Python教程

# Python语言的学习（上）

**1、基础知识**

数据结构：常量、变量、元组（列表）、字典

数据类型：int float str

算法：分支和循环

面向对象：类、属性、方法

封装、继承、多态

**2、Python语言的特点：**

例子：定义三个不同类型的变量，赋值，输出

无头文件、无主函数、无大括号、无分号，简洁，靠缩进来区分语句，且变量的定义不需要书写类型，直接赋值，Python自动识别数据类型，字符串赋值时可用双引号或单引号，两者区别：双引号可用于打印对象中含有单引号时

注释：单行注释---#，多行注释---’’’(一对缩进相同)

注意：不要用一些关键字命名：selenium

**3、输入输出函数：**

输入函数：input

例子：a=input(“提示信息：”)、

多个数据同时输入：a,b = input('请输入：').split()—split内可以添加分隔符

输入：happy hello

注：input函数输入的所有数据都为str类型

数据类型查看函数：type(变量名)

例子：print(type(a))

输出函数：print('结果为：',a,b)

例子：print(("%d,%f,%s")%(a,b,c)) –字符串采用%s

也支持format用法：print('xxx={},yyy={}'.format(a,b))

**4、元组/列表/序列**

**定义：**

s=(1,2.5,'much')----tuple

s=[1,2.5,'much']----list

#tuple类型的元组内的值不可改变，而list类型的元组可以改变

与数组不同的地方：1.括号；2.元素类型

**引用：**

#引用方式：

#1整体应用---元组名

例子：print(s)

#2下标引用---下标从0开始

例子：print(s[2])

#3下限、上限和步长

例子：print(s[1:7:2])

---下限表示从谁s[1]打印，上限表示打印到谁s[7]但是该数默认不打印，步长是下一元素的增量，步长为1可省略，

#4 特殊引用

例子：print(s[-2])、print(s[1::2])

练习：定义一个1-9的元组，1、输出倒数第3个元素；2、输出值468

a = [x for x in range(1,9)] a[-3] a[3] a[5] a[7]

tuple = tuple(a)

思考：如何定义1-999的元组？

tuple([x for x in range(1,999)])

range(n):产生一个0到n-1的列表

range(m,n)：产生一个m到n-1的列表

range(m,n,i)：产生一个m到n-1，步长为i的列表

输出：list(range(1,1000))

**操作方法：**

* + - 1. 逆序存放：s.reverse()---改变原来元组的值
      2. 排序存放：s.sort()------改变原来元组的值
      3. 排序：sorted(s)---------不改变原来元组的值，只返回一个排序结果
      4. 插入：s.insert(n,m)-----在某一位置(s[n]前面)插入该值m，n为下标
      5. 追加：s.append(n)-------在该元组末尾追加元素n
      6. 连接两个列表：m.extend(n) ---将n列表追加到m列表之后
      7. 删除指定元素：m.pop(n) ------删除m[n]并返回该值，n为下标
      8. 删除指定元素第一次出现的值：m.remove(n) –将第一次出现的元素n删除
      9. 返回该值在列表中出现的次数：m.count(n) –返回元素n在列表中出现的次数
      10. 返回列表中元素最大值：max(s)
      11. 返回列表中元素最小值：min(s)
      12. 长度：len(s) ---返回列表元素的个数
      13. 删除：del s[n]----删除s[n]，n为下标
      14. 获得元素的下标：s.index(n) -------得到元素n在列表中的下标
      15. 清空列表：s.clear() --------清空列表
      16. in和 notin---用来检查是否在列表中

练习：列表[11,13,5,7,0,56,23,34,72]，

求该列表中的最大值，最小值及列表中一共有几个元素

a = [11,13,5,7,0,56,23,34,72]

max(a)

min(a)

len(a)

获取56元素在列表的位置

a.index(56)

追加元素7

a.append(7)

删除元素0

a.pop(a.index(0))

排序列表（由大到小）

a.reverse()

排序由小到大

a.sorted()

将列表[66,67,68]与原列表组合

b = [66,67,58]

a.extend(b)

**5、列表推导式**

 python中列表推导式用于使用其他列表创建一个新列表。

其基本形式为： [表达式 for 变量 in 列表]

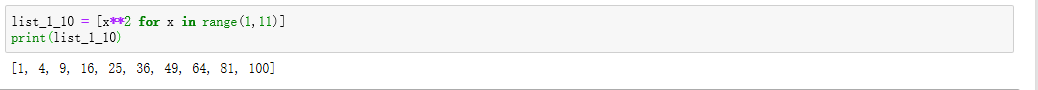
如：

# 想得到1-10的平方组成的list

list\_1\_10 = [x\*\*2 for x inrange(1,11)]

print(list\_1\_10)

输出为：



复杂一点的列表表达式可以对原列表进行变换和筛选。

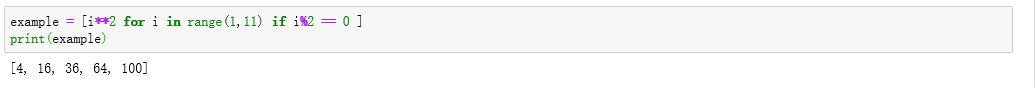
如：

# 想得到1-10中为偶数的平方组成的list

example = [i\*\*2 for i inrange(1,11) if i%2 == 0 ]

print(example)

输出为：



以及对多重嵌套的list进行变换筛选。

如：

# 想得到多重嵌套中的数是2的倍数的平方组成的list

example2 = [[1,2,3],[4,5,6],[7,8,9],[10]]

example3 = [j\*\*2 for i in example2 for j in i if j%2 == 0]

print(example3)

上面的执行顺序可以理解为：

for i in example2：

for j in i：

if j%2 == 0：

j\*\*2

输出为：



如：

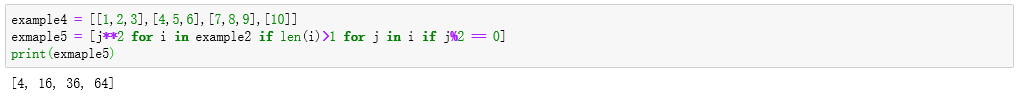
# 想得到多重嵌套的list中一重嵌套中list长度大于1的list中的数为2的倍数的平方组成的list

example4 = [[1,2,3],[4,5,6],[7,8,9],[10]]

exmaple5 = [j\*\*2 for i in example2 iflen(i)>1 for j in i if j%2 == 0]

print(exmaple5)

输出为：



**6、集合**

定义：s = {3,1,2}，集合是无序的，且不重复

引用：不能通过下标引用

操作：

添加-s.add(n)---向集合s添加元素n

删除-s.remove(n) ----删除集合中的元素n，如果元素不存在，报Keyerror异常

删除-s.discard(n) ----删除集合中的元素n，如果元素不存在，不报异常

清空-s.clear() ------清空集合

in和 notin---用来检查是否在集合中

**7、字典**

**定义：**#大括号，键值对形式存在{‘key’:value}

s={'a':10,'b':20,'c':15}

#字典内的元素没有顺序，不能够通过下标引用

**引用：**只能通过键来引用或整体引用

print(s['c'])；print(s)

**赋值：**当key不存在时即可为字典插入，但是如果存在，即为修改

s['d']=11

**操作方法：**

1、取出所有的键：s.keys()

2、取出所有的值：s.values()

3、取出所有的键值对，作为一个元组内的元素：s.items()

4、清空字典：s.clear()

5、删除：del s[‘key’]

**8、分支-判断**

|  |  |  |  |
| --- | --- | --- | --- |
| 分类 | If-分支 | If-else分支 | If-elif嵌套 |
| 语法 | If 条件：  语句体 | If 条件**：**  语句体1  else**：**  语句体2 | If 条件1：  语句体1  elif条件2：  语句体2  elif条件3：  语句体3  else：  语句体 |
| 例子 | a=0  if a>0:  a+=1  print a | a=input()  if a>0:  a+=1  else:  a-=1  print a | a=input()  if a>0:  a+=1  elif a==0:  a-=1  elif a<0:  a+=2  print a |

例子：

输入一个数，判断该数的范围：90-100：等价为A……60以下：等级为E

A = []

x = input()

def function(x)

定义一个列表，从键盘输入一个随机数，判断该数是否在列表中

注：Python不支持&&、||，需要and、or代替，同时也支持多个条件连接：0<a<10

预习：

面向对象的类、属性、方法、实例化、

作业：输出100以内的所有质数（只能被1和其本身整除的数）

1. **Python语言学习下**

**1、循环**

循环四要素：

1. 定义循环变量；2、循环条件；3、循环体；4、循环变量发生变化

n = 0; n<5; print n; n+=1

把程序做成循环的步骤：

A、确定循环体，全部选中用Tab键内推，用循环结构包裹

B、从上到下依次编写：1/2/4

**a、while循环**

**语法：**

循环变量

while条件**：**

循环体

循环变量发生变化

**例子：**

i=0

while i<5:

print i

i=i+1

练习：计算1+2+3+4……+100的和

def sum(m):

sm = 0

for i in range(0,m+1):

sm +=i

return sm

**b、for循环**

**语法：**

for 变量 in 序列:

循环体

**例子：**

for i in [1,2,1,4]:

print i,'ok'

range（n）：产生一个0到n-1的序列

range(m,n)：产生一个m到n-1的序列

range(m,n,i)产生一个m到n-1且步长为i的序列

**c、continue和break**

continue：停止当前这次循环，继续下次循环

break：跳出所有循环

**例子：**

for i in range(1,5):

if i==3:

break/continue

print i

else:

print i

break：跳出最近这层整体循环

continue：停止当前这次循环，继续判断下次循环是否成立

练习：

求10阶乘

def fn(n):

if n == 0 or n == 1:

return 1

else:

return n \* fn(n-1)

求100以内能被3整除的数，并将作为列表输出

nums = []

for i in range(0,101):

if i % 3 == 0:

nums.append(i)

列表[1,2,3,4,3,4,2,5,5,8,9,7],将此列表去重，得到一个唯一元素的列表

ids = [1,2,3,4,3,4,2,5,5,8,9,7]

new\_ids =[]

for id in ids:

if id not in new\_ids:

new\_ids.append(id)

求斐波那契数列1 1 2 3 5 8 13 ……

求10000以内的质数

nums = []

def f(n):

for i in range(2,n):

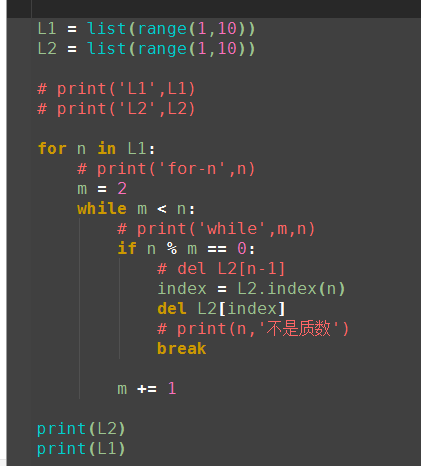
for j in range(2,i):

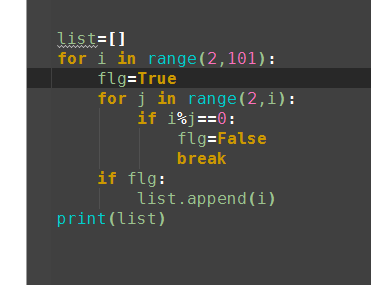
if i % j == 0:

break

else:

nums.append(i)





**2、函数**

**a、定义：**

def函数名(参数1，参数2）：----（参数可有可无）

语句体

return 结果----（return可有可无）

**b、调用:**

函数名

变量=函数名(参数1，参数2)----（变量，参数可有可无）

1、调用；2、传参；3、接收返回值

**c、例子：**

def add(a,b):

print a+b

a,b=input('请输入两个数，并用逗号隔开：')

add(a,b)

**练习：**

设计一个计算器，输入两个数，自动实现加减乘除（进阶：根据用户输入的计算符号计算结果）

a, b = input("请输入个数，以逗号分隔：")

def add (a,b):

return a + b

def sub(a, b):

return a - b

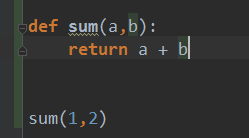
def mul(a, b):

return a -b

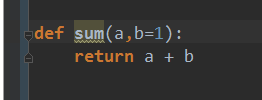
def div(a,b):

return a % b

一、位置参数：



二、默认参数：

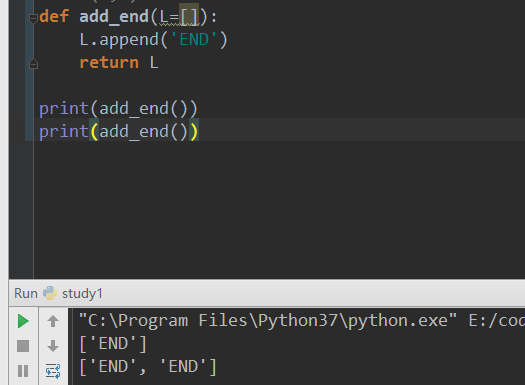


sum(1)和sum(1,1)效果一样

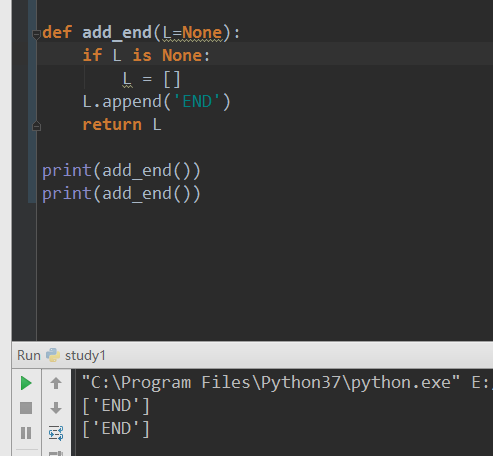
从上面的例子可以看出，默认参数可以简化函数的调用。设置默认参数时，有一点要注意：

必选参数在前，默认参数在后，否则Python的解释器会报错

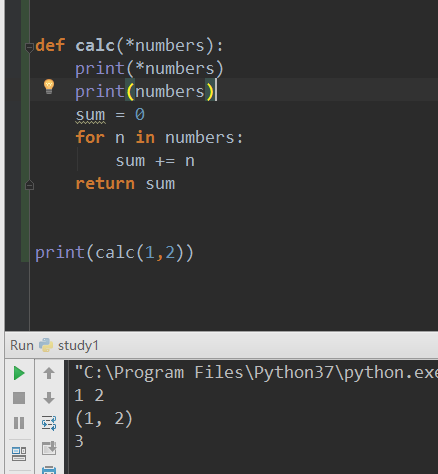
定义默认参数要牢记一点：默认参数必须指向不变对象！



解决方案：让参数指向不可变对象

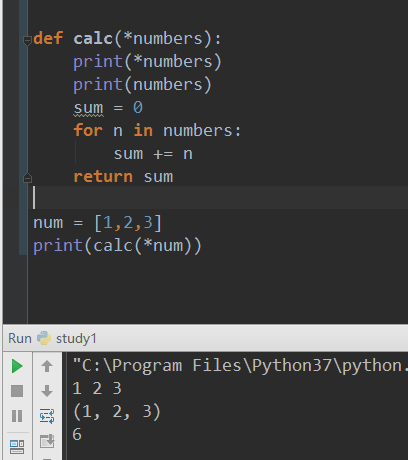


三、可变参数

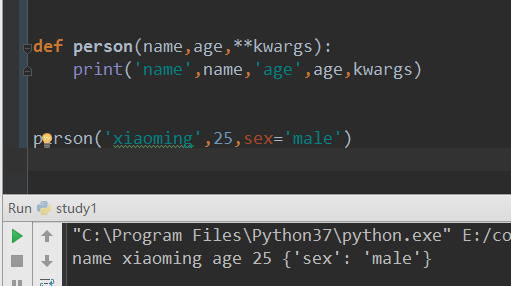


在参数前面加了一个\*号。在函数内部，参数numbers接收到的是一个tuple，可变参数允许你传入0个或任意个参数

也可以使用已有列表作为入参传入



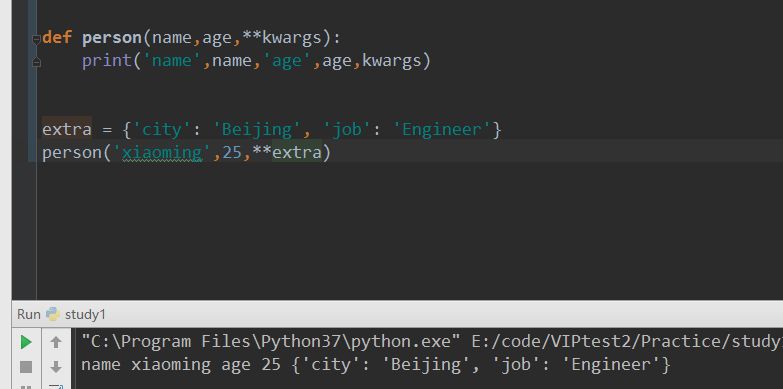
四、关键字参数



可变参数允许你传入0个或任意个参数，这些可变参数在函数调用时自动组装为一个tuple。而关键字参数允许你传入0个或任意个含参数名的参数，这些关键字参数在函数内部自动组装为一个dict。

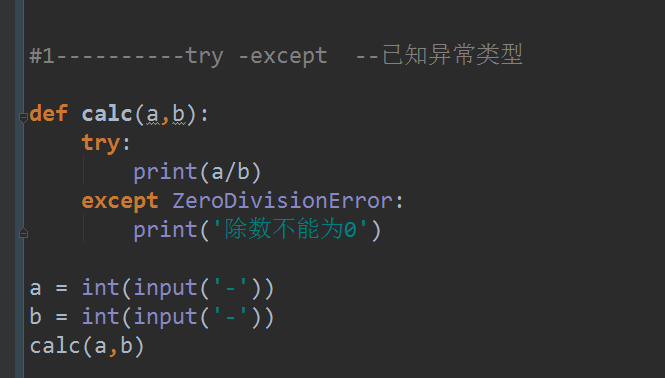
关键字参数有什么用？它可以扩展函数的功能。比如，在person函数里，我们保证能接收到name和age这两个参数，但是，如果调用者愿意提供更多的参数，我们也能收到。试想你正在做一个用户注册的功能，除了用户名和年龄是必填项外，其他都是可选项，利用关键字参数来定义这个函数就能满足注册的需求。

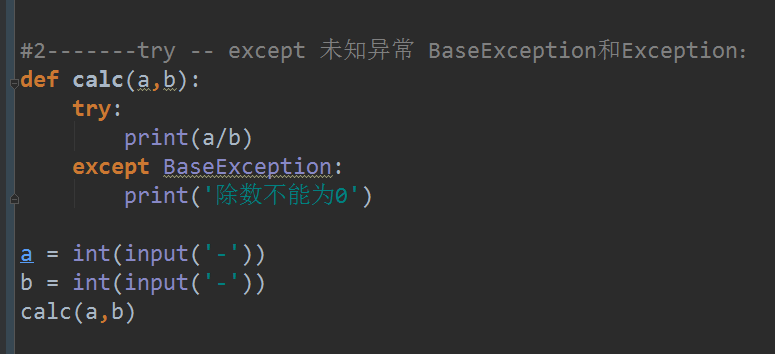
和可变参数类似，也可以先组装出一个dict，然后，把该dict转换为关键字参数传进去：

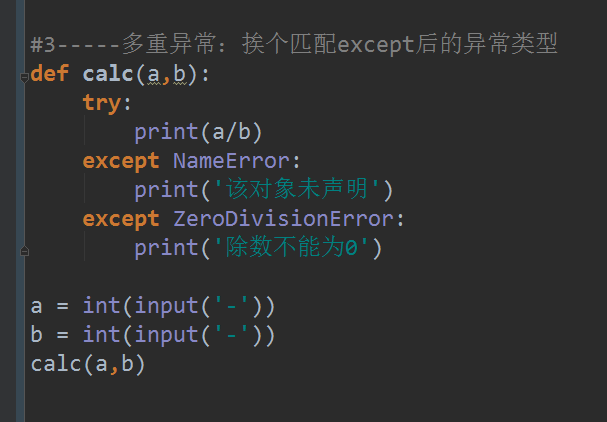


练习：

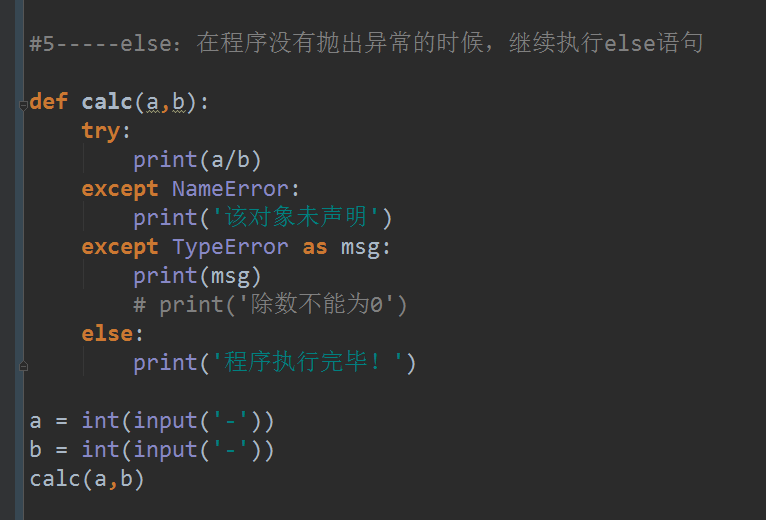
**3、异常处理**

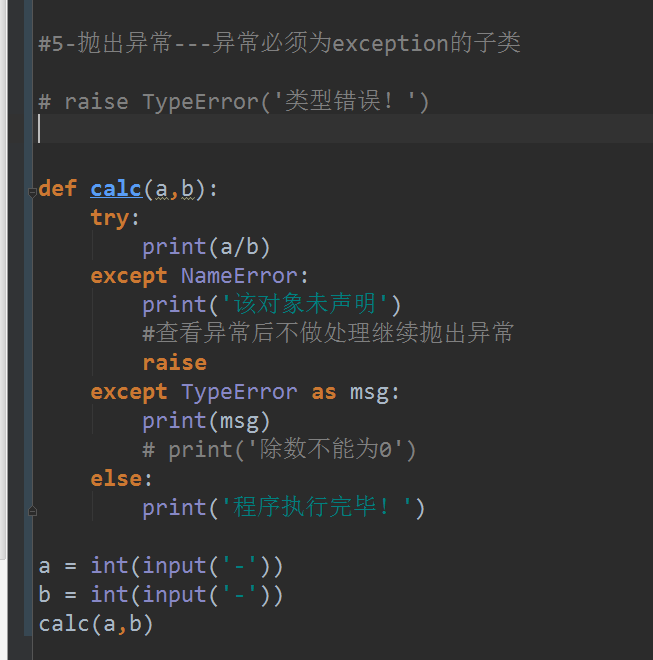












注意：当exception中含有return时，finally会覆盖最后一次return的值

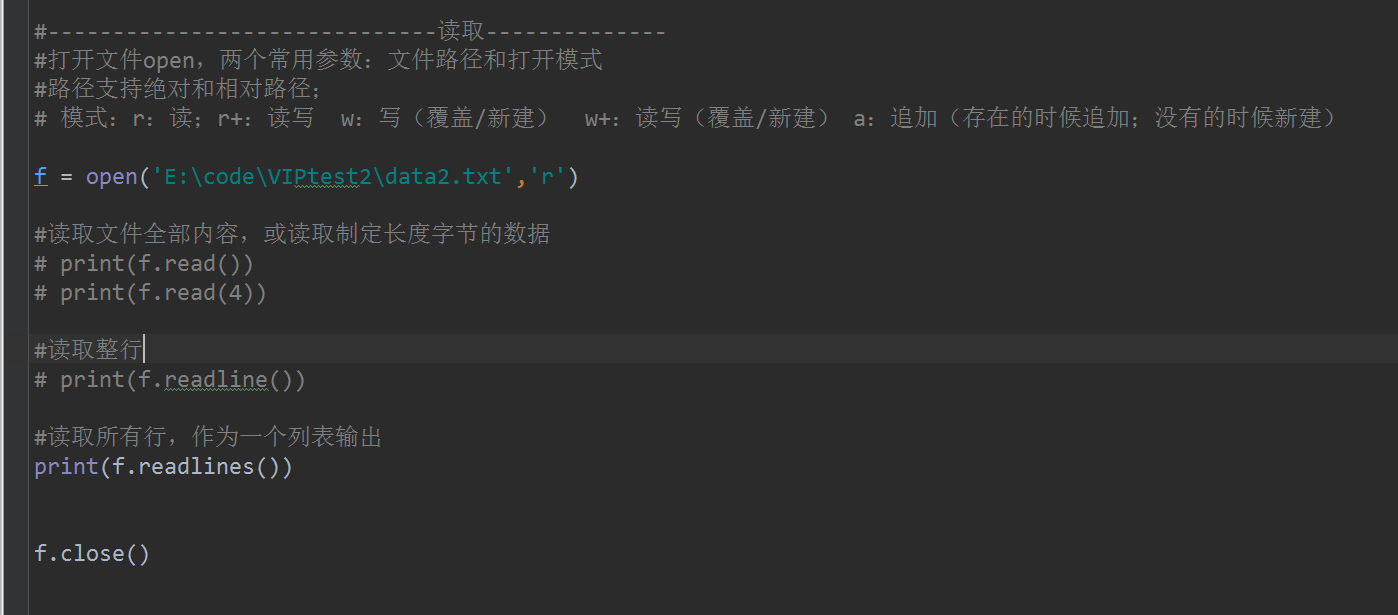
异常类型：

python标准异常  
异常名称 描述  
BaseException 所有异常的基类  
SystemExit 解释器请求退出  
KeyboardInterrupt 用户中断执行(通常是输入^C)  
Exception 常规错误的基类  
StopIteration 迭代器没有更多的值  
GeneratorExit 生成器(generator)发生异常来通知退出  
StandardError 所有的内建标准异常的基类  
ArithmeticError 所有数值计算错误的基类  
FloatingPointError 浮点计算错误  
OverflowError 数值运算超出最大限制  
ZeroDivisionError 除(或取模)零 (所有数据类型)  
AssertionError 断言语句失败  
AttributeError 对象没有这个属性  
EOFError 没有内建输入,到达EOF 标记  
EnvironmentError 操作系统错误的基类  
IOError 输入/输出操作失败  
OSError 操作系统错误  
WindowsError 系统调用失败  
ImportError 导入模块/对象失败  
LookupError 无效数据查询的基类  
IndexError 序列中没有此索引(index)  
KeyError 映射中没有这个键  
MemoryError 内存溢出错误(对于Python 解释器不是致命的)  
NameError 未声明/初始化对象 (没有属性)  
UnboundLocalError 访问未初始化的本地变量  
ReferenceError 弱引用(Weak reference)试图访问已经垃圾回收了的对象  
RuntimeError 一般的运行时错误  
NotImplementedError 尚未实现的方法  
SyntaxError Python 语法错误  
IndentationError 缩进错误  
TabError Tab 和空格混用  
SystemError 一般的解释器系统错误  
TypeError 对类型无效的操作  
ValueError 传入无效的参数  
UnicodeError Unicode 相关的错误  
UnicodeDecodeError Unicode 解码时的错误  
UnicodeEncodeError Unicode 编码时错误  
UnicodeTranslateError Unicode 转换时错误  
Warning 警告的基类  
DeprecationWarning 关于被弃用的特征的警告  
FutureWarning 关于构造将来语义会有改变的警告  
OverflowWarning 旧的关于自动提升为长整型(long)的警告  
PendingDeprecationWarning 关于特性将会被废弃的警告  
RuntimeWarning 可疑的运行时行为(runtime behavior)的警告  
SyntaxWarning 可疑的语法的警告  
UserWarning 用户代码生成的警告

**4、文件IO**

读取

格式：



写入：





作业：读取data文件中的数据，将所有的数字按照从小到大的顺序写入backup文件

**5、模块和包**

模块：模块是一个包含所有你定义的函数和变量的文件，其后缀名是.py。模块可以被别的程序引入，以使用该模块中的函数等功能。这也是使用 python 标准库的方法。

包：1. 当一个目录内含有\_\_init\_\_ . py文件时，就可以视该目录为一个包

调用包和模块

import 模块名/包名

from包名import模块

绝对导入

fromparcticeimportmodule

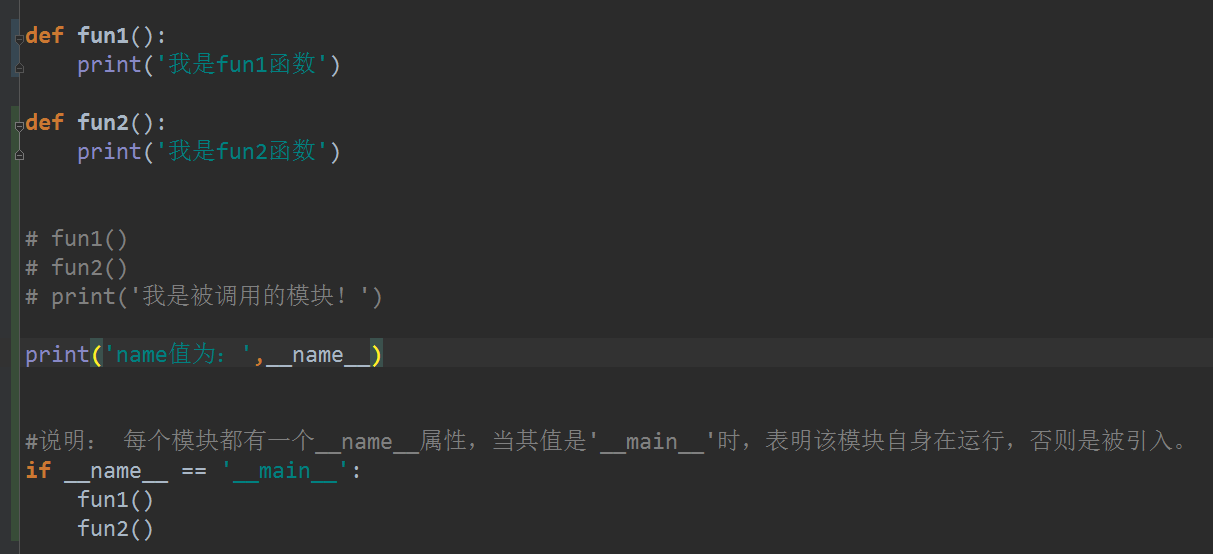
相对导入

from .database import Database # 点号表示使用当前路径的database模块

from ..database import Database # 使用两个点号表示访问上层的父类



\_\_name\_\_的作用：



一个模块被另一个程序第一次引入时，其主程序将运行。如果我们想在模块被引入时，模块中的某一程序块不执行，我们可以用\_\_name\_\_属性来使该程序块仅在该模块自身运行时执行。

**6、面向对象的类、属性、方法**

1. **基础：**

类：具有相同的特性，且能够完成某些动作的事物组成的一个集合

属性：类中的事物所具有的特性，趋于静态

方法：类中的事物所能够完成的动作或功能，趋于动态

实例/对象：类中某一个具体的对象或实例

实例化：定义类中某一个对象的过程

1. **类的定义：简单类的创建1🡪2🡪3**

**封装：**

用法：

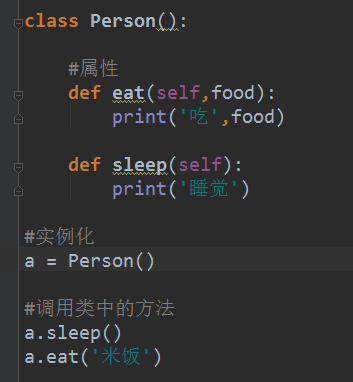
class类名（object）：

属性

方法

注意：属性和方法可以为空，里面直接写pass即可，表示是一个空类

例子：



注：dir(类名)，可查看类中的方法

1. **类的实例化：**

变量=类名('xx') ---是否需要传参就看\_\_init\_\_方法中除了self是否还有其他参数

例子：

select avg(r.chinese) from report r left join student s on r.id = s.id group by s.class

1. **类中方法的调用：**

实例名.方法名('xx') ---是否需要传参就看该方法中除了self是否还有其他参数

**练习：**

a、定义一个学生类：Student、内部含有一个方法：study，实现打印：xx学习xx课程

b、定义一个类名：Student—学生、类内部含有一个属性：sno—学号，一个方法：study—学习，实现打印：学号为xx的学生，学习xx课程

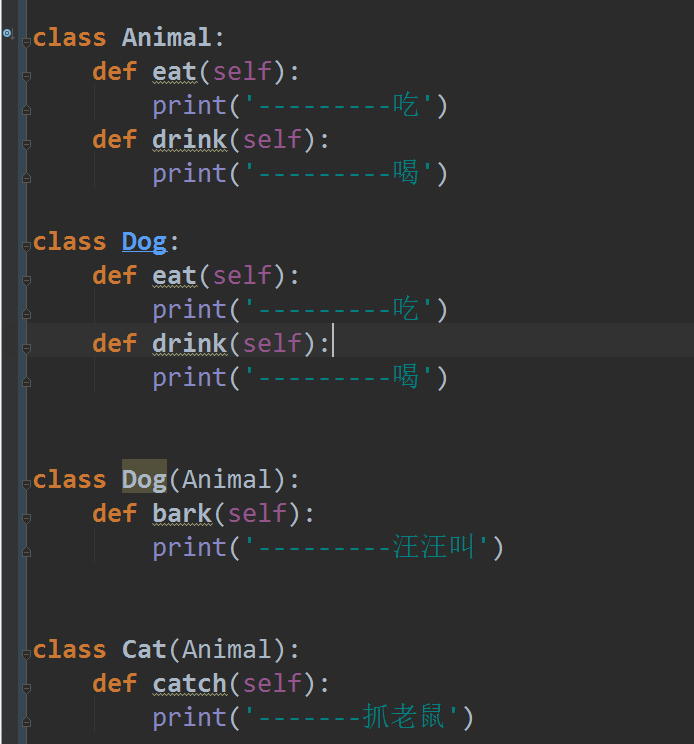
1. **继承**

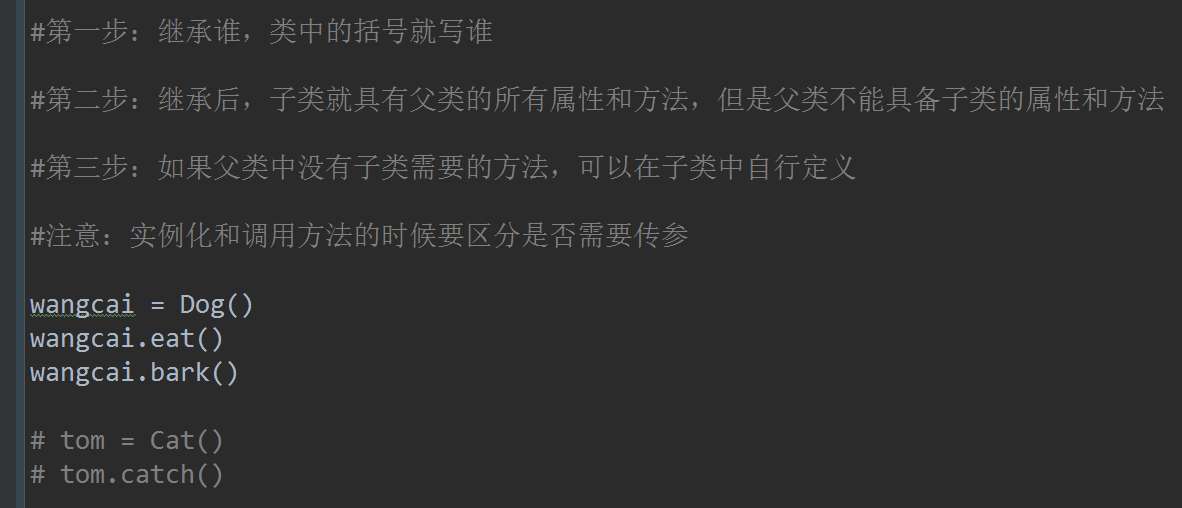
**为了提高代码的复用性，可以通过继承来减少属性和方法的编写**

语法：

继承谁就在括号中写谁-类名

例子：





练习：

定义一个Teacher类，继承Person类，拥有自身的属性工号：gh，自身的方法：teach教课（课程）；

1、实现gh为xx的老师，教xx课

2、实现gh为xx老师，在xx上班，一月工资xx

3、名字是xx，工号为xx的老师，吃饭

1. **重写：**

当继承的父类方法不满足自身需要的时候，我们可以重写该父类的方法，注意，必须同名

例子：



调用父类的方法有两种：

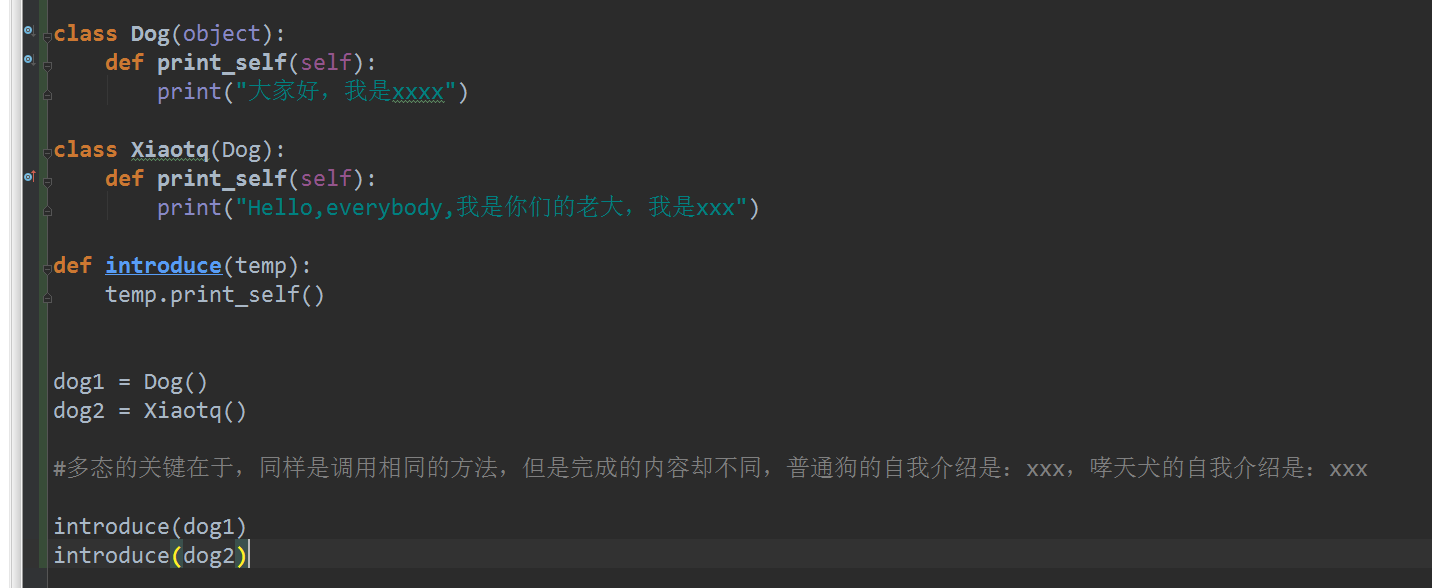


多重继承的注意事项：



1. **多态：**

**多态性是指具有不同功能的函数可以使用相同的函数名，这样就可以用一个函数名调用不同内容的函数。在面向对象方法中一般是这样表述多态性：向不同的对象发送同一条消息，不同的对象在接收时会产生不同的行为（即方法）。也就是说，每个对象可以用自己的方式去响应共同的消息。所谓消息，就是调用函数，不同的行为就是指不同的实现，即执行不同的函数。**



1. **私有属性和私有方法**

类的私有属性

**\_\_abc**：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.\_\_abc**。

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

**\_\_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.\_\_ methods**。



Python练习题：

1、打印小猫爱吃鱼，小猫要喝水

class cat:

def \_\_init\_\_(self):

self.favorite = "eat finsh"

def drink(self):

print("drink water")

2、小明爱跑步，爱吃东西。

1）小明体重75.0公斤  
2）每次跑步会减肥0.5公斤  
3）每次吃东西体重会增加1公斤  
4）小美的体重是45.0公斤

class xm:

def \_\_init\_\_(self):

self.weight = 75

def run(self):

self.weight -=1

def eat(self):

self.weight += 0.5

3、摆放家具

需求：

1）.房子有户型，总面积和家具名称列表

   新房子没有任何的家具

2）.家具有名字和占地面积，其中

   床：占4平米

   衣柜：占2平面

   餐桌：占1.5平米

3）.将以上三件家具添加到房子中

4）.打印房子时，要求输出:户型，总面积，剩余面积，家具名称列表

class House:  
 def \_\_init\_\_(self):  
 self.chuang = 4  
 self.yigui = 2  
 self.canzhou = 1.5  
 self.chuang\_name = "床"  
 self.yigui\_name = "衣柜"  
 self.canzhuo\_name = "餐桌"  
  
 def output(self, area, huxing):  
 jiaju = []  
 area = area - self.chuang - self.yigui - self.canzhou  
 jiaju.append(self.chuang\_name)  
 jiaju.append(self.yigui\_name)  
 jiaju.append(self.canzhuo\_name)  
 print("房子：户型:{0},总面积:{1},家具列表：{2}".format(huxing, area,jiaju))  
  
  
new\_house = House()  
new\_house.output(area=50, huxing="俩室一厅")

4.士兵开枪

需求：  
1）.士兵瑞恩有一把AK47  
2）.士兵可以开火(士兵开火扣动的是扳机)  
3）.枪 能够 发射子弹(把子弹发射出去)  
4）.枪 能够 装填子弹 --增加子弹的数量

class Gun:  
 def \_\_init\_\_(self):  
 self.danyao\_num = 30  
 self.name = "ak47"  
 def shut(self):  
 print('开火')  
 self.danyao\_num -= 1  
 def lodding(self):  
 self.danyao\_num += 30  
  
class Solid():  
 def shuout(self, gun):  
 gun.shut()  
  
ak47 = Gun()  
rean = Solid()  
  
rean.shuout(gun=ak47)  
print(ak47.danyao\_num)

**7、字符串操作**

这些方法实现了string模块的大部分方法，如下表所示列出了目前字符串内建支持的方法，所有的方法都包含了对Unicode的支持，有一些甚至是专门用于Unicode的。

常用的字符串操作方法如下：

|  |  |
| --- | --- |
| [string.capitalize()](https://www.runoob.com/python/att-string-capitalize.html) | 把字符串的第一个字符大写 |
| [string.count(str, beg=0, end=len(string))](https://www.runoob.com/python/att-string-count.html) | 返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则  返回指定范围内 str 出现的次数 |
| [string.decode(encoding='UTF-8', errors='strict')](https://www.runoob.com/python/att-string-decode.html) | 以 encoding 指定的编码格式解码 string，如果出错默认报一个 ValueError  的 异 常 ， 除非 errors 指 定 的 是 'ignore' 或 者'replace' |
| [string.encode(encoding='UTF-8', errors='strict')](https://www.runoob.com/python/att-string-encode.html) | 以 encoding 指定的编码格式编码 string，如果出错默认报一个ValueError  的异常，除非 errors 指定的是'ignore'或者'replace' |
| [string.endswith(obj, beg=0, end=len(string))](https://www.runoob.com/python/att-string-endswith.html) | 检查字符串是否以 obj 结束，如果beg 或者 end 指定则检查指定的范围内  是否以 obj 结束，如果是，返回 True,否则返回 False. |
| [string.find(str, beg=0, end=len(string))](https://www.runoob.com/python/att-string-find.html) | 检测 str 是否包含在 string 中，如果 beg 和 end 指定范围，则检查是否  包含在指定范围内，如果是返回开始的索引值，否则返回-1 |
| [string.index(str, beg=0, end=len(string))](https://www.runoob.com/python/att-string-index.html) | 跟find()方法一样，只不过如果str不在 string中会报一个异常. |
| [string.isalpha()](https://www.runoob.com/python/att-string-isalpha.html) | 如果 string 至少有一个字符并且所有字符都是字母则返回 True,  否则返回 False |
| [string.isdigit()](https://www.runoob.com/python/att-string-isdigit.html) | 如果 string 只包含数字则返回 True 否则返回 False. |
| [string.islower()](https://www.runoob.com/python/att-string-islower.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些  (区分大小写的)字符都是小写，则返回 True，否则返回 False |
| [string.isnumeric()](https://www.runoob.com/python/att-string-isnumeric.html) | 如果 string 中只包含数字字符，则返回 True，否则返回 False |
| [string.isupper()](https://www.runoob.com/python/att-string-isupper.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分  大小写的)字符都是大写，则返回 True，否则返回 False |
| [string.join(seq)](https://www.runoob.com/python/att-string-join.html) | 以 string 作为分隔符，将 seq 中所有的元素(的字符串表示)合并为  一个新的  字符串 |
| [string.lower()](https://www.runoob.com/python/att-string-lower.html) | 转换 string 中所有大写字符为小写. |
| [string.lstrip()](https://www.runoob.com/python/att-string-lstrip.html) | 截掉 string 左边的空格 |
| [string.replace(str1, str2,  num=string.count(str1))](https://www.runoob.com/python/att-string-replace.html) | 把 string 中的 str1 替换成 str2,如果 num 指定，则替换不超过 num 次. |
| [string.rstrip()](https://www.runoob.com/python/att-string-rstrip.html) | 删除 string 字符串末尾的空格. |
| [string.lstrip()](https://www.runoob.com/python/att-string-lstrip.html) | 截掉 string 左边的空格 |
| [string.split(str="", num=string.count(str))](https://www.runoob.com/python/att-string-split.html) | 以 str 为分隔符切片 string，如果 num 有指定值，则仅分隔 num+  个子字符串 |
| [string.startswith(obj, beg=0,end=len(string))](https://www.runoob.com/python/att-string-startswith.html) | 检查字符串是否是以 obj 开头，是则返回 True，否则返回 False。  如果beg 和 end 指定值，则在指定范围内检查. |
| [string.strip([obj])](https://www.runoob.com/python/att-string-strip.html) | 在 string 上执行 lstrip()和 rstrip() |
| [string.swapcase()](https://www.runoob.com/python/att-string-swapcase.html) | 翻转 string 中的大小写 |
| [string.upper()](https://www.runoob.com/python/att-string-upper.html) | 转换 string 中的小写字母为大写 |

练习1：

有这样一个文件，文件内容如下：

Lucy|18601914231|男|19890218

Jack|18101913132|女|19810311

Tom|18201912533|女|19830713

Lily|18301911734|男|19870210

任务1-找出所有L开头的人名

任务2-按照年龄进行排序

任务3-找出所有女性用户的信息

**8、os操作**

**os** 模块提供了非常丰富的方法用来处理文件和目录。常用的方法如下表所示：

|  |  |
| --- | --- |
| **序号** | **方法及描述** |
| 1 | [os.access(path, mode)](https://www.runoob.com/python/os-access.html) 检验权限模式 |
| 2 | [os.chdir(path)](https://www.runoob.com/python/os-chdir.html) 改变当前工作目录 |
| 3 | [os.chflags(path, flags)](https://www.runoob.com/python/os-chflags.html) 设置路径的标记为数字标记。 |
| 4 | [os.chmod(path, mode)](https://www.runoob.com/python/os-chmod.html) 更改权限 |
| 5 | [os.chown(path, uid, gid)](https://www.runoob.com/python/os-chown.html) 更改文件所有者 |
| 6 | [os.chroot(path)](https://www.runoob.com/python/os-chroot.html) 改变当前进程的根目录 |
| 7 | [os.close(fd)](https://www.runoob.com/python/os-close.html) 关闭文件描述符 fd |
| 8 | [os.closerange(fd\_low, fd\_high)](https://www.runoob.com/python/os-closerange.html) 关闭所有文件描述符，从 fd\_low (包含) 到 fd\_high (不包含), 错误会忽略 |
| 9 | [os.dup(fd)](https://www.runoob.com/python/os-dup.html) 复制文件描述符 fd |
| 10 | [os.dup2(fd, fd2)](https://www.runoob.com/python/os-dup2.html) 将一个文件描述符 fd 复制到另一个 fd2 |
| 11 | [os.fchdir(fd)](https://www.runoob.com/python/os-fchdir.html) 通过文件描述符改变当前工作目录 |
| 12 | [os.fchmod(fd, mode)](https://www.runoob.com/python/os-fchmod.html) 改变一个文件的访问权限，该文件由参数fd指定，参数mode是Unix下的文件访问权限。 |
| 13 | [os.fchown(fd, uid, gid)](https://www.runoob.com/python/os-fchown.html) 修改一个文件的所有权，这个函数修改一个文件的用户ID和用户组ID，该文件由文件描述符fd指定。 |
| 14 | [os.fdatasync(fd)](https://www.runoob.com/python/os-fdatasync.html) 强制将文件写入磁盘，该文件由文件描述符fd指定，但是不强制更新文件的状态信息。 |
| 15 | [os.fdopen(fd[, mode[, bufsize]])](https://www.runoob.com/python/os-fdopen.html) 通过文件描述符 fd 创建一个文件对象，并返回这个文件对象 |
| 16 | [os.fpathconf(fd, name)](https://www.runoob.com/python/os-fpathconf.html) 返回一个打开的文件的系统配置信息。name为检索的系统配置的值，它也许是一个定义系统值的字符串，  这些名字在很多标准中指定（POSIX.1, Unix 95, Unix 98, 和其它）。 |
| 17 | [os.fstat(fd)](https://www.runoob.com/python/os-fstat.html) 返回文件描述符fd的状态，像stat()。 |
| 18 | [os.fstatvfs(fd)](https://www.runoob.com/python/os-fstatvfs.html) 返回包含文件描述符fd的文件的文件系统的信息，像 statvfs() |
| 19 | [os.fsync(fd)](https://www.runoob.com/python/os-fsync.html) 强制将文件描述符为fd的文件写入硬盘。 |
| 20 | [os.ftruncate(fd, length)](https://www.runoob.com/python/os-ftruncate.html) 裁剪文件描述符fd对应的文件, 所以它最大不能超过文件大小。 |
| 21 | [os.getcwd()](https://www.runoob.com/python/os-getcwd.html) 返回当前工作目录 |
| 22 | [os.getcwdu()](https://www.runoob.com/python/os-getcwdu.html) 返回一个当前工作目录的Unicode对象 |
| 23 | [os.isatty(fd)](https://www.runoob.com/python/os-isatty.html) 如果文件描述符fd是打开的，同时与tty(-like)设备相连，则返回true, 否则False。 |
| 24 | [os.lchflags(path, flags)](https://www.runoob.com/python/os-lchflags.html) 设置路径的标记为数字标记，类似 chflags()，但是没有软链接 |
| 25 | [os.lchmod(path, mode)](https://www.runoob.com/python/os-lchmod.html) 修改连接文件权限 |
| 26 | [os.lchown(path, uid, gid)](https://www.runoob.com/python/os-lchown.html) 更改文件所有者，类似 chown，但是不追踪链接。 |
| 27 | [os.link(src, dst)](https://www.runoob.com/python/os-link.html) 创建硬链接，名为参数 dst，指向参数 src |
| 28 | [os.listdir(path)](https://www.runoob.com/python/os-listdir.html) 返回path指定的文件夹包含的文件或文件夹的名字的列表。 |
| 29 | [os.lseek(fd, pos, how)](https://www.runoob.com/python/os-lseek.html) 设置文件描述符 fd当前位置为pos, how方式修改: SEEK\_SET 或者 0 设置从文件开始的计算的pos;  SEEK\_CUR或者 1 则从当前位置计算; os.SEEK\_END或者2则从文件尾部开始. 在unix，Windows中有效 |
| 30 | [os.lstat(path)](https://www.runoob.com/python/os-lstat.html) 像stat(),但是没有软链接 |
| 31 | [os.major(device)](https://www.runoob.com/python/os-major.html) 从原始的设备号中提取设备major号码 (使用stat中的st\_dev或者st\_rdev field)。 |
| 32 | [os.makedev(major, minor)](https://www.runoob.com/python/os-makedev.html) 以major和minor设备号组成一个原始设备号 |
| 33 | [os.makedirs(path[, mode])](https://www.runoob.com/python/os-makedirs.html) 递归文件夹创建函数。像mkdir(), 但创建的所有intermediate-level文件夹需要包含子文件夹。 |
| 34 | [os.minor(device)](https://www.runoob.com/python/os-minor.html) 从原始的设备号中提取设备minor号码 (使用stat中的st\_dev或者st\_rdev field )。 |
| 35 | [os.mkdir(path[, mode])](https://www.runoob.com/python/os-mkdir.html) 以数字mode的mode创建一个名为path的文件夹.默认的 mode 是 0777 (八进制)。 |
| 36 | [os.mkfifo(path[, mode])](https://www.runoob.com/python/os-mkfifo.html) 创建命名管道，mode 为数字，默认为 0666 (八进制) |
| 37 | [os.mknod(filename[, mode=0600, device])](https://www.runoob.com/python/os-mknod.html) 创建一个名为filename文件系统节点（文件，设备特别文件或者命名pipe）。 |
| 38 | [os.open(file, flags[, mode])](https://www.runoob.com/python/os-open.html) 打开一个文件，并且设置需要的打开选项，mode参数是可选的 |
| 39 | [os.openpty()](https://www.runoob.com/python/os-openpty.html) 打开一个新的伪终端对。返回 pty 和 tty的文件描述符。 |
| 40 | [os.pathconf(path, name)](https://www.runoob.com/python/os-pathconf.html) 返回相关文件的系统配置信息。 |
| 41 | [os.pipe()](https://www.runoob.com/python/os-pipe.html) 创建一个管道. 返回一对文件描述符(r, w) 分别为读和写 |
| 42 | [os.popen(command[, mode[, bufsize]])](https://www.runoob.com/python/os-popen.html) 从一个 command 打开一个管道 |
| 43 | [os.read(fd, n)](https://www.runoob.com/python/os-read.html) 从文件描述符 fd 中读取最多 n 个字节，返回包含读取字节的字符串，文件描述符 fd对应文件已达到结尾,  返回一个空字符串。 |
| 44 | [os.readlink(path)](https://www.runoob.com/python/os-readlink.html) 返回软链接所指向的文件 |
| 45 | [os.remove(path)](https://www.runoob.com/python/os-remove.html) 删除路径为path的文件。如果path 是一个文件夹，将抛出OSError; 查看下面的rmdir()删除一个 directory。 |
| 46 | [os.removedirs(path)](https://www.runoob.com/python/os-removedirs.html) 递归删除目录。 |
| 47 | [os.rename(src, dst)](https://www.runoob.com/python/os-rename.html) 重命名文件或目录，从 src 到 dst |
| 48 | [os.renames(old, new)](https://www.runoob.com/python/os-renames.html) 递归地对目录进行更名，也可以对文件进行更名。 |
| 49 | [os.rmdir(path)](https://www.runoob.com/python/os-rmdir.html) 删除path指定的空目录，如果目录非空，则抛出一个OSError异常。 |
| 50 | [os.stat(path)](https://www.runoob.com/python/os-stat.html) 获取path指定的路径的信息，功能等同于C API中的stat()系统调用。 |
| 51 | [os.stat\_float\_times([newvalue])](https://www.runoob.com/python/os-stat_float_times.html) 决定stat\_result是否以float对象显示时间戳 |
| 52 | [os.statvfs(path)](https://www.runoob.com/python/os-statvfs.html) 获取指定路径的文件系统统计信息 |
| 53 | [os.symlink(src, dst)](https://www.runoob.com/python/os-symlink.html) 创建一个软链接 |
| 54 | [os.tcgetpgrp(fd)](https://www.runoob.com/python/os-tcgetpgrp.html) 返回与终端fd（一个由os.open()返回的打开的文件描述符）关联的进程组 |
| 55 | [os.tcsetpgrp(fd, pg)](https://www.runoob.com/python/os-tcsetpgrp.html) 设置与终端fd（一个由os.open()返回的打开的文件描述符）关联的进程组为pg。 |
| 56 | [os.tempnam([dir[, prefix]])](https://www.runoob.com/python/os-tempnam.html) 返回唯一的路径名用于创建临时文件。 |
| 57 | [os.tmpfile()](https://www.runoob.com/python/os-tmpfile.html) 返回一个打开的模式为(w+b)的文件对象 .这文件对象没有文件夹入口，没有文件描述符，将会自动删除。 |
| 58 | [os.tmpnam()](https://www.runoob.com/python/os-tmpnam.html) 为创建一个临时文件返回一个唯一的路径 |
| 59 | [os.ttyname(fd)](https://www.runoob.com/python/os-ttyname.html) 返回一个字符串，它表示与文件描述符fd 关联的终端设备。如果fd 没有与终端设备关联，则引发一个异常。 |
| 60 | [os.unlink(path)](https://www.runoob.com/python/os-unlink.html) 删除文件路径 |
| 61 | [os.utime(path, times)](https://www.runoob.com/python/os-utime.html) 返回指定的path文件的访问和修改的时间。 |
| 62 | [os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]])](https://www.runoob.com/python/os-walk.html) 输出在文件夹中的文件名通过在树中游走，向上或者向下。 |
| 63 | [os.write(fd, str)](https://www.runoob.com/python/os-write.html) 写入字符串到文件描述符 fd中. 返回实际写入的字符串长度 |

Os包内path模块的操作方法，需要熟记

|  |  |
| --- | --- |
| os.path.abspath(path) | 返回绝对路径 |
| os.path.basename(path) | 返回文件名 |
| os.path.dirname(path) | 返回文件路径 |
| os.path.exists(path) | 路径存在则返回True,路径损坏返回False |
| os.path.getmtime(path) | 返回最近文件修改时间 |
| os.path.getctime(path) | 返回文件 path 创建时间 |
| os.path.getsize(path) | 返回文件大小，如果文件不存在就返回错误 |
| os.path.isabs(path) | 判断是否为绝对路径 |
| os.path.isfile(path) | 判断路径是否为文件 |
| os.path.isdir(path) | 判断路径是否为目录 |
| os.path.islink(path) | 判断路径是否为链接 |
| os.path.join(path1[, path2[, ...]]) | 把目录和文件名合成一个路径 |
| os.path.realpath(path) | 返回path的真实路径 |
| os.path.samefile(path1, path2) | 判断目录或文件是否相同 |
| os.path.split(path) | 把路径分割成dirname和basename，返回一个元组 |
| os.path.splitext(path) | 分割路径，返回路径名和文件扩展名的元组 |

练习：

目录下有这些文件：

A1.txt

B2.txt

C1.doc

D1.excel

任务1-将该目录下的文件按照后缀进行分类，然后分别新建且放入不同的文件夹内，比如txt文件放入txt目录下等