一、基础知识

1. python创始人：荷兰人桂多.范罗苏姆
2. 开源的 面向对象的 动态 强类型 胶水类型
3. 编码规范：pep8(移进 空行 空格)
4. Python编译器：CPython iPython PyPy Jpthon ironPython
5. Python关键字：①async

②await

Python标识符：①区分大小写 sxt和SXT不一样

②第一个字符必须是字母，下划线。其后的字符可以是字母，数字，下划线

③不能使用关键字，比如if，or，while

④以下划线开头或结尾的名称通常具有特殊的含义，避免使用

1. 理解变量赋值中的引用的概念:

(参考https://www.cnblogs.com/f-ck-need-u/p/10123145.html)

1. Python中一切皆对象：①标识 ②类型 ③值

二、基础知识(八大数据类型)

1. Number(不可变)：整数，浮点数，复数

2. String(不可变)：①常见的转义字符：\t(横向制表符) \n(换行符) \\(反斜杠符号)

\’(单引号) \”(双引号) \r(回车)

②切片(包头不包尾)：str = “runoob” str[:] = str[0:] = str[:6] = ‘runoob’

str[:3] = str[-6:-3] = ‘run’ str[-1:-4:-1] = ‘boo’(其中第三个值只能为1或-1，代表从

左往右或者从右往左对字符串进行切片)

③字符串运算：+(字符串连接) \*(重复输出字符串) in/not in(字面意思) r/R(使\n等转义字符能以字符串形式正确输出)

④字符串格式化：%format f-string

(参考https://www.cnblogs.com/lvcm/p/8859225.html)

⑤字符串的常见方法：str.join(seq)--用str把seq序列中的字符串分隔开

len()--字符串长度

count(str,start = 0,end = len(string))--指定范围内子串str出现的次数

capitalize() 字符串的第一个字母变成大写,其他字母变小写 title() upper() 小写字母转为大写字母 max(str)、min(str)

replace(old,new,max)

1. Boolean(True Flase):
2. None空值
3. List(列表，可变) extend(...)，insert(...)没有返回值！！只是在原列表基础上扩展！

insert(index, obj) 其中index的值就是obj被插入后处于的位置

pop([index=-1])

remove(obj)若obj不存在则会报错

sort(cmp=None, key=None, reverse=False)



1. Tuple(元组，不可变)：①单元素必须有，比如(“xyz”)是字符串 (“xyz”,)是元组

②元组没有pop方法，列表、字典、集合都有！！！ 元组和字典没有remove方法！其余有！ ③元组不能直接用tup1[0] = 100这种方式来修改元素值，但可以通过’+’的拼接方式添加值，删除只能del把整个元组给删掉

tuple( iterable ) ---把列表变为元组 x,y,z = “202” -> x=2 \*x,y,z = “202” -> x=[‘2’]

1. Dict(字典，可变)：①key唯一(如果重复最后的一个键值对会替换前面的，值不需要唯一)且不可变，如字符串，数字或元组，值可以取任何数据类型

dict = {} --- 可以直接通过dict['School'] = "RUNOOB"来添加键值对，此时dict = {'School'：RUNOOB}

②dict.clear():删除字典内所有元素 .copy():返回一个字典的浅复制

.fromkeys(seq[, value]):创建一个字典，seq(可以是元组或者列表)中元素做字典的键，value 为字典所有键对应的初始值



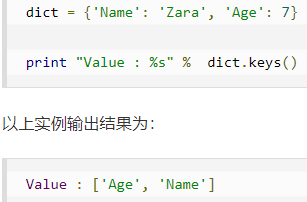
.get(key, default=None)：返回指定键的值

key in dict:判断key在不在字典内

.items()：以列表返回可遍历的(键, 值) 元组数组

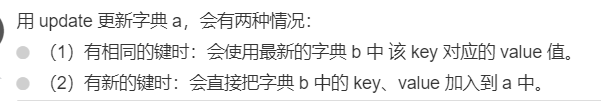


.key()：返回一个字典所有的键 .values()：返回字典中的所有值



.setdefault(key, default=None)：和 get()方法 类似, 如果键不存在于字典中，将会添加键并将值设为默认值。

.update(dict2)



.pop(key, default=None) del语句和pop()作用相同，pop()有返回值。

.popitem() 返回并删除字典中的最后一对键和值。

如果字典已经为空，却调用了此方法，就报出 KeyError 异常。

1. 集合：①特性：去重！

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典 parame = {value01,value02,...}

或者set(value) --- set只能接受一个参数，也就是说value = ‘fdfsd’或者(1,2,3,4)

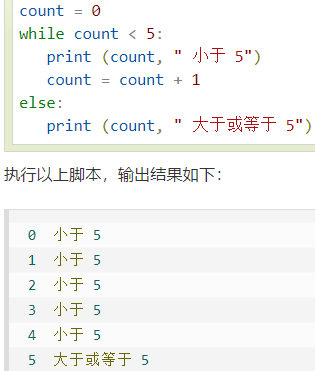
②常见方法：add、update(也是加)、remove、pop等

set1.difference(set2) set1.difference\_update(set2) 两函数的作用都是做减法，区别是第一个函数返回一个新集合，第二个函数直接在set1身上进行修改，相同原理的还有求无重复并集，求交集：set1.symmetric\_difference(set2)与它的update set1.intersection\_update(set2, set3 ... etc)与它的update

isdisjoint() 方法用于判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False

1. 条件控制
2. 循环：break，continue 没有do...while！！！

在点出一个比较少见的while..else: while...else在条件语句为 false 时执行 else 的语句块(for...else同理)，举例：

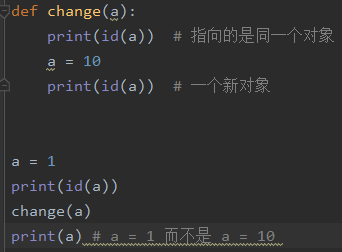


1. 函数
2. 定义
3. pass
4. 参数：①不定长参数：\*args(将参数打包成tuple给函数体调用，存放所有未命名的变量参数) \*\*kwargs(打包关键字参数成dict给函数体调用，因此格式应该为k=v)

要明确，def printinfo(\*vartuple)的返回值一定是元组类型(哪怕只有一个参数) 而def printinfo( arg1, \*vartuple)在一个参数时可以返回arg1的类型

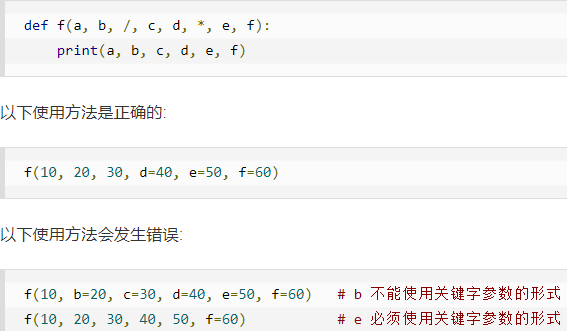
②函数的参数传递：

1. 传不可变对象:值传递，如 fun(a)，传递的只是a的值，没有影响a对象本身。如果在 fun(a)内部修改a的值，则是新生成来一个 a

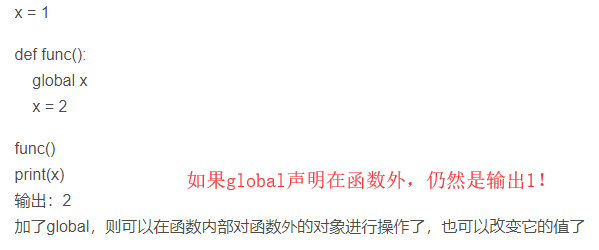


1. 引用传递，如 fun(la)，则是将la真正的传过去，修改后fun外部的la也会受影响

③强制位置参数：函数参数列表中/前的参数必须使用指定位置参数，\*后的参数只能使用关键字参数的形式，举例：



1. 就近原则，yyds！同时注意！Python中定义函数时，若想在函数内部对函数外的变量进行操作，就需要在函数内部声明其为global。



五、异常

1. 异常和错误的区别

2. try：{执行代码} expect：{发生异常时执行的代码} else：{没有异常时执行的代码} finally：{无论如何都会执行的代码：}

3. 使用raise触发异常 raise + Exception

4. 用户可以自定义自己的异常

六、面向对象编程及其他重点内容

1. ①实例化(init new)：self不是python关键字！

②私有方法、属性：无法在类外直接访问，切记是\_\_而不是\_

③实例方法：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

类方法：使用装饰器@classmethod。第一个参数必须是当前类对象，该参数名一般约定为“cls”，通过它来传递类的属性和方法（不能传实例的属性和方法）

这里需要注意，如果存在类的继承，那类方法获取的类是类树上最底层的类。

④静态方法：使用装饰器@staticmethod。参数随意，没有“self”和“cls”参数，但是方法体中不能使用类或实例的任何属性和方法

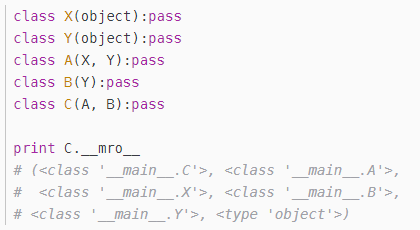
可用类名.方法名或实例.方法名调用静态方法或类方法

1. 三大特性：

①封装(定义一个类本身就是一种封装)

②继承:

1. 多继承问题：为了方便且快速地看清继承关系和顺序，可以用\_\_mro\_\_方法来获取这个类的调用顺序，举例：



\_\_mro\_\_如何返回值详见https://www.cnblogs.com/whatisfantasy/p/6046991.html中C3 MRO部分的讲解

1. 组合：可以形象的理解为类的属性可以是其他类

③多态：

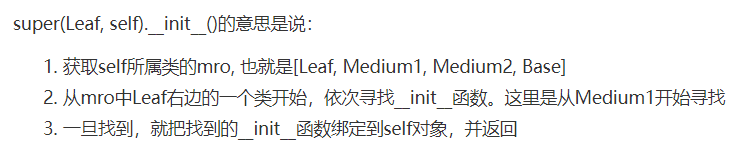
A.多态是指同一个方法调用由于对象不同可能会产生不同的行为。注意！

1.多态是方法的多态，属性没有多态。

2.多态的存在有2个必要条件：继承、方法重写。

B.钻石继承问题(菱形继承问题)

把类.方法名改为super().方法名，super的调用顺序和原则参考如下：



即super不在单纯是找父类，而是根据\_\_mro\_\_的顺序，每遇到一个super往右边找一个类

具体例题详见课件面向对象(3)16 17页！！！！！！！！！

1. 对象三要素：

①value：对象的值

②type：标识对象的类型

type() 和 isinstance() 函数的区别:

isinstance(object, classinfo)判断参数 object 是否是 classinfo 类或者其子类的实例 type()只能判断是否是该类本身的实例

③id: 身份，唯一标识一个对象

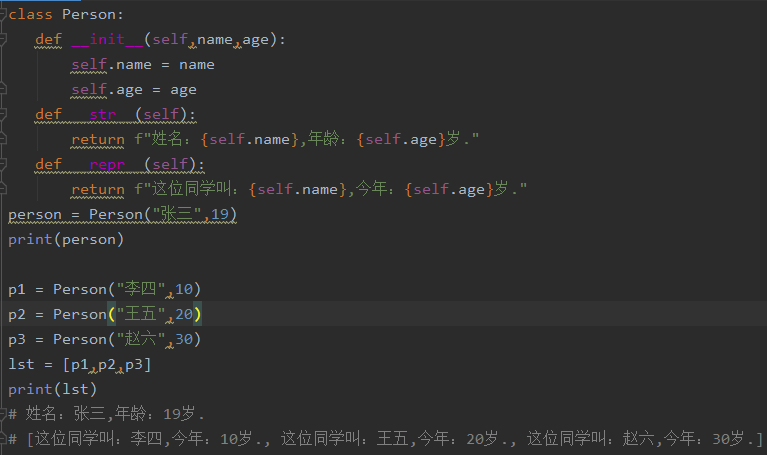
1. 魔法函数：①定义和基本信息：python中定义的以双下划线开头结尾的函数

可以定制类的特性，魔法函数不是object类的一个方法，是在自定义的类里可以自己添加任意一个魔法函数

②常用魔法函数及实例：

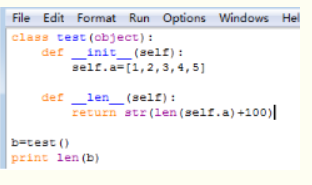
1. \_\_init\_\_ : 构造函数，在生成对象时调用
2. \_\_del\_\_ : 析构函数，释放对象时使用
3. \_\_str\_\_ \_\_repr\_\_:通常在打印对象的时候，打印的总是对象的地址，为了在打印的时候打印出具体的信息，我们会采用在类中重写str或repr方法

举例：

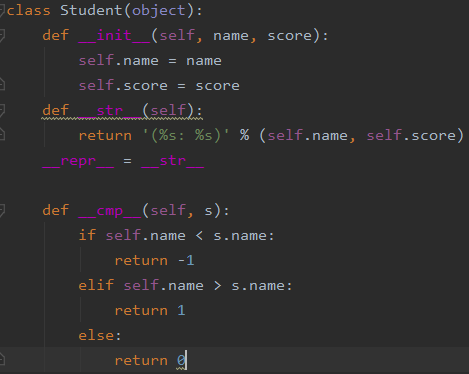


两者的区别：\_\_str\_\_()，用于直接打印对象本身，\_\_repr\_\_()，用于打印列表里面包含对象；\_\_str\_\_是面向用户的，而\_\_repr\_\_面向开发者

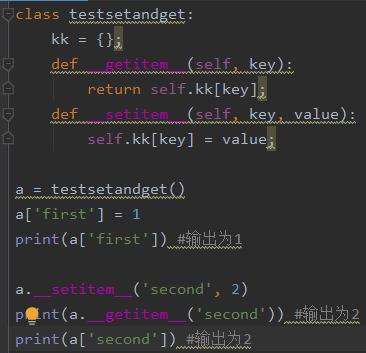
1. \_\_len\_\_:len调用后会调用对象的\_\_len\_\_函数,即可以len(对象名)输出类里\_\_len\_\_函数的东西，需要注意是该函数要求我们返回的值必须为int，否则会报错，举例：



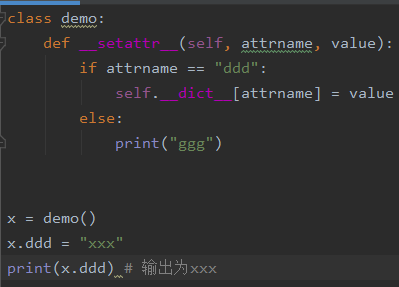
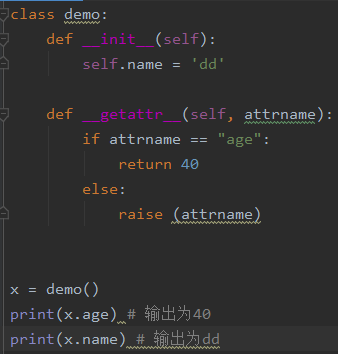
1. \_\_cmp\_\_:比较运算，比较好理解，给个例子即可：



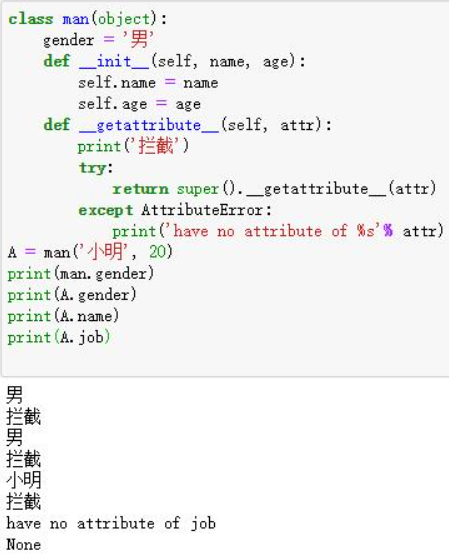
1. \_\_getitem\_\_ \_\_setitem\_\_:如果在类中定义了\_\_getitem\_\_()方法，那么他的实例对象（假设为P）就可以这样P[key]取值。当实例对象做P[key]运算时，就会调用类中的\_\_getitem\_\_()方法,\_\_setitem\_\_同理，可以通过P[key] = v来赋值,举例：



同理\_\_getattr\_\_ \_\_setattr\_\_:attr是属性的意思，我们知道python类中的公有属性可以直接通过对象名.属性名进行访问，而这两个魔法函数可以对类中还没有定义的属性进行操作，举例：



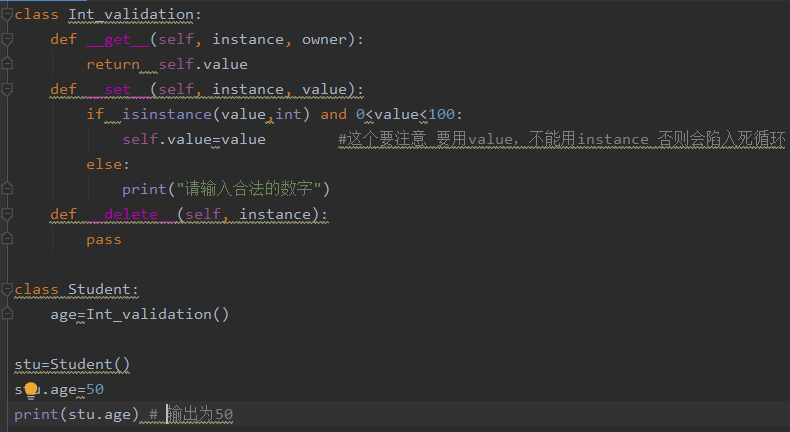
\_\_getattribute\_\_：该方法可以拦截对对象属性的所有访问企图，当属性被访问时，自动调用该方法，举例：



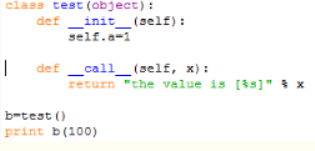
\_\_get\_\_ \_\_set\_\_：举例：首先我们要把共同的校验封装在一个类里也就是 Int\_validation 类，

重写两个魔法函数 （get 和set）

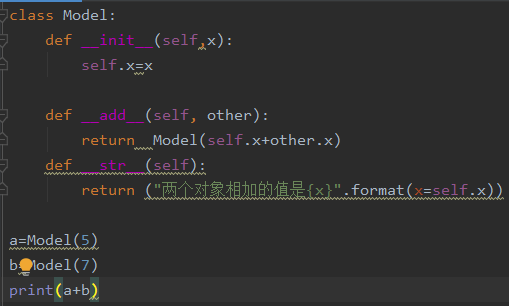
age=Int\_validation() 这个是关键，age 本来是个对象，放在类里当作了Student类属性



1. \_\_call\_\_: 对象通过提供\_\_call\_\_()方法可以模拟函数的行为，如果一个对象提供了该方法，就可以像函数一样使用它：对象名(参数)，举例：



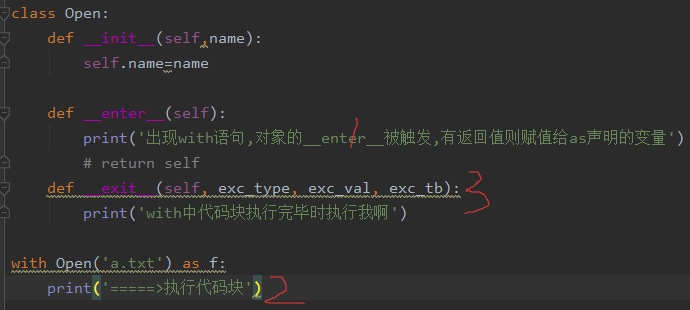
1. 各种运算，这里以\_\_add\_\_(加运算)为例：即可以把两个类当做运算数直接进行加操作(等其他操作)，举例：



\_\_sub\_\_: 减运算 \_\_mul\_\_: 乘运算 \_\_truediv\_\_: 除运算 \_\_mod\_\_: 求余运算

\_\_pow\_\_: 乘方 同理

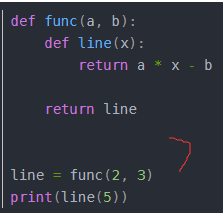
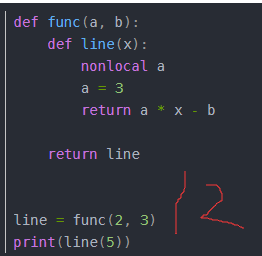
i:\_\_enter\_\_ \_\_exit\_\_:为了让一个对象兼容with语句，必须在这个对象的类中声明\_\_enter\_\_和\_\_exit\_\_方法,举例：



exit()中的三个参数分别代表异常类型，异常值和追溯信息,with语句中代码块出现异常，则with后的代码都无法执行，如果\_\_exit()返回值为True,那么异常会被清空，就好像啥都没发生一样，with后的语句正常执行

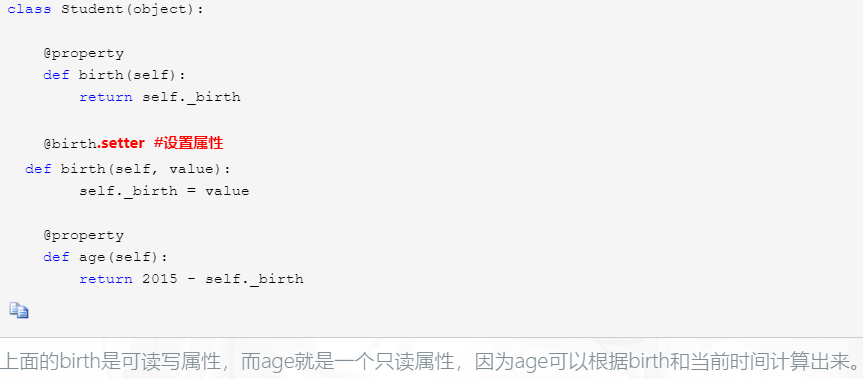
1. 闭包：①定义：在一个外函数中定义了一个内函数，内函数里运用了外函数的临时变量，并且外函数的返回值是内函数的引用。这样就构成了一个闭包。

②在闭包内函数中，我们可以随意使用外函数绑定来的临时变量，如果想要在内函数中修改外函数的值，需要使用 nonlocal 关键字声明变量，举例：

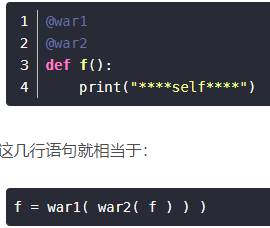
 

1. 装饰器：

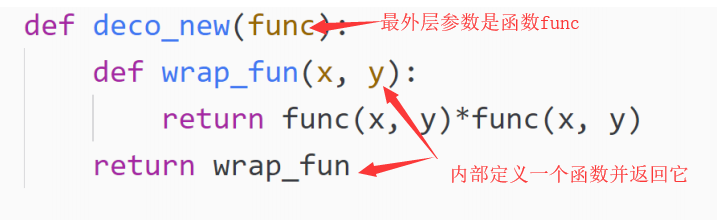
①@property的作用和getter setter的解释：@property是用来修饰方法的，把一个getter方法变成属性，只需要加上@property就可以了，可以使方法像属性一样被访问。可通过只使用@property的方法而不设置@方法名.setter。这样可以隐藏属性名，让用户进行使用的时候无法随意修改，而另一个装饰器@方法名.setter负责把一个setter方法变成属性赋值，举例：



②多装饰器：一般情况下，在函数中可以使用一个装饰器，但是有时也会有两个或两个以上的装饰器。多个装饰器装饰的顺序是从里到外(就近原则)，而调用的顺序是从外到里（就远原则） 参考https://blog.csdn.net/evergreenswj/article/details/90510582 记住这一点：



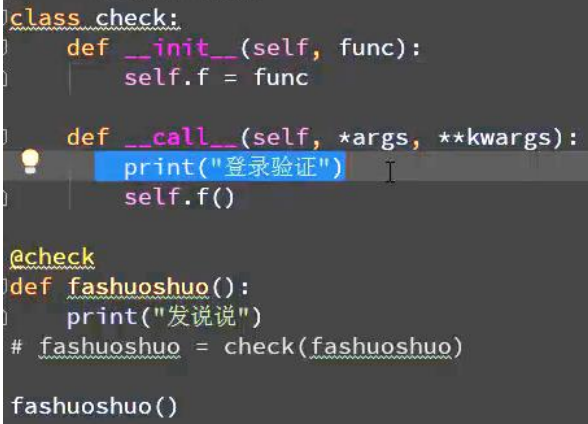
③带参数的装饰器：一般情况下装饰器的模板如下：



而如果我们想让最外层的参数不是函数，即实现带参装饰器，则需要让装饰器再加一层，举例：



④类装饰器：类里包含一个成员变量f,代表调用的函数，把被装饰的函数前加上@装饰类名，然后调用该函数就会自动调用装饰类中的\_\_call\_\_方法



1. 抽象基类：①定义：不能被实例化的类，职责是定义子类应该实现的方法

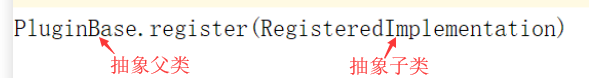
②借助abc模块来实现抽象类，两种方式，见下图：





③继承抽象类的两种方式：

1. 调用抽象基类的register 方法，来把其他类注册到抽象基类下。这种方法实现的，我们称之为称为虚拟子类，举例：



注册方式的特点：

1.虚拟子类可以实现抽象基类中的部分API接口，也可以根本不实现，但是issubclass(), issubinstance()进行判断时仍然返回真值。当没有实现抽象方法时，实例化时候不会报错， 只有在调 用时候才会报错。

2.父类不会出现在类的MRO，故而也不能通过super()来调用父类的方法。

1. 继承抽象类：直接从抽象基类派生子类，举例：



继承抽象类的特点：1. 除非子类实现抽象基类的抽象方法，否则子类不能实例化，

1. 抽象类中也可以包含具体方法，在子类中可以通过super()来调用。

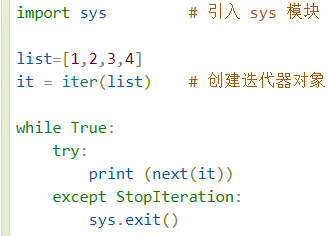
8.迭代器与生成器

①迭代器定义：迭代是Python最强大的功能之一，是访问集合元素的一种方式。

迭代器是一个可以记住遍历的位置的对象，迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。

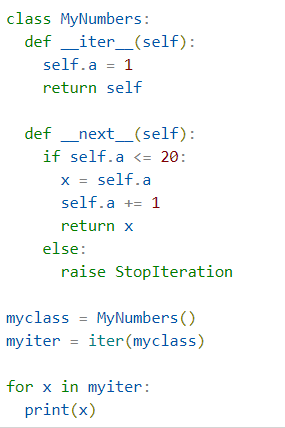
②迭代器如何构造：有两个基本的方法：iter() 和 next()。

字符串，列表或元组对象用于创建迭代器，举例：



如果把一个类作为一个迭代器使用需要在类中实现两个方法 \_\_iter\_\_() 与 \_\_next\_\_() \_\_iter\_\_() 方法返回一个特殊的迭代器对象， 这个迭代器对象实现了 \_\_next\_\_() 方法并通过 StopIteration 异常标识迭代的完成。

\_\_next\_\_() 方法会返回下一个迭代器对象，举例(输出结果为1...20)：

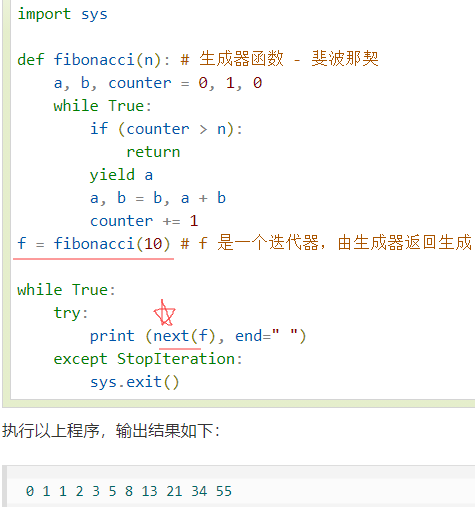


③生成器定义：在Python中，使用了yield的函数被称为生成器

④生成器举例：生成器是一个返回迭代器的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器。

在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回 yield 的值, 并在下一次执行 next() 方法时从当前位置继续运行。

调用一个生成器函数，返回的是一个迭代器对象，举例：



9.lambda表达式(匿名函数)：

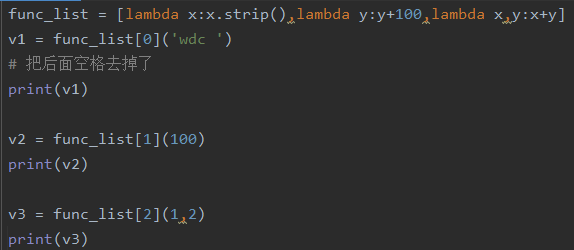
def func(x,y):

return x+y 等价于

lambda x,y: x+y lambda表达式返回一个函数对象

例子：func = lambda x,y:x+y

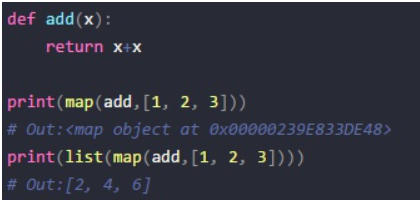
注意，lambda表达式也可以放在列表中：



10. 高阶函数：①定义：一个函数可以作为参数传给另外一个函数，或者一个函数的返回值为另外一个函数（若返回值为该函数本身，则为递归），满足其一则为高阶函数。

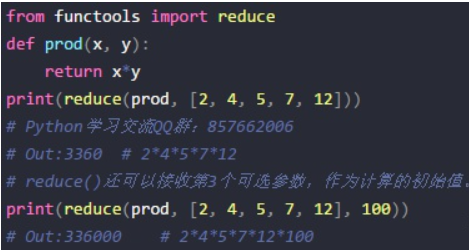
②常见的高阶函数：

a.map()函数:是Python内置的高阶函数，它接收一个函数f和一个list,并通过把函数f依次作用在 list 的每个元素上，得到一个新的list并返回,举例：

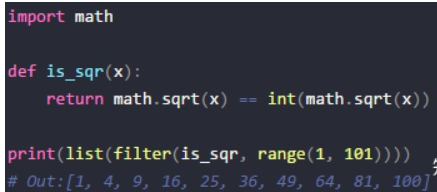


b.reduce()函数:reduce()函数也是Python内置的一个高阶函数。

reduce()函数接收的参数和 map()类似，一个函数 f，一个list，但行为和 map()不同，reduce()传入的函数f必须接收两个参数，reduce()对list的每个元素反复调用函数f，并返回最终结果值。在 Python3 中，reduce() 函数已经被从全局名字空间里移除了，它现在被放置在 functools 模块里，如果想要使用它，则需要通过引入 functools 模块来调用 reduce() 函数,举例：



1. filter()函数:是 Python 内置的另一个有用的高阶函数，filter()函数接收一个函数 f 和一个list，这个函数f的作用是对每个元素进行判断，返回 True或 False，filter()根据判断结果自动过滤掉不符合条件的元素，返回由符合条件元素组成的新list。



11. 列表表达式

12. 正则表达式

13. 内置模块:①时间模块：import time,time.time()

②import random,

#0,1之间时间生成的浮点数

print(random.random())

#随机生成传入参数范围内的数字(包头包尾) 即 1，2，3

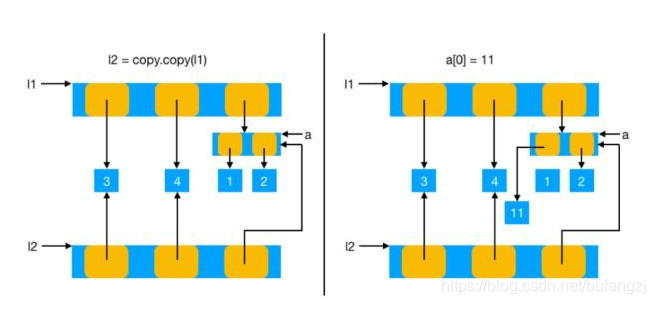
print(random.randint(1, 3))

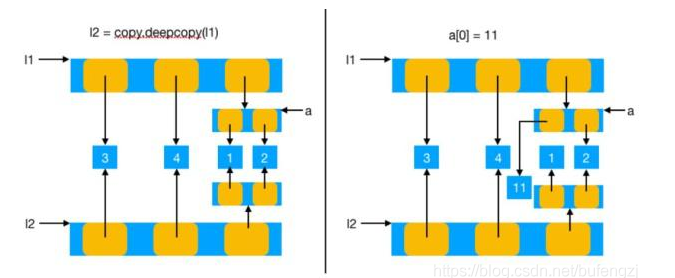
#随机生成传入参数范围内的数字，range顾头不顾尾

print(random.randrange(1, 3))

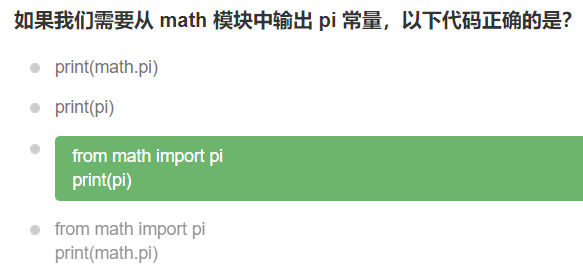
③import sys，sys.exit(0)退出程序

14. 浅深拷贝：





易错题：





中国人".encode("utf-8")