.items(),key=lambda x:x[1],reverse=True)**

1. **集合类型由一组无序排列不重复的元素组成。集合中的元素类型只能是不可变数据类型。集合分为可变集合（set）和不可变集合（frozenset），下面只介绍常用的可以进行添加、删除元素操作的可变集合。**

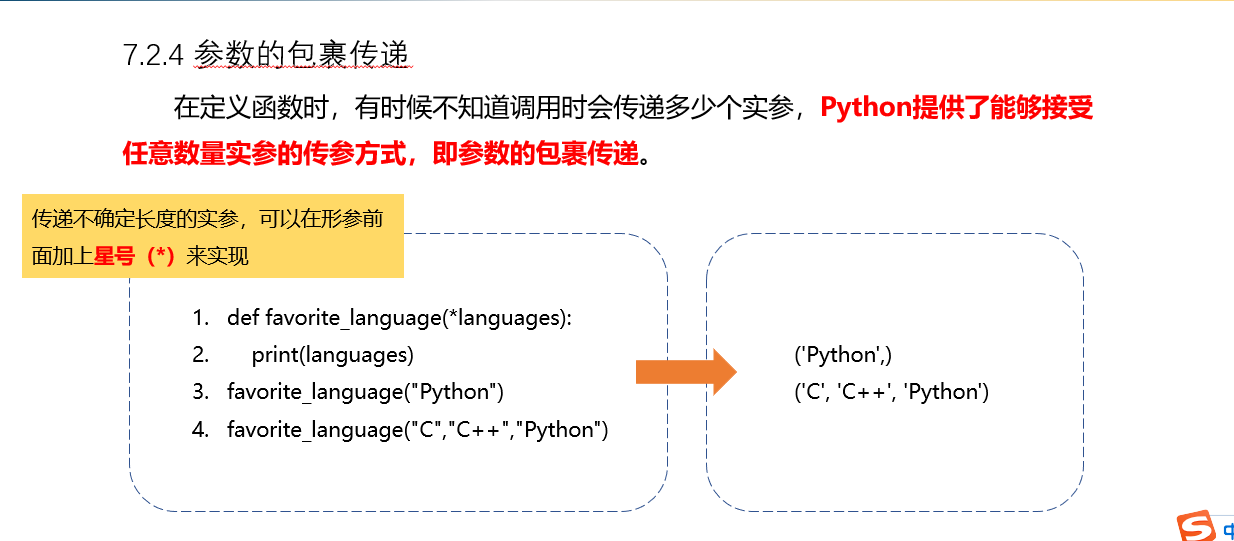
**可变集合由两种创建方式：**

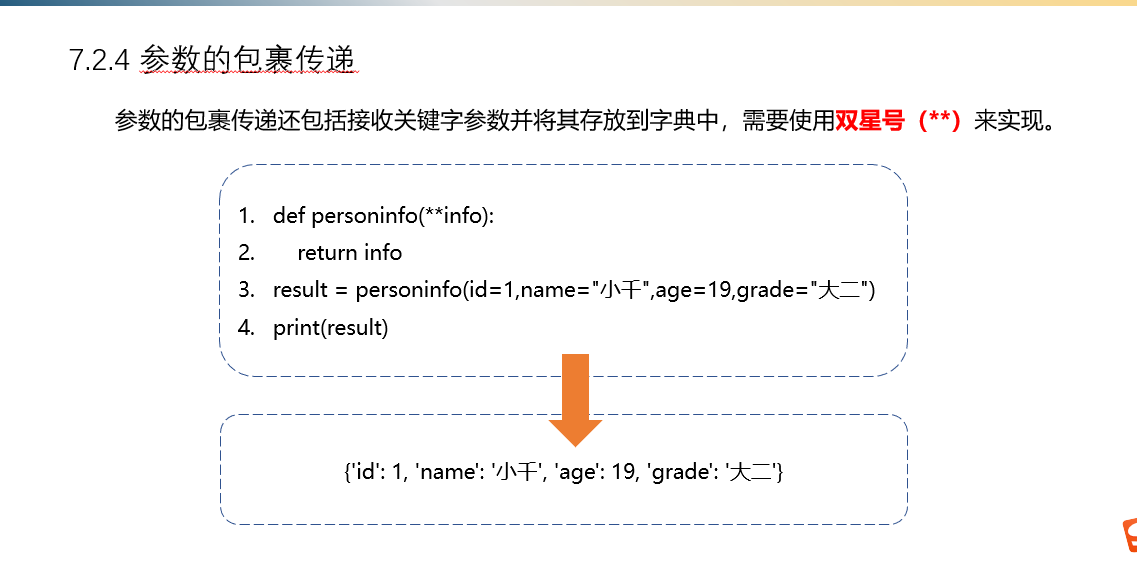
**直接用花括号“{}”创建，元素之间以逗号隔开**

**set()函数创建，其它数据类型也可以通过set()函数转换为集合**

1. 
2. **集合中的元素是不重复的，当创建的集合时输入了重复的元素时，会自动只保留一个**
3. 
4. 
5. 
6. 
7. **当函数的返回值为多个时，返回的多个值是由逗号隔开的，此时有了构成元组的标志，函数的返回值构成了一个元组**
8. **当定义函数却没有写函数体时，代码会高亮显示，表示语法有问题，程序无法顺利运行，此时pass可以充当未完成的代码，使得整个程序能够正常运行。在if、for、while等语句后的语句块，也可以用pass进行占位。**
9. **文档字符串中一般包括函数的功能、参数以及返回值，查看文档字符串可以通过\_\_doc\_\_属性。**
10. **需要注意的是，有默认值的形参必须要在没有默认值的形参的右侧，否则函数将会报错。**
11. **即使形参有默认值，也可以在调用函数时给其传入实参。**
12. **在定义函数时，有时候不知道调用时会传递多少个实参，Python提供了能够接受任意数量实参的传参方式，即参数的包裹传递。**

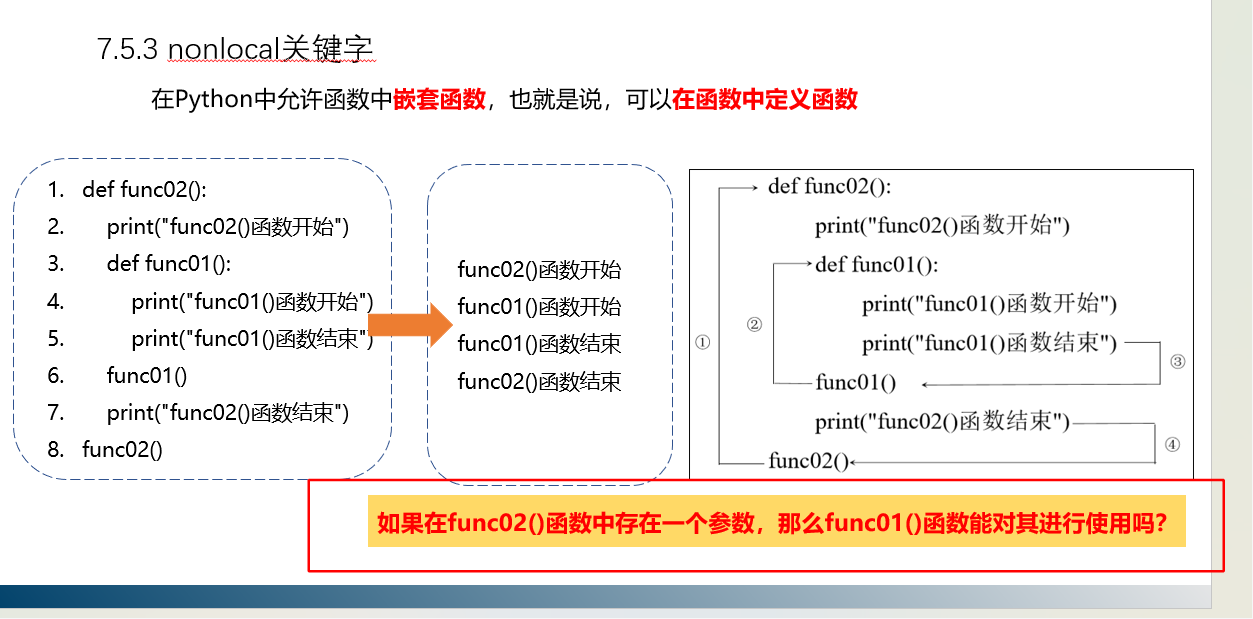
**传递不确定长度的实参，可以在形参前面加上星号（\*）来实现**





1. **形参的排列顺序是先位置形参，然后关键字形参，再是\*元组形参，最后是\*\*字典形参，否则程序就会异常。**
2. **在Python中，有数值、字符串以及元组等不可变的数据类型，也存在列表、字典等可变的数据类型，可变和不可变数据类型在作为函数的实参时，有很大的差别。**
3. **参数的解包裹传递机制为：当实参为元组或列表时，将其拆分，使得元组或列表中的每一个元素对应一个位置形参；当实参为字典时，将字典拆分，使得字典中的每一个键值对作为一个关键字传递给形参。**
4. **根据变量作用域的不同，变量可以分为局部变量 全局变量**
5. **使用关键字global可以声明变量是全局变量；**

**当列表、字典等数据类型作为全局变量时，在函数内部可以对全局变量进行修改，不需要global关键字进行声明。**

1. 
2. **如果需要查看局部变量与全局变量，可以使用globals()与locals()函数获取。**
3. 

**使用nonlocal关键字**

图示

描述已自动生成

1. **1.构造方法\_\_init\_\_()**

**构造方法一般用于类的初始化操作，在创建实例对象时被自动调用和执行。**

**2.self的作用**

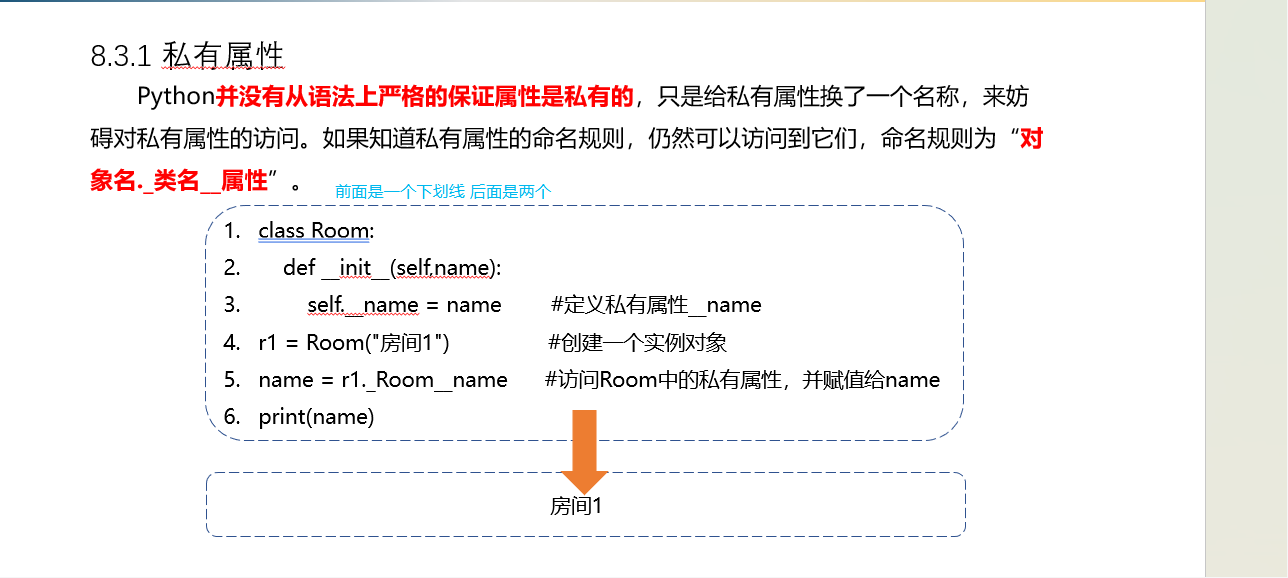
**self的作用是代表将来要创建的实例对象本身，让实例能够访问类中的属性和对象。**

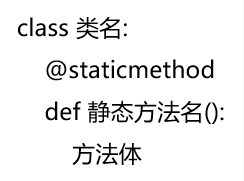
**3.实例方法**

**最少含有一个self参数，用于绑定实例对象的方法称为实例方法，可以被实例对象直接调用。**

1. **在Python中在属性前加两个下划线可以定义私有属性。所谓私有属性，是指不允许外界进行访问，只能在类内访问。**

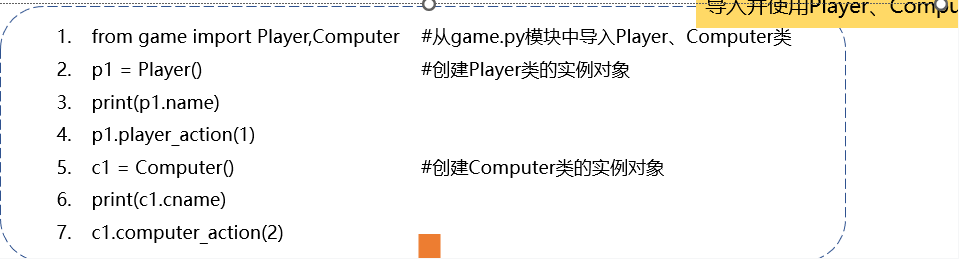
**self.\_\_name = name #定义私有属性\_\_name**

1. 
2. **类属性在类中可以定义类属性，类属性可以通过类名直接进行访问。对于类属性，可以通过“类名.类属性”进行直接访问或修改。通过实例对象也可以访问类属性**
3. **实例对象不能对类属性进行修改，只有类对象能修改类属性的值。**
4. **私有方法的定义与私有属性类似，以两个下划线开头进行定义，在类外不能进行访问。与私有属性类似，Python也没有在语法上严格保证方法是私有的，在类外也能访问私有方法，访问方式是“对象名.\_类名\_\_方法”。**
5. **在类中可以定义类方法，调用类方法可以用类名也可以用实例对象。类方法需要通过装饰器@classmethod进行定义。**文本

   描述已自动生成
6. **静态方法需要用装饰器@staticmethod进行定义，静态方法不同于实例方法，不需要填写形参self；也不同于类方法，不需要填写形参cls。静态方法可以通过类名或实例对象进行调用。**
7. **静态方法可以访问类属性，不能访问实例属性。**
8. **导入自定义模块，需要将模块所在的文件夹设置为根目录**
9. **导入模块中特定的类，需要给类创建对象进行使用。**

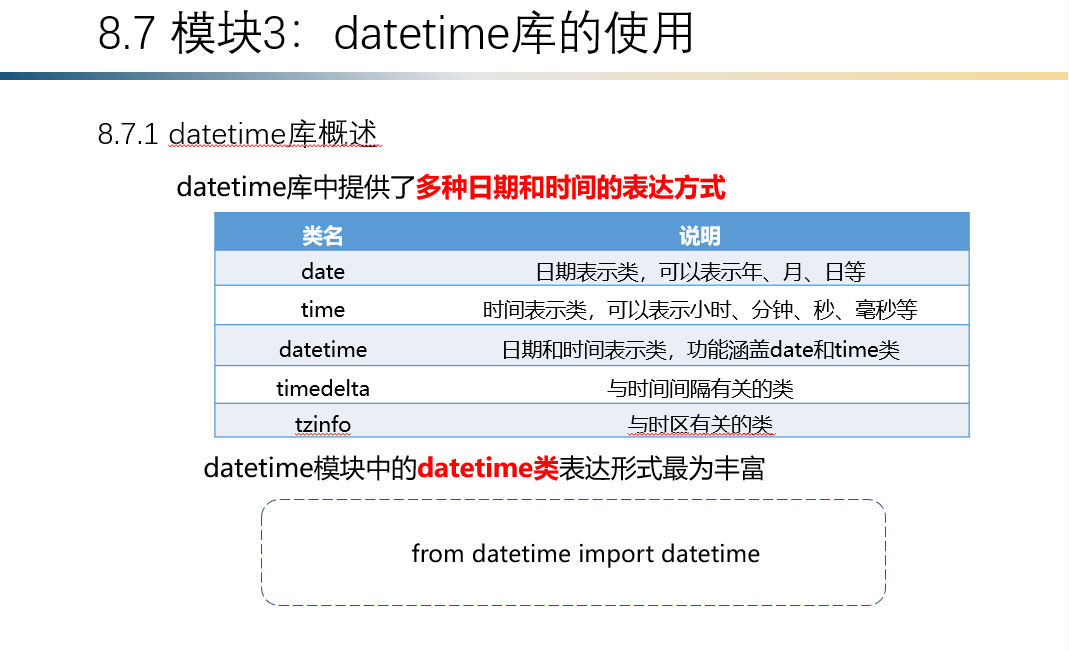


1. **不但可以导入模块中某个特定的类，还可以导入多个特定的类。**



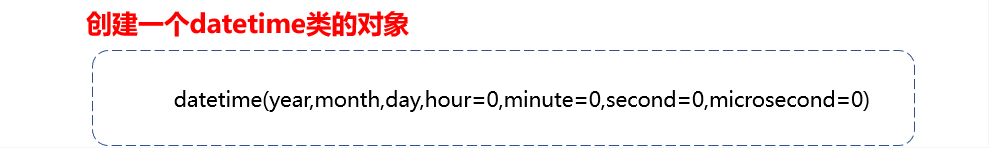
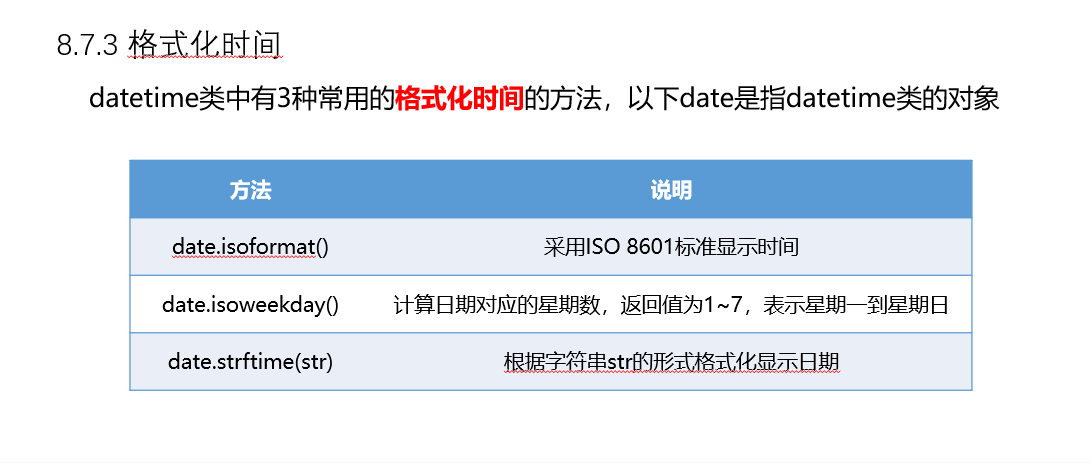
1. **导入模块中的所有类 from 模块名 import \***

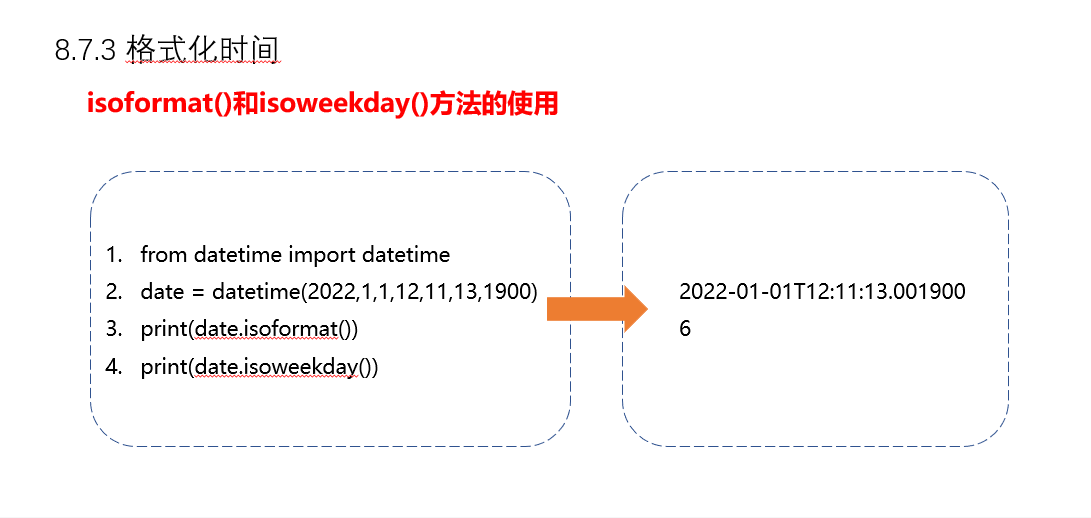
**当导入了模块中的所有类后，使用类只需要通过类名。这种导入方式有两个缺点，第一，这种导入方式不能明确看出程序使用了哪些类；第二，这种导入方式不能确定哪些类属于这个模块，如果导入了多个模块，还会出现与其它变量同名的可能。所以不建议通过这种导入方式导入模块中所有的类。**

1. **导入模块中所有的类可以通过导入整个模块的形式，这种方式在使用类时，需要以“模块.类名”的形式，故而不会与其它模块或者文件中的名称发生冲突。**
2. 
3. **获取当前时间：**

**在datetime类中存在获取当前时间的两个类方法，用类名可以直接调用。**

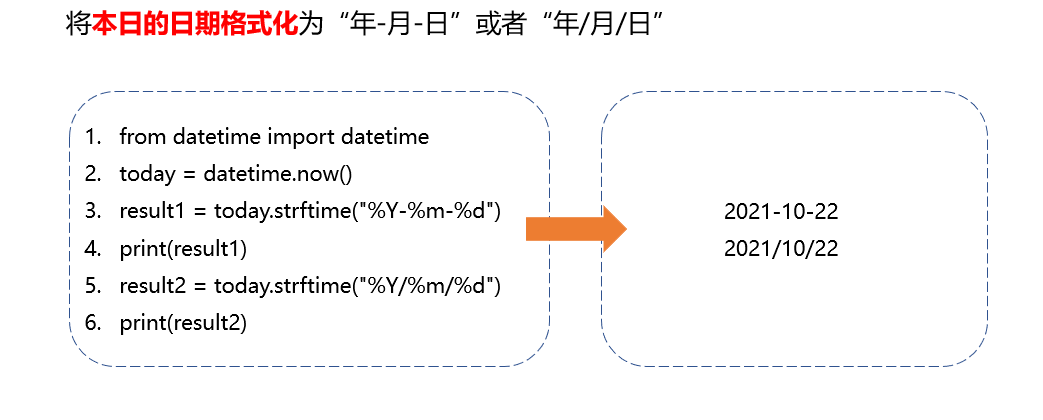
* **datetime.now()：可以获得当前日期和时间**
* **datetime.utcnow()：可以获得当前日期和时间对应的UTC(世界标准时间)**

1. 
2. 

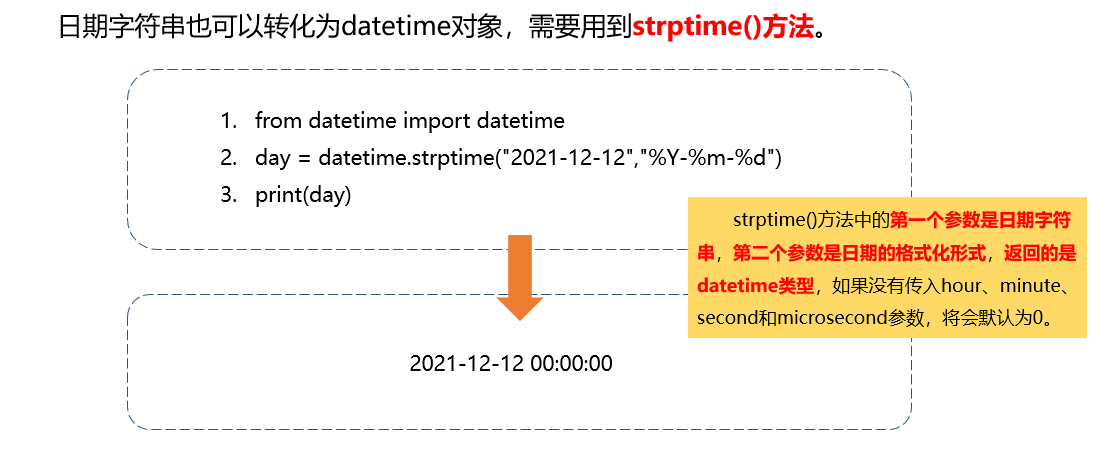


表格

描述已自动生成

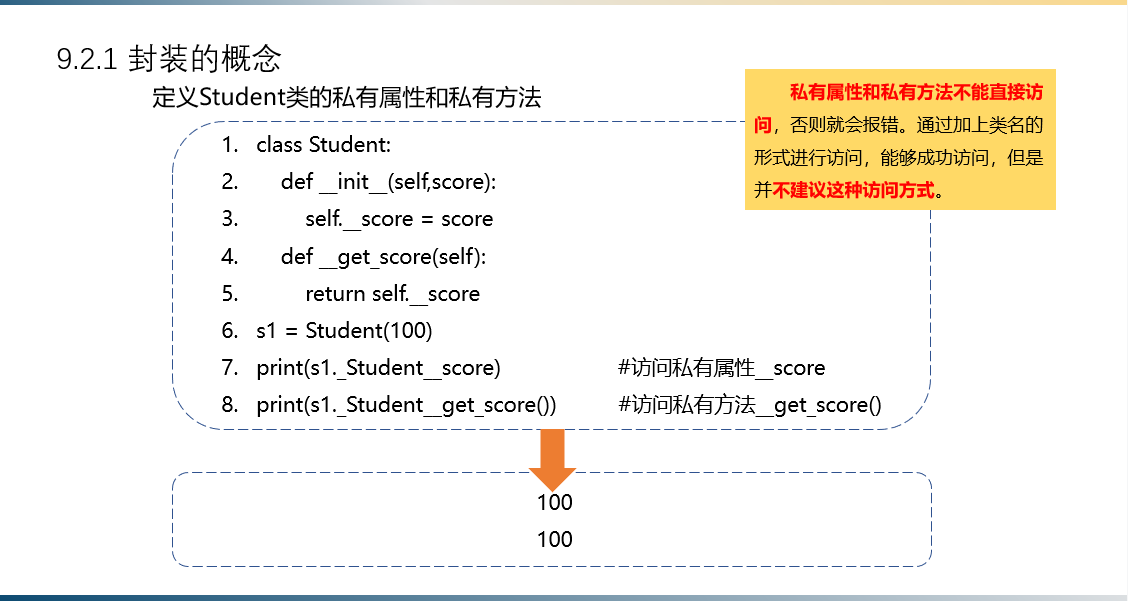
图片包含 图示

描述已自动生成



1. **面向对象程序设计的主要特征可以概括为:**

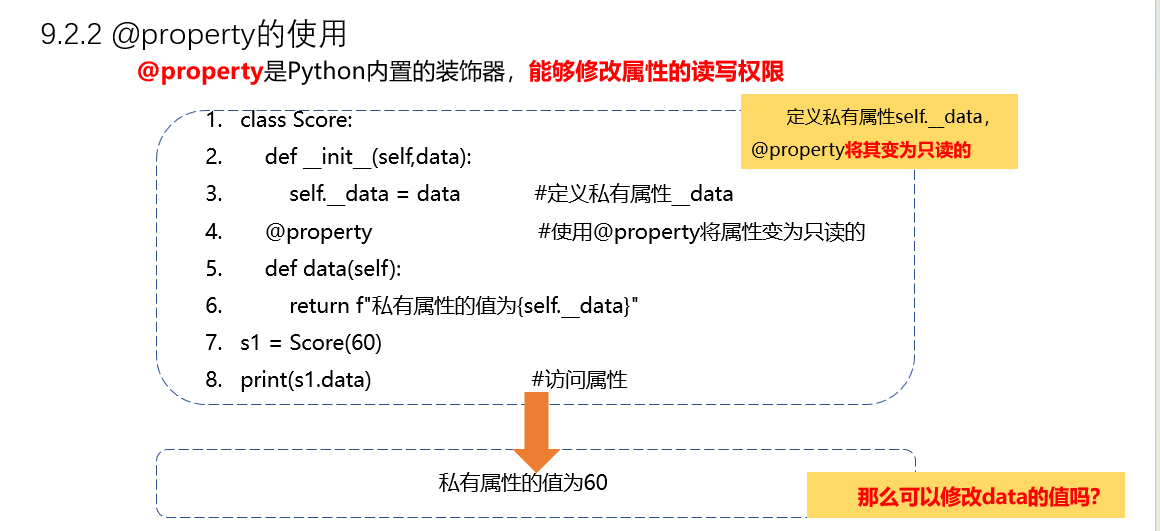
**封装、继承、多态**

1. **封装是面向对象程序设计的核心思想，将对象的属性和行为封装起来，避免了外界直接访问对象而造成的过度依赖，也阻碍了外界修改对象的内部数据而可能引发的难以预见的结果。**
2. **继承主要描述的是类与类之间的关系，是面向对象程序设计中提高重用性的重要措施。继承可以在无需编写原有类的情况下，对原有类的功能进行扩展，既包括原有类的所有属性和行为，又可以定义自身的属性和行为，称为继承了原有类。原有类称为父类或者基类，继承原有类的这个类称为子类或派生类。**
3. **多态是指属性或者行为在基类及其派生类中具有不同的含义或者形式，多态特性使得开发更加科学以及符合人类的思维习惯。**
4. **私有属性和私有方法都属于Python封装的形式，通过在属性和方法前加上两个下划线的方式，阻碍类外对类内属性和方法的访问和修改。但是Python中的私有属性和私有方法不是完全私有化的，是一种通过改变名称而阻碍类外访问的方式，实际上，还是可以通过“\_类名\_\_属性名”或者“\_类名\_\_方法名”的形式对其进行访问。**
5. 
6. **@property是Python内置的装饰器，能够修改属性的读写权限**

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

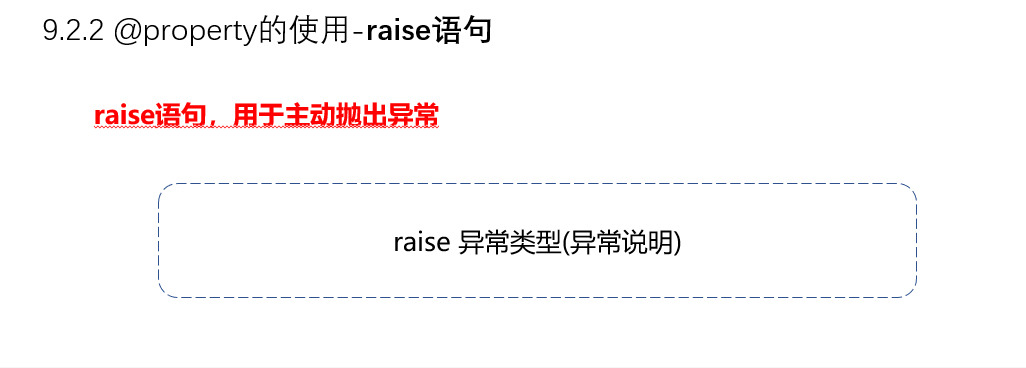
1. **@property是Python内置的装饰器，能够修改属性的读写权限**



1. **如果希望能将Score类中的私有属性变为可修改的，需要通过@property装饰的方法名后加“.setter”**



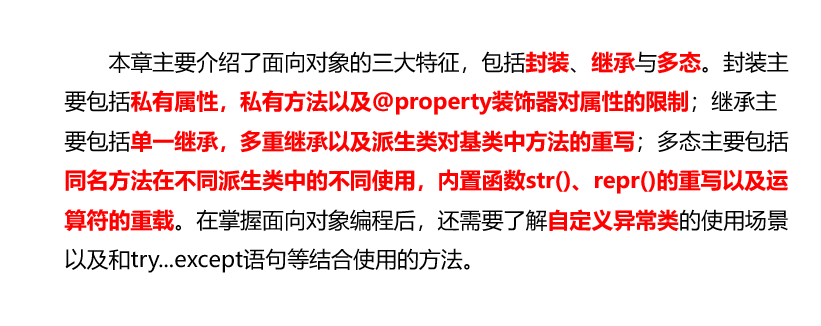
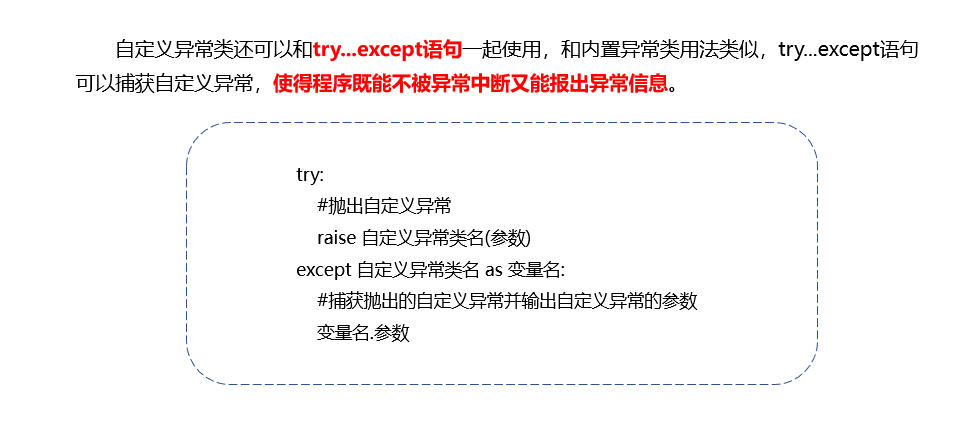
1. **raise语句，用于主动抛出异常**



1. **继承：单一继承、方法重写、多重继承**
2. **单一继承是指定义的派生类只有一个基类**
3. **派生类继承自基类，可以使用基类中的所有公有成员，不能使用基类中的私有成员。派生类可以定义新的属性和方法，从而完成对基类的扩展。注意Python中所有的类都继承自object类，这种继承写法可以省略。**
4. **多重继承是指派生类继承了多个基类**
5. **多态是指基类的同一个方法在不同派生类对象中具有不同的表现和行为**
6. **重写str()和repr()函数，需要在函数名前加两个下划线，函数名后加两个下划线：**

**\_\_str\_\_()函数会将对象转化为人更容易理解的字符串**

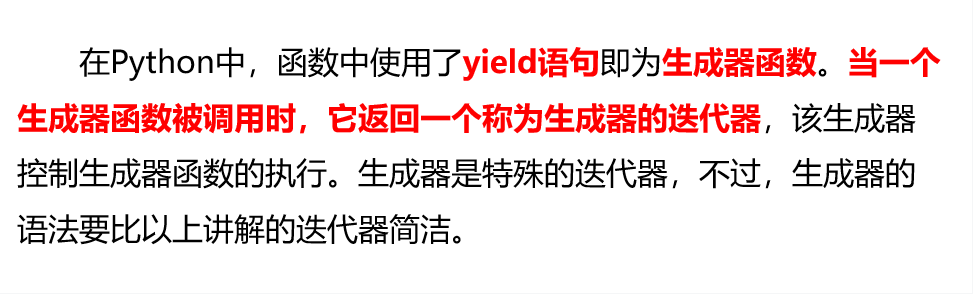
**\_\_repr\_\_()函数会将对象转换为解释器可识别的字符串**

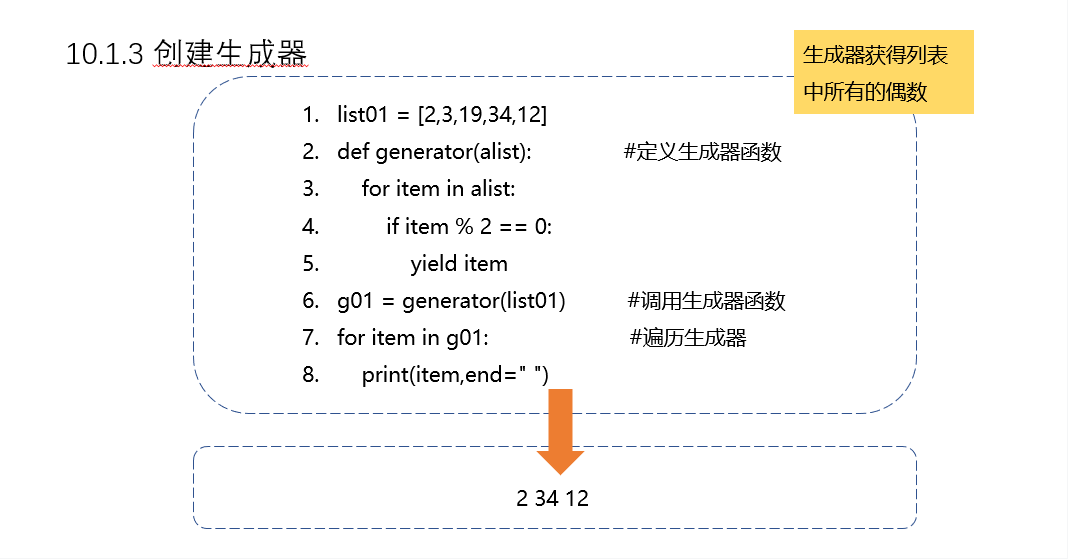
1. **BaseException类是所有异常类的直接或者间接基类，但是自定义的类不能直接继承此类，而是要继承Exception类。自定义异常类一般以Error或者Exception为后缀进行命名。**
2. 

文本

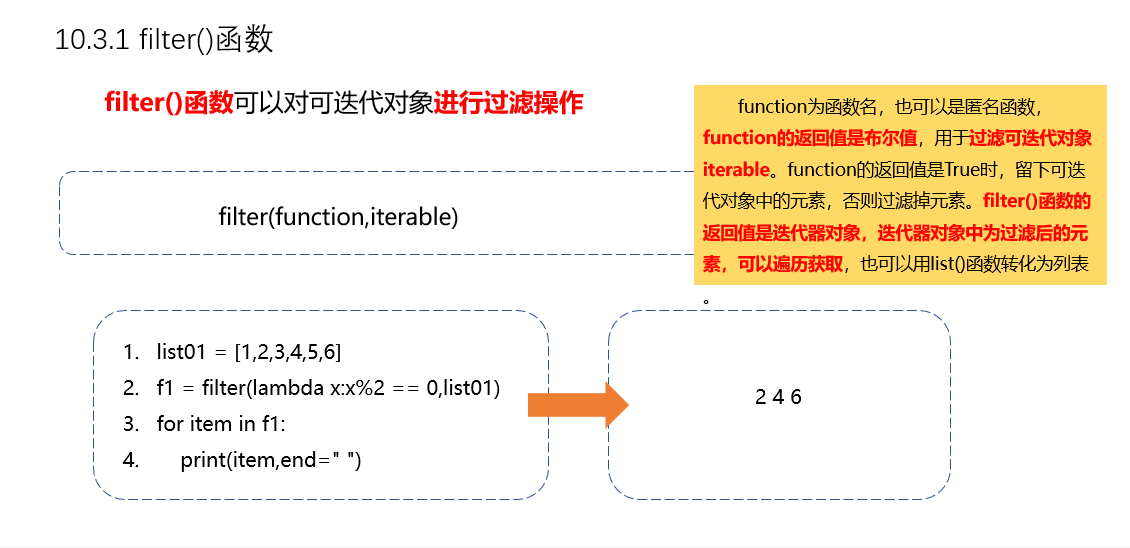
中度可信度描述已自动生成

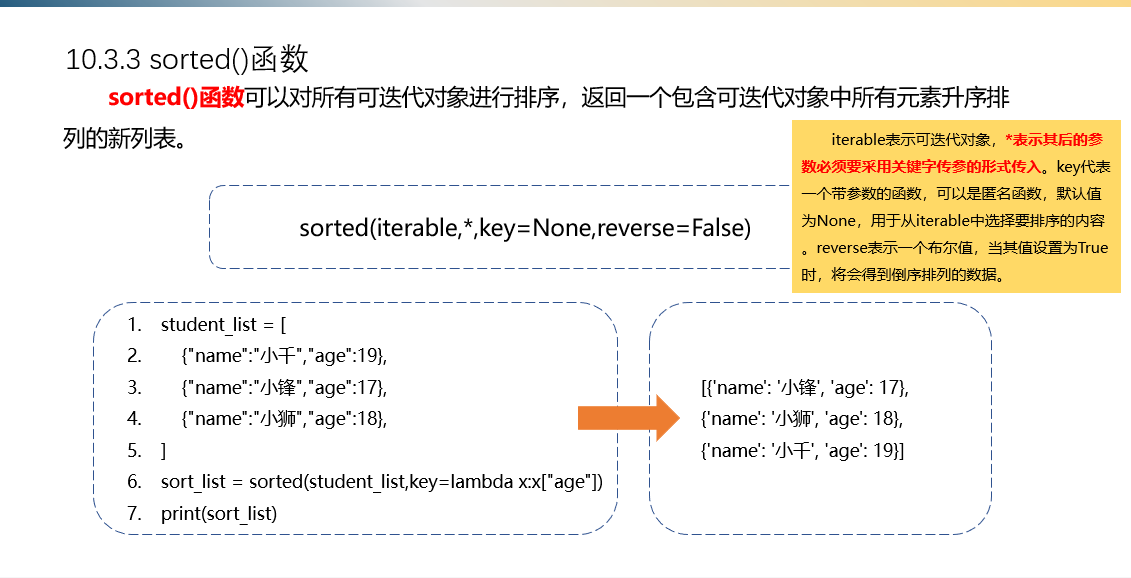
1. **生成器**

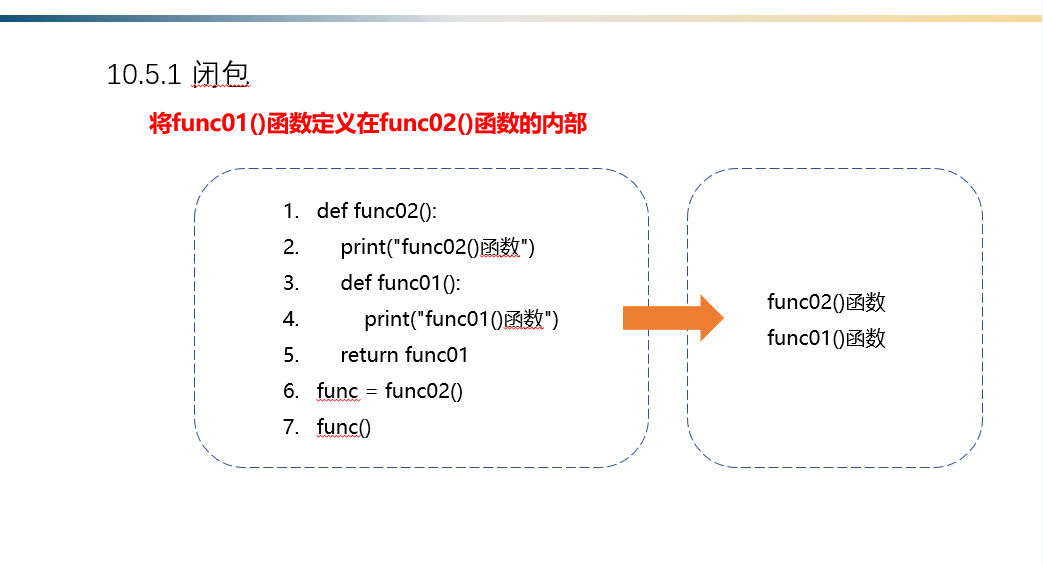


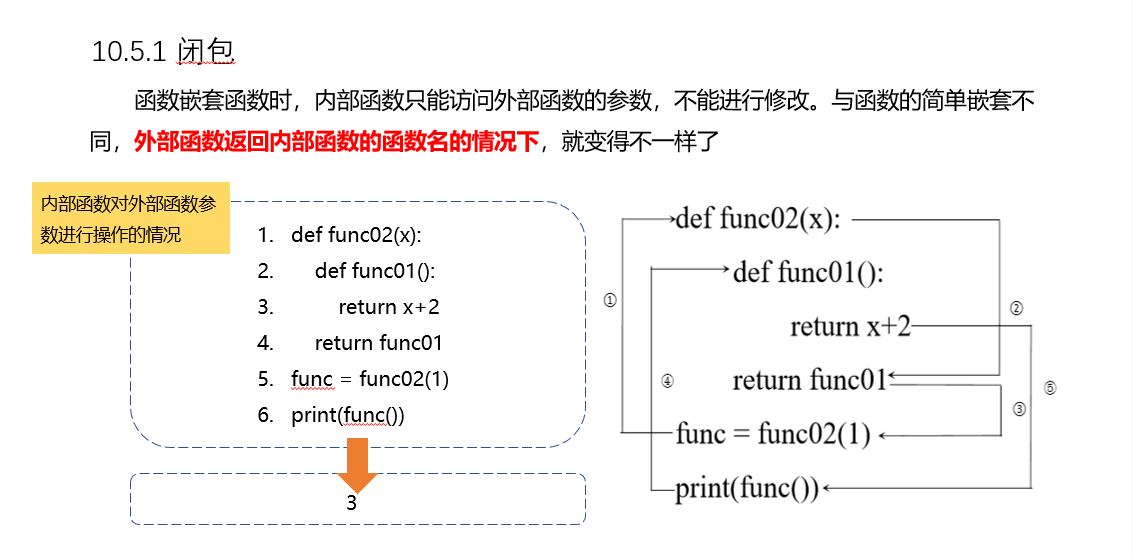


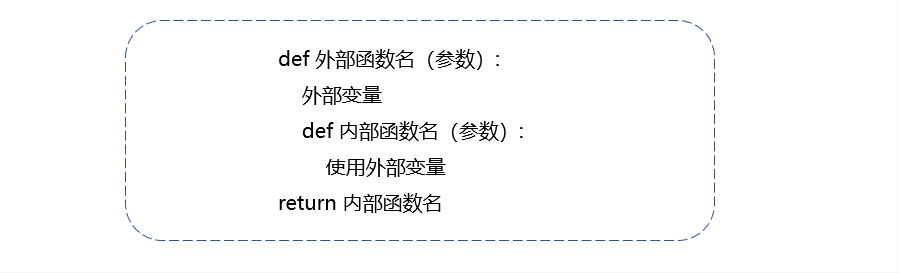
1. 图示

   描述已自动生成
2. **匿名函数**
3. **一个函数可以作为参数传入另一个函数**
4. **匿名函数是指没有名称的、临时使用的微函数，用lambda表达式进行声明。**
5. 
6. 
7. 图示

   描述已自动生成
8. 
9. **返回值是函数的函数属于高阶函数**



1. 
2. **闭包**



**如果内部函数引用了外部函数的变量（包括其参数），并且外部函数返回内部函数名，这种函数架构称为闭包。闭包必须满足以下三个条件：**

* **内部函数的定义嵌套在外部函数中；**
* **内部函数引用外部函数的变量；**
* **外部函数返回内部函数名。**

1. **装饰器可以让其它函数在不做修改的前提下增加额外功能。装饰器的本质是一个嵌套函数，外部函数的参数是需要被装饰的函数名，内部函数用于增加被装饰函数的新功能。@符号可以将装饰器函数与被装饰函数联系起来。**

