# 注释

* 分类：
* 单行注释，语法：# 注释内容
* 单行注释可以单独成行，也可以放在一条具体的代码语句后面（这个时候至少打两个空格）
* 不能跨行，只能单行
* #与注释内容之间有一个空格(不加也可以，但是编译器会有警告，加上一个空格是一种良好的编程习惯)
* 多行注释，语法包括两种：
* “””

第一行注释

第二行注释

第三行注释

“””(这个是英文版本下的双引号)

* ‘’’

注释1

注释2

注释3

‘’’(这个是英文版下的单引号)

# 变量

* 变量的作用：
* 告之系统要开辟一段存储空间存放指定的数值并且给这个数值当前所在的内存空间取一个名字，就是变量的名字
* 变量的命名规则：
* 由字母(大小写均可)，数字，下划线组成
* 不能由数字开头
* 不能使用内置的关键字
* 严格区分大小写
* 常见的内置关键字:None and as assert def del elif except finally from global import in is lambda nonlocal not or pass raise try with yield
* 变量命名习惯，风格：
* 见名知义
* 大驼峰：变量中的每个单词首字母都大写，如：MyName
* 小驼峰：第二个(包括第二个)以后的单词首字母大写，如：myName
* 下划线：组成变量的每个单词之间用下划线，如：my\_name
* 定义变量：变量名 = 值
* 注意：在c系列语言中定义变量要指定变量的类型，但是在python中无需指定类型，而且字符串赋值给变量的时候c系列语言中使用的是双引号，而python中单引号和双引号都是可以去引用字符串的，但是在字符串之间有单引号，我们用双引号去引起来更方便，字符串中有双引号，我们用单引号括起来更方便

# 数据类型

* 数值
* int（整型）
* float（浮点型）
* 布尔型
* True（真）
* False（假）
* str（字符串）
* list（列表）
* tuple（元组）
* set（集合）
* dict（字典）
* 可以用type()函数进行检测数据类型

# 格式化输出

* 常用格式符号：
* %s：字符串
* 也可以用来输出整数小数
* %d：有符号十进制整数
* %06d，表示输出的整数显示的位数，不足补齐0
* %f：浮点数
* %.2f表示保留两位小数
* %c：单个字符
* %u：无符号十进制整数
* %o：八进制整数
* %x：十六进制整数(小写ox)
* %X：十六进制整数(大写oX)
* %e：科学计数法(小写’e’)
* %e：科学计数法(大写’E’)
* print()里面也可以用print(f’...{变量1}....{变量2}...’)这种格式去书写

# 转义字符

* \n：换行
* 注意print()函数会自动换行的（系统默认）
* 系统默认:print(‘string’, end=“\n”)，可以改成print(‘string’, end=“\t”)
* \t：制表符，一个tab键（4个空格）的距离

# 输入

* 输入特点：
* 当程序指向到input时，等待用户输入，输入完成后才会继续向下执行
* 在python中，input接收到用户的信息后，一般存储为变量，方便使用
* 在python中,input把接收到的任意用户输入的数据都当做字符串去处理
* 转换数据类型的函数：
* int(x)：将x转换成一个整型
* float(x)：将x转换成浮点类型
* str(x)：将x转换成字符串类型
* eval(str)：用来计算在字符串中的有效python表达式，并且返回一个对象
* tuple(s)：将序列s转换成一个元组
* list(s)：将序列s转换成一个列表
* python交互式开发，可以在python console控制台进行书写，此时是放在临时内存里面的，是不会写入硬盘的，在控制台可以快速测试程序，一些简单的小程序是可以在控制台进行验证的，如果想退出的话，那么直接关闭项目然后再进入项目就可以了

# 运算符

* 算术运算符
* +
* -
* \*
* / # python中算的是多少就是多少
* // # 整除，这里和c系语言有区别，值得关注一下
* %
* \*\* # 指数
* () # 小括号
* 优先级：() 高于 \*\* 高于 \* / // % 高于+ -
* 算数运算符的优先级始终大于赋值（包括符合）运算符的优先级
* 赋值运算符
* =
* 单变量赋值

如：num1 = 1

* 多个变量赋值

如：num1, float1, str1 = 1, 0.5, “TOM”

* 多变量赋相同值

如：num1 = num2 = num3 = 100

* 复合赋值运算符
* +=
* -=
* \*=
* /=
* //=
* %=
* \*\*=
* 先算复合赋值运算符右侧的表达式
* 再算复合赋值运算的算数运算
* 最后赋值运算
* 比较运算符
* ==
* ！=
* >
* <
* >=
* <=

（返回值为True 和 False）

* 逻辑运算符
* and
* or
* Not

# 条件语句

* if语法：

if 条件：

条件成立执行的代码1

条件成立执行的代码2

............

* 注意了，python语言中是通过缩进来标识语句块
* if - else语法：

if 条件：

条件成立执行的代码1

条件成立执行的代码2

条件成立执行的代码3

....................

else：

条件成立执行的代码4

条件成立执行的代码5

条件成立执行的代码6

* 多重判断：

if 条件1：

条件1成立执行的代码7

条件1成立执行的代码8

elif 条件2： # elif是else if的缩写

条件2成立执行的代码9

条件2成立执行的代码10

......................

else：

以上条件都不成立时所执行的代码

* 与c系列语言有所区别，python语言中条件and可以合为一体

如：(age >= 18) and (age < 60) 和 18 <= age < 60 是等价的

# 三目运算符

* 语法：

条件成立时执行的表达式 if 条件 else 条件不成立时候执行的表达式

# while循环

* while循环语法：

While 条件：

条件成立执行的代码1

条件成立执行的代码2

.......................

* while - else循环语法：

while 条件：

条件成立时重复执行的代码

else：

循环正常结束后要执行的代码

* 为什么要引入while - else这种代码？
* 因为while语句下面的一条语句如果没有else这种语法的话，那么不管循环是不是正常终止（有可能break中断循环），都会执行。但是如果是while - else语句的话，只有循环正常结束的时候才能执行else的语句

# for循环

* for循环语法：

for 临时变量 in 序列：

重复执行的代码1

重复执行的代码2

......................

* for else语法：

for 临时变量 in 序列：

重复执行的代码1

重复执行的代码2

else：

循环正常结束的时候所执行的代码

# 字符串

* 字符串是不可变数据类型
* 三种表示字符串的方式:
* 单引号：’TOM’
* 双引号：”TOM”
* 三引号：
* ‘’’I am TOM’’’
* “””I am TOM”””
* 三引号是可以换行的
* 下标（和c系语言一样）
* 什么是切片：就是指对操作的对象截取其中的一部分的操作。字符串，列表，元组都支持切片操作
* 语法：序列[开始位置下标：结束位置下标：步长]
* 不包含结束位置下标对应的数据，正负整数都是可以的
* 左闭右开
* 步长是选取间隔，正负整数均可，默认步长为1
* 选取方向与步长冲突则会出错
* 字符串操作方法：
* 查找：
* find()函数：
* 语法：字符串序列.find（子串，开始位置下标，结束位置下标）
* 开始和结束位置下标可以省略，表示在整个字符串进行查找
* 作用：find()：检测某个子串是否包含在这个字符串中，如果在这个字符串中的话，返回子串开始位置的下标，否则返回-1
* index()函数：
* 语法：字符串序列.index（子串，开始位置下标，结束位置下标）
* 开始和结束位置下标可以省略，表示在整个字符串进行查找
* 作用：index()；检测某个字符串在母字符串中是否存在，如果存在的话，返回第一个子字符串在母字符串中的下标位置，否则报错
* count()函数：
* 语法：字符串序列.count（子串，开始pos，结束pos）
* 开始位置和结束位置的下标可以省略，表示整个字符串查找
* 作用：count()函数：统计子字符串在母字符串中出现的次数，如果存在的话，就返回出现的次数，否则返回0
* rfind()函数：与find()功能相同，但是从右开始差查找
* rindex()函数:与index()功能相同，但是从右边开始查找
* 字符串修改方法：
* replace()函数：
* 语法：字符串序列.replace(旧字符串，新字符串，替换次数)
* 如果替换次数不写，表示默认替换所有字符串，如果替换次数超过旧字符串的个数，也即替换所有，如果少于旧字符串的个数的话，就按照实际的来替换
* split()函数：
* 语法：字符串序列.split(分割字符，num)
* 将来返回的数据是num + 1个
* 返回一个列表
* join()函数:用一个字符或者子串合并字符串，即是将多个字符串合并为一个新的字符串
* 语法：字符或者是子串.join(多子串组成的序列)
* 大小写转换函数：
* 字符串序列.capitalize()：将字符串的首字母大写
* 字符串序列.title()：将字符串中的每个单词的首字母都转化成大写
* 字符串序列.lower()：将字符串中所有的大写字母转换成小写
* 字符串序列.upper()：将字符串中所有的小写字母转换成大写
* 删除首尾空白方法：
* lstrip()：字符串序列.lstrip()，删除字符串左边的空白
* rstrip()：字符串序列.rstrip()，删除字符串右边的空白
* strip()：字符串序列.strip()，删除字符串两边的空白
* 设置字符串对齐方法：
* ljust()：字符串序列.ljust(域宽，填充字符)
* rjust()：字符串序列.rjust(域宽， 填充字符)
* center()：字符串序列.center(域宽，填充字符)
* 注意：填充字符默认为空格
* 检测字符串开头和结尾的单词方法：
* strartswith()函数：字符串序列.startswith(子串，开始位置下标，结束位置下标)
* 如果起止位置不写的话，则默认整个字符串
* 返回bool值
* endswith()函数：字符串序列.endswith(子串，开始位置下标，结束位置下标)
* 如果起止位置不写的话，则默认整个字符串
* 返回bool值
* 判断数字，字母，空格方法：
* isalpha()函数：字符串序列.isalpha()，如果字符串中的字符全部是字母的话，返回True，否则返回False
* isdigit()函数：字符串序列.isdigit()，如果字符串中的字符全部是数字的话，返回True，否则返回False
* isalnum()函数：字符串序列.isalnum，如果字符串中的字符全部是数字或字母或字母与数字的组合的话，返回True，否则返回False
* isspace()函数：如果字符串中的字符全部是空格的话，返回True，否则返回False

# 列表

* 列表是可变数据类型
* python中的列表与c系列语言中的数组相似
* 相同点：都可以存储大批量数据
* 不同点：数组只能存储同种类型的数据，然而列表可以存储不同类型的数据
* 形式：[数据1，数据2，数据3，...........]
* 如何访问列表的元素？
* 下标
* 函数:
* index()：列表序列.index(待查找字符串，开始下标，结束下标)

查找成功，返回第一个找到的字符串的下标，查找失败报错

* count():列表序列.count(待查找字符串，开始下标，结束下标)

查找成功返回字符串的个数，没有找到返回0

* len()函数:len(列表序列) 返回列表长度
* 判断列表元素是否存在:
* in
* not in
* 此二者均返回bool值
* 列表增加数据的三种函数：
* append():列表序列.append()
* 如果追加的是一个序列的话，就会把序列整体给添加到原序列中，此时相当于里面有部分二维数组了
* extend()函数:列表序列.extend()
* 如果追加的是一个序列的话，就会把序列中的字符一个一个给添加到原序列中
* Insert()函数：列表序列.insert(位置下标，数据)
* 删除列表中的数据的四种函数
* del 目标列表序列
* del还可以删除指定下标的数据
* pop()函数：列表序列.pop(下标)：删除指定下标处的数据，如果没有指定下标的话，就默认删除最后一个数据，无论是按照下标删除还是默认删除，最后都会返回被删除的数据
* remove()函数:列表序列.remove(数据)，删除指定的数据
* 如果数据有两项的话，删除第一个
* clear()函数：列表序列.clear()，清空列表
* clear与del是有区别的,clear是清空，但是会保留一个壳子；但是del删除后就啥都没有了
* 列表修改：
* 修改指定下标处的数据
* reverse()函数：列表序列.reverse()
* sort()函数：列表序列.sort(key=None, reverse=False)
* reverse表示排序规则,reverse=True表示降序，reverse=False表示升序，而且不指定reverse的bool值的话，默认为升序
* 列表复制：

列表序列.copy() 我们往往会将复制得到的新的列表赋值给一个新的列表

* 返回一个完全一样的列表

# 元组

* 元组和列表一样都可以存着批量数据，但是，它不能修改
* 多个数据元组：t1 = (10, 20, 30)
* 单个数据元组：t2 = (10,)
* 如果定义的元组只有一个数据的话，那么这个数据后面最好也添加逗号，否则数据类型为唯一的这个数据的数据类型
* 元组中嵌套列表还是可以进行修改的!!!!!!!!
* 元组常见的操作：
* 按下标查找数据
* 元组序列.index()
* 元组序列.count()
* len(元组名)

# 字典

* 字典为可变类型
* 字典和c系语言中的struct结构体数组类似
* 字典里面的数据是以键值对的形式出现的，字典数据和数据顺序没有关系，即字典不支持下标，也就是无论后期数据顺序如何变化，只需要按照对应的键的名字查找数据即可
* 字典特点：
* 符号为大括号
* 数据为键值对形式出现
* 各个键值对之间用逗号隔开
* 创建字典：
* 有数据
* 如：dict1 = {‘name’: ‘Tom’, ‘age’: 20, ‘gender’: ‘nan’}
* 无数据
* 法1：dict2 = {}
* 法2：dict3 = dict() # 这个时候就是调用相应的函数了
* 字典新增：
* 写法：字典序列[key] = 值
* 如果key存在的话，就修改这个key所对应的值

如果key不存在的话，就新增加这个键值对

* 字典删除：
* del 字典序列：删除整个字典
* del 字典序列[key]：删除其中一个键值对
* clear()：字典序列.clear() 清除字典
* 字典修改：详情见字典新增里面的内容
* 字典查找
* [key]进行查找 如：print(dict1[‘name’])
* 存在key值，打印
* 不存在key值，报错
* 函数查找
* get()函数
* 字典序列.get(key, 默认值)
* 如果查找的key存在的话就返回对应的键值，如果查找的key不存在的话返回第二个默认参数，如果没有指定第二个默认参数，就返回None
* Keys()函数
* 字典序列.keys()
* 查找字典序列中所有的key值，返回一个可迭代对象
* Values()函数
* 字典序列.values()
* 查找字典序列中所有的键值，返回一个可迭代对象
* Items()函数
* 字典序列.items()
* 查找字典中所有的键值对，返回一个可迭代对象，里面的数据是元组，元组数据1是key，元组数据2是键值

# 集合

* 集合是可变数据类型
* 集合的创建
* 集合的创建可使用{}和set()函数
* 单个数据只能用set()函数,set()函数最多只能添加一个数据
* 多个数据用{},单个{}表示创建一个字典
* 如果集合是空集的话，只能用set()函数，因为{}表示创建一个空的字典
* 集合是无序的，不能对其用下标进行操作
* 集合增加数据
* add()函数：集合序列.add(要增加的单一数据)
* 如果增加的是已经有的数据，则不会有任何变化
* 如果增加的是没有的数据，就会添加上去
* 而且要增加的数据只能是单一数据，不能是一个序列
* update()函数：集合序列.update(要增加的序列)
* 与add()函数前两条规则一样
* 要增加的数据只能是一个序列，不能是单一数据
* 集合删除数据
* remove()函数：集合序列.remove() 删除集合中指定的数据，数据不存在就报错
* discard()函数：集合序列.discard() 与remove()函数有区别，当要删除的数据不存在的时候，不会报错
* pop()函数：随机删除某个数据，并且返回这个数据

# 公共操作

* 运算符
* + ：连接作用
* 支持的容器类型:字符串、列表、元组
* \* ：复制作用
* 支持的容器类型：字符串、列表、元组
* in ：元素是否存在
* 支持的容器类型：字符串、列表、元组、字典
* not in ：元素是否不存在
* 支持的容器类型：字符串、列表、元组、字典
* len()函数：计算容器中的数据的个数 所有容器都支持
* del()函数：删除指定的内容 所有容器都支持
* max()和min()函数：查找指定序列中的最大最小值 所有容器都支持
* range(start, end, step)函数：生成从start到end的数字，步长为step，供for循环使用，返回一个可迭代对象
* 左闭右开（不包括end）
* step省略则默认为1
* start省略，默认从0开始
* enumerate(可遍历对象, start = 0)函数
* 返回结果是元组
* 元祖的第一个数据是原迭代对象的数据对应的下标
* 元组的第二个数据是原迭代对象的数据
* start可以修改，不修改的话默认从0开始
* 容器转换函数
* Tuple()函数可以将list和set序列转换成tuple类型
* List()函数可以将tuple和set序列转换成list类型
* Set()函数可以将tuple和list序列转换成set类型，多余的数据自动滤掉只保留一个

# 推导式

* 列表推导式
* 作用：用一个表达式创建一个有规律的列表或者是控制一个有规律的列表
* 列表推导式又叫做列表生成式
* [xx for xx in range()]
* 字典推导式
* {xx1: xx2 for ... in ...}
* 集合推导式
* {xx for xx in ...}

# 函数

* 函数定义形式：

def 函数名(参数)：

“””

说明文档

“””

代码1

代码2

......

* 调用函数：
* 调用形式：函数名（参数表（可无））
* 在python中，函数必须先定义后使用
* help()函数可以查看函数的作用
* 在函数中如果要修改全局变量的话，要加关键字global
* 参数：
* 位置参数，就是实参与形参一一对应的
* 关键字参数，这个时候传递的参数的顺序就没有那么多限制了
* 缺省参数，这时候有默认值的参数可以不用传入
* 不定长参数，不确定传入的参数的个数
* \*args：包裹位置参数
* \*\*kwargs：包裹关键字参数
* 返回一个字典
* 拆包：
* 元组拆包
* 字典拆包
* 得到的是keys值
* 交换变量的值
* 设置一个临时变量，和c系语言中的变量交换的方式是一模一样的
* 直接交换
* 如：a, b = 1, 2

a, b = b, a

* lambda表达式:
* 语法：lambda: 参数列表: 表达式
* lambda表达式的参数可有可无，函数的表达式在lambda表达式中完全适用
* lambda表达式可以接受任意数量的参数但是只能有一个返回值
* 参数形式：
* 无参数
* 一个参数
* 默认参数
* 可变参数
* 应用：
* 带判断的lambda
* 列表数据按照字典的key对应的键值进行排序

# 二十、内置高阶函数

* 何为高阶函数：高阶函数就是把函数作为参数传到另外一个函数里面
* map()函数:map(func, list)
* 意思就是将func函数的功能作用到list列表中的每一个元素
* 返回一个map类对象
* 打印返回值得到一个列表的地址值，用list()函数可以将其强制类型转化成列表输出
* reduce()函数:reduce(func, list)
* 要求func函数必须有两个参数
* 返回一个reduce类的对象
* 用list函数可以转化
* 每次func计算的结果都会和列表的下一个元素做累积运算
* filter()函数:filter(func, list)
* 返回一个filter类对象
* 用list函数可以进行转化

# 二十一、类和对象

* 创建一个类：
* class 类名():

代码

.......

* 类的命名规则遵守大驼峰的命名习惯
* 创建一个对象
* 对象名 = 类名()
* 类里面的函数括号里面的self就是指代调用该函数的对象，有点类似于c++中成员函数的隐含参数\*this指针
* 添加和获取对象属性
* 类外添加和获取属性
* 类内添加和获取属性
* \_\_xx\_\_():这样的函数在python语言中是具有特殊功能的函数
* \_\_init\_\_(self):是在类体内定义的用来初始化对象属性的成员函数
* 在创建一个对象的时候默认调用，不需要用户手动调用
* init中的self参数不需要程序猿去传递，编译器会自动把当前对象的引用进行传递
* \_\_str\_\_(self):用来在打印对象的时候返回指定的信息
* 类体中定义了str函数，打印对象的时候返回指定的信息
* 类体中没有定义str函数，打印对象的时候返回的是地址值
* \_\_del\_\_(self):用来在删除对象的时候完成指定的操作，返回指定的信息
* 继承
* 基类/父类：

Class 基类名：

代码1

代码2

.........

* 派生类/子类

Class 派生类名字(基类名):

代码1

代码2

.........

* 单继承
* 多继承
* 多继承的时候如果遇到同名属性或者方法，默认使用第一个基类的，这是指子类没有对属性或者方法进行修改的前提下进行的
* 子类重写父类同名方法和属性:这样子类对象调用同名函数的话就是调用子类的了，而不是默认第一个基类中的函数
* \_\_mro\_\_()函数：用来查看子类与其所有父类之间的关系
* 子类调用父类同名函数
* 想要子类对象调用父类中的同名成员函数，可以将子类对父类的初始化init()函数和同名函数进行封装，然后再通过子列对象调用封装好了的这个函数，进而通过这个封装好了的成员函数调用父类的同名函数，也可以用super()函数
* 调用父类的方法：父类.相关函数或者是数据成员
* Super()函数只能调用父类
* 私有权限
* 方式：数据成员或者是成员函数前面加上两个下划线\_\_
* 对象不能访问私有数据成员和成员函数
* 子类无法继承父类私有的数据成员和成员函数
* 但是私有数据成员和成员函数是可以在类内通过特定的成员函数进行访问和修改
* 在python中，一般用get\_xx()函数获取私有数据，用set\_xx()函数修改私有数据

# 二十二、多态

* 实现步骤
* 定义父类，并提供公共方法
* 定义子类，并重写父类方法
* 传递子类对象给调用者，可以看到不同子类执行的效果是不同的
* 属性
* 类属性
* 类属性就是直接在类体里面书写类似于变量名 = 值这种形式的代码
* 类属性可以在里面添加
* 类属性也可以在外面用类名进行添加
* 类属性就是类对象所拥有的属性，它被该类所有的实例对象所共有
* 类属性可以通过类对象或者是实例对象进行访问
* 记录某项数据并要求始终保持一致的时候，定义类属性
* 实例属性要求每一个对象为其开辟一段存储空间来记录数据，而类属性则是所有的对象所共有的，只占一份存储空间，更加节省空间
* 修改类属性只能用类名不能用对象名，用对象名的话就是创建一个属性实例了
* 实例属性
* 添加
* 可以在\_\_init\_\_()函数中进行添加
* 也可以在类体外用实际对象名进行添加
* 实际属性每个对象独有
* 类方法：
* 当方法中需要使用类对象时（如访问私有类属性等）,就用类方法
* 类方法一般和类属性配合使用
* 定义方式：用@classmethod装饰器对函数进行装饰即可
* 静态方法：
* 减少self,cls等参数的传递，节约内存
* 要用装饰器@staticmethod进行修饰
* 可以用类名来访问，也可以用实际对象来访问

# 二十三、异常

* 语法：

try:

可能发生错误的代码

except:

发生错误时执行的代码

* 捕获指定异常类型
* 捕获一个指定的异常类型

try:

可能发生错误的代码

except 异常类型:

如果捕获到该异常类型执行的代码

* 如果尝试执行的代码的异常类型和要捕获的异常的类型不一致，那么无法捕获异常
* 一般try下面只放一行要执行的代码
* 捕获多个指定的异常类型

try:

可能发生错误的代码

except (异常类型1, 异常类型2, ......):

如果捕获到该异常类型执行的代码

* 捕获异常描述信息

try:

可能发生错误的代码

except (异常类型1, 异常类型2, ......) as result:

print(result)

* 捕获所有的异常

try:

可能发生错误的代码

except Exception as result:

print(result)

* exception是所有异常类的父类
* 异常的else类型

try:

可能发生错误的代码

except Exception as result:

print(result)

else:

try正常执行的时候所执行的代码

* 异常的finally类型

try:

可能发生错误的代码

except Exception as result:

print(result)

else:

try正常执行的时候所执行的代码

finally:

不管try下的代码异常与否，finally下的代码最终都是要执行的

* 命令提示符运行py文件
* 在此电脑里面找到py文件所在目录打开
* 选中目录，删除，输入cmd
* 接着在命令窗口输入python和想要运行的指定的文件
* Tab键可以补齐文件名
* 自定义异常
* 自定义的异常不是语法错误，而是根据程序实际需要来设置的自定义的异常
* 抛出异常的语法：raise 异常类(参数1,参数2,......)

# 二十四、模块

* 模块实际上就是一个后缀名为.py的文件，包含了python类对象和python类语句。模块里面能够定义函数，类，变量，以及可执行的代码
* 导入模块的方式
* import 模块名1,模块名2.....
* from 模块名 import 功能名1,功能名2.......
* from 模块名 import\*
* 如果一个模块文件中有\_\_all\_\_变量的话，当使用from 模块名 import\*这种导入模块方式时，只能导入\_\_all\_\_列表里面的函数
* import 模块名 as 别名
* from 模块名 import 功能名 as 别名
* 制作模块的步骤:
* 定义模块（就是新建立一个py文件）
* 测试模块
* 调用模块
* 模块定位注意事项
* 自己的文件名不要和已有的模块名重复，否则导致模块功能无法正常使用
* 使用from 模块名 import 功能的时候，如果功能名重复，调用到的是最后定义或导入的功能

# 二十五、包

* 所谓的包就是将有联系的功能模块放在一同一个文件夹下，并且在这一个文件夹里面创建一个名字为\_\_init\_\_.py文件，这个文件夹就称之为包
* 注意，新建包后，包的内部会自动创建\_\_init\_\_.py文件，这个文件控制着包的导入行为
* 导入包:
* 方式一：import 包名.模块名
* 方式二：from 包名 import \*,还要在\_\_init\_\_.py文件里面添加\_\_all\_\_ = []来控制允许导入的模块名