**1.**注释：# 表示单行注释 ''' ''' 三个单引号或者三个双引号可以多行注释

**2**.变量：变量是指可以变化的量，不需要事先声明，只要创建赋值之后，就可以使用。

**3**.python数据类型：整型（int）例如：2 浮点型(float) 例如：1.2 复数类型(complex) 例如：-1开根号 布尔型(bool) 例如：True,False None类型，只有一个None值，表示没有。当一个函数没有返回值的时候，会返回None 字符串(str) ,例如’abc’

元组(tuple) 列表(list) 字典(dict) 集合(set) bytes类型 ：表示二进制字符串

**4**.type(),内置函数，可以查看变量的类型。例如：a=12 print(type(a))

**5**.标识符的命名规则与规范：标识符由字母，数字，下划线组成，而且不能以数字开头。严格区分大小写，不能使用关键字。要顾名思义，最好不要使用中文。常见命名规范：1驼峰命名法:第一个单词的首字母小写，以后每个单词的首字母都大写。2大驼峰命名法：每个单词的首字母都大写。在Python里定义类名时，使用大驼峰命名法。3使用下划线连接: user\_name python推荐使用的命名规范。

**6.**输出语句print: print(value, ..., sep=' ', end='\n'),sep = ' ' ==> 缺省参数，如果不传入值，使用默认值。end= '\n' ==>缺省参数，如果不传入值，使用默认是 \n。sep 参数作用是用来规定输出打印时，多个被打印的值之间使用哪个字符进行分割。\n ==>表示换行

end 当执行完一个print语句以后，接着要执行哪个操作。

*print*('hello','mytown',sep='\*',end=' ')  
*print*('world')

1. 格式化输出：%s 字符串的占位符；%d 整型的占位符；%f 浮点型的占位符，默认保留小数点后6位；%.nf：指定保留到小数点后几位；%nd ==> 如果整数不够 n位，默认在前面 n-数字长度 个空格；%-nd ==> 如果整数不够 n位，后面补 n-数字长度 个空格；%0nd ==> 如果整数不够n位，在前面补 n-数字长 个0；%% ==> 表示一个 %；%c ==> 找到指定数字对应的编码；%o ==> 以八进制的形式输出。 十进制的10转换成为八进制输出；%x ==> 以十六进制的形式输出；%e / %E 科学计数法（小写'e'）(大写“E”）

*print*('%c'%97)  
*print*('%o'%8)  
*print*('%x'%15)

**8**.输入语句input: input 拿到的执行结果是一个字符串类型的数据;字符串和数字不能做加法运算

**9**.类型转换：使用int内置类转换成为整数：字符串转整数：int('hello') ==>报错

x = int("123.455") ==> 报错

a = '34'

b = int(a)

print(type(a)) ==> 字符串

print(type(b)) ==> 整数

布尔值转整数：int(False) ==> 0；int(True) ==> 1；

浮点数转整数：b = int(123.456) ==> 123

带进制的转换：int 类型将一个字符串转换成为整数时，默认会把字符串当成十进制来看待；print(int('FF',16)) print(int('45',8))

float(x)将x转换为一个浮点数

str(x)将对象 x 转换为字符串

bool(x)：将对象x转换成为布尔值，结果为False的有0，空字符串，None，空元组，空列表，空字典。

eval(str)

用来计算在字符串中的有效Python表达式,并返回一个对象

chr(x)

将一个整数转换为一个Unicode字符

ord(x)

将一个字符转换为它的ASCII整数值

bin(x)

将一个整数转换为一个二进制字符串

oct(x)

将一个整数转换为一个八进制字符串

hex(x)

将一个整数转换为一个十六进制字符串

list(s )

将序列 s 转换为一个列表

tuple(s)

将序列 s 转换为一个元组

**10**.算术运算符：+，-，\*，/，%，//,\*\* 4 \*\* (1/2) ==> 2.0；pow(2,3) ==> 8；sqrt(4) ==> 2

数字和字符串之间不能做除了乘法以外的其他算数运算；

乘法的结果是将字符串重复出现指定的次数；

字符串和字符串之间只能做加法运算，作用是拼接两个字符串

**11**.赋值运算符：等号的左边一定不能是表达式或者常量；连续赋值：x = y = 10 = z ==>报错

m = n = p = 8 print(m, n, p)

x, y, z = 1, 2, 3 print(x, y, z)

可变参数

\* 列表

a, \*b, c, d = 1, 2, 3, 4, 5, 6, 7, 8

print(a, b, c, d) # 1 [2, 3, 4, 5, 6] 7 8

**12**.比较运算符：数字和字符串之间做除了 == 和 != 以外的其他比较运算都会报错；数字和字符串做 == 运算，结果都是False；字符串和字符串之间比较，会按照字符出现的编码顺序逐一进行比较

**13**.逻辑运算符：逻辑与(and):只有所有的运算数都是True,结果才是True;只要有一个是False,结果就是False;and 短路问题，遇到False后面的代码就不再执行;逻辑与运算做(取值时)，取第一个为False的值,如果所有的运算数都是True,取最后一个值;

逻辑或(or):只要有一个运算数是True,结果就是True;只有所有的运算数都是False,结果才是False；or 短路问题，遇到True后面的代码就不再执行；逻辑或运算做(取值时)，取第一个为True的值；如果所有的运算数都是False,取最后一个值；

逻辑非(not)

逻辑运算的结果不一定是布尔值;c = True and True and None and True

print(c) # None

1. 位运算符：按位取反~：~n ==> -(n+1)；左移<< n << m ==> n \* 2^m；右移>> n >> m ==> n / 2^m；
2. 三元表达式：a=num if num>num1 else num1
3. 九九乘法表：分别用for循环和while循环写

i=0  
*while* i<9:  
 i+=1  
 j=0  
 *while* j<i:  
 j+=1  
 *print*('%d\*%d=%d'%(j,i,j\*i),end=' ')  
 *print*()  
*for* i *in range*(1,10):  
 *for* j *in range*(0,i):  
 j+=1  
 *print*('%d\*%d=%d'%(j,i,j\*i),end=' ')  
 *print*()

1. 字符串：一对单引号或者双引号里面的内容。\：转义字符，作用是让紧跟着\后面的字符转义。在字符串前面加r，表示这是一个原生字符串。
2. 字符串的切片：

b = 'abcdefghijklmn'  
*print*(b[2:8]) *# 从下标为2的元素开始到下标为8的元素(包含2,不包含8)  
print*(b[:]) *# abcdefghijklmn 从头到尾  
print*(b[2:]) *# cdefghijklmn 从第二个开始，截取到最后(包含第二个)  
print*(b[:5]) *# abcde 从开始截取到第5个，不包含第5个  
print*(b[2:10:2]) *# cegi 从第二个开始，截取到第10个，每隔一次获取一个,2指的是步长  
print*(b[::-1]) *# nmlkjihgfedcba 从头到尾，往左取  
print*(b[2:10:-1]) *# 空字符串  
print*(b[10:2:-1]) *# kjihgfed  
print*(b[-8:-1]) *# 从倒数第8个开始，截取到倒数第1个(包含倒数第8,不包含倒数第一)*

1. Len()函数可以用来获取字符串，元组，列表，字典的长度。

Find()

如果存在就返回该内容在字符串中第一次出现的开始位置索引值，如果不存在，则返回-1。可以指定开始区间和结束区间，print(a.find(‘w’,1,4))。Index()跟find()方法一样，find方法未找到时，返回-1;而index方法未找到时，会报一个异常。

1. startswith:判断是否是以某个字符开头；endswith:判断是否是以某个字符结尾；isalpha:判断是否是纯字母；isdigit:判断是否是纯数字；isalnum判断是否是由数字和字母混合组成的。Isspace:判断是否全部是由空格组成的

temp='hello'  
*print*(temp.isalpha())  
*print*(temp.isdigit())  
*print*(temp.isalnum())

1. count()方法：计算某个字符在字符串中出现的个数，可以指定开始和结束区间。

*# 计算出现次数:count 可以指定start和end用来查找指定范围字符出现的次数  
print*('今天天气好晴朗，处处好风光呀好风光'.count('好'))  
*print*('今天天气好晴朗，处处好风光呀好风光'.count('好', 0, 8))

1. replace()方法可以用某个字符来替换字符串中的另一个字符，但是原字符串不会改变，因为字符串是不可变的，会生成一个新的字符串来保存修改后的结果。

a='hello world'  
x=a.replace('l','j')  
*print*(x)

1. split()方法，用来分隔字符串，得到的结果是一个列表。以字符串中的某个字符作为分隔点，如下：

z = 'hello\_hi\_how\_good\_better\_bad\_worse'  
a = z.split('\_')  
*print*(a)

1. splitlines()方法，换行分隔；如下：

m = 'hello\r\nhi\r\ngood\r\nbetter\r\n'  
*print*(m.splitlines())  
*print*(m.split('\r\n'))

['hello', 'hi', 'good', 'better']

['hello', 'hi', 'good', 'better', '']

1. partition()方法，是以字符串中的某个字符为分隔点，将字符串分为三部分。

*print*('hello'.partition('e'))

('h', 'e', 'llo')

1. 修改大小写，capitalize 将第一个单词的首字母大写；Title:所有单词的首字母都大写；upper:所有字母都大写；lower:所有字母都小写。

*# 空格处理:ljust,rjust,center,lstrip,rstrip,strip  
print*('yes'.ljust(10, "\_")) *# 使用指定的字符左对齐填充到指定长度  
print*('no'.rjust(10, '-'))  
*print*('you'.center(10, '\*'))

*print*(' hello '.lstrip())  
*print*(' hello '.rstrip())  
*print*(' hello '.strip())

*# 字符串拼接:join  
print*('-'.join(['你好', '热', 'hello', 'thanks'])) *# 你好-热-hello-thanks  
print*('\*'.join('abc')) *# a\*b\*c*

1. 创建列表的方法：[]或者list.

*[] 表示一个列表*a = [] *# a 是一个空列表  
  
# 列表里也可以放入数据，每个数据之间使用逗号来进行分割  
# 一个数据我们称之为一个元素  
# 一个列表里可以存放不同类型的数据，但是我们建议一个列表只存一种类型的数据，便于数据的管理*b = ['hello', 7, 'good', 23.45, *True*, 'hi']  
  
*# 使用list创建一个列表*x = *list*() *# 空列表  
print*(x)  
  
*# 还可以使用list创建列表并且插入元素*y = *list*((1, 2, 3, 4, 5))  
*print*(y)  
z = *list*([6, 7, 8, 9, 10])  
*print*(z)

29.列表的增删改查：增加有append(),在列表末尾加一个元素；insert(i,’’),在指定位置之前插入一个；extend(),合并两个列表；删除方法有remove(),删除指定元素；pop(),删除最后一个元素；del[i]，删除指定下标的元素，也可以删除整个变量。

*# 增删改查*names = ['张三', '李四', 'tom', 'jerry', 'chris', 'henry', 'tom', 'tony']  
  
*# 查找数据。len获取长度的  
# 下标：从0开始，每个元素都有一个对应的位置  
# 下标的取值范围 [0,len)  
print*(names[3])  
*print*(names[-1])  
*print*(names[*len*(names) - 1])  
  
*# name = input('请输入您要查找的姓名')  
# if name in names:  
# value = names.index(name)  
# print('您要查找的姓名编号是%d' % value)  
# else:  
# print('您查找的姓名不存在')  
  
# value = names.rindex('helen') 列表没有rindex方法  
# names.find('helen') 列表没有find方法  
  
  
# 修改数据  
# 列表是可变的*names[1] = '尼古拉斯赵四'  
*print*(names)  
  
*# 往列表里加入数据  
# append 在列表的最后面添加一个数据*names.append('stark')  
*print*(names)  
*# insert 在指定位置插入数据*names.insert(2, '迪迦奥特曼')  
*print*(names)  
*# extend可以加入多个数据，但是需要的参数是可迭代对象*names.extend(('jack', 'rose'))  
*print*(names)  
*# 使用 + 运算符可以直接加入一个列表*names += ['苏克', '贝塔']  
*print*(names)  
  
*# 删除数据  
# pop 方法可以删除最后一个元素  
# pop 方法还有一个返回值，返回被删掉的数据*result = names.pop()  
*print*(result)  
*print*(names)  
*# remove 可以移除指定的元素  
# 如果被移除的元素不在列表里，remove方法会报错*names.remove('迪迦奥特曼')  
*print*(names)  
*# names.remove('迪迦奥特曼')  
# del 运算符可以删除指定位置的元素  
del* names[3]  
*print*(names)  
*# del 运算符可以删除一个变量  
# a = 2  
# del a  
# print(a + 1)  
  
# 列表不支持减法运算  
# names - ['henry']  
# print(names)*

1. 遍历列表：

names = ['张三', '尼古拉斯赵四', 'tom', 'chris', 'henry', 'tom', 'tony', 'stark', 'jack', 'rose', '苏克']

for i ,j in enumerate(names):

print('第%d个元素是%s'%(i,j))

1. 求100以内所有的素数

*# 求100以内所有的素数  
for* i *in range*(2,101):  
 count=0  
 *for* j *in range*(2,i):  
 *if not* i%j:  
 count=1  
 *break  
 if* count==0:  
 *print*(i,end=' ')  
*print*()

1. 求水仙花数

*# 求水仙花数（一个三位数，每个位上数字三次方之和等于这个数本身）*num=*int*(*input*('请输入一个三位数：'))  
a=num//100  
b=num%100//10  
c=num%10  
*if* a\*\*3+b\*\*3+c\*\*3==num:  
 *print*('这个数是水仙花数')  
*else*:  
 *print*('这个数不是水仙花数')

1. 给办公室分配老师问题：

一个学校，有3个办公室，现在有8位老师等待工位的分配，请编写程序，完成随机的分配

import random

offices=[[],[],[]]

teachers=['A','B','C','D','E','F','G','H','I']

for teacher in teachers:

index=random.randint(0,2)

offices[index].append(teacher)

for i,j in enumerate(offices):

print('第%d个办公室有%d个人，分别是:'%(i,len(offices[i])),end=' ')

for x in j:

print(x,end=' ')

print()

1. math模块：

# import math

#

# # 向上取整

# print(math.ceil(23.000001)) # 24

# # 向下取整

# print(math.floor(12.98)) # 12

# print(math.fabs(-21)) # 21.0

# print(math.factorial(5)) # 120

# # print(math.isfinite(5 / 0))

# print(math.pi) # 3.141592653589793

# print(math.sin(math.pi / 6))

# print(math.cos(math.pi / 6))

# print(math.tan(math.pi / 2))

1. random模块：

# import random

# # randint(a,b) 生成[a,b]的随机整数

# # randrange(a,b) 生成[a,b)的随机整数

random() 生成 [0,1) 的随机浮点数

# # choice从列表里速记选取一个数据

# print(random.choice(['hello', 'hi', 'how', 'are', 'you']))

# # sample从列表里随机选择指定个数的元素

# print(random.sample(['hello', 'hi', 'how', 'are', 'you'], 3))

1. 元组的使用：

元组和列表很像，也是用来保存一堆数据

# 区别：

# 1. 表示方式不一样。列表使用[]或者list();元组使用()或者tuple

# 2. 列表是可变的，元组是不可变的！！！

a = (11,)

只有一个元素的元组，必须要在元素后写一个逗号。

a = 1, 2, 3 # 如果是逗号，会把它当做一个元组

print(a)

b = (1, 2, 3)

# int() float() str() bool() list() tuple()

c = tuple((4, 5, 6))

print(c)

d = tuple([4, 5, 6])

print(d)

# 如何定义只有一个数据的元组(元素,)

y = ('hello',)

z = y

print(y)

元组是一个不可变的数据类型，它不支持添加元素和修改以及删除元素的操作

元组是一个不可变的数据类型，它不支持添加元素和修改以及删除元素的操作

# x.append('hi')

# x[0] = 'yes'

# del x[0]

# print(x)

y = y + ('hi', 'good')

print(y)

print('y修改以后的地址是%X' % id(y))

print(z)

1. 可变数据类型和不可变数据类型：

不可变数据类型：数据是不可变的，如果要修改的话，内存地址会发生变换

# 常见的不可变数据类型: 字符串str 数字 int float 元组 tuple

可变数据类型：数据可修改，改变数据内存地址不会变换

# 可变的数据类型: 列表list 字典dict

37.浅拷贝：copy方法可以浅拷贝一个新的数据，指向不同的内存地址；python 有一个copy模块模块，可以拷贝一个数据。 copy模块里有一个copy方法，可以实现浅拷贝

xxxx = copy.copy(names)。

1. 浅拷贝和深拷贝的区别：浅拷贝就是在一个列表里，元素都是一个单独的元素，不是又一个列表，这个时候，浅拷贝的话，两个列表指向不同的地址，改变新列表的元素，不会对原列表有影响。当新列表里面有元素是其他列表时，此时浅拷贝之后，里面元素是列表的地址和原列表地址是同一个地址，不变。只有深拷贝的话，才会将这个也改变掉。浅拷贝：import copy,copy.copy;深拷贝：copy.deepcopy()

38.字典数据类型

# 字典也可以保存多个数据，保存数据的格式是以键值对的形式保存 key-value

# 每一对键值对之间使用 , 进行分割

# 使用一对 {} 来表示字典

# 字典里的key不允许重复，如果出现重复的kye,后出现的key会覆盖之前出现的kye

# value可以是任意类型的数据

字典里的key 也可以是任意类型的数据，但是一般使用字符串作为字典的key

使用dict 类也可以创建一个字典

# info = dict({'name': 'lisi'})

# print(info)

1. 字典的增删改查：

info['id'] = 100 # 为不存在的键赋值就是添加元素；

字典的每个元素中的数据是可以修改的，只要通过key找到，即可修改；

pop(key)

删除指定的key,并且返回对应的value；

popitem

删除一个元素，并且返回元组。元组里有两个数据，第0个是被删除的key,第一个是value；

del info['name'] # del 可以通过键删除字典里的指定元素；

clear清空整个字典；

for key in dict.keys():

for value in dict.values():

for item in dict.items():

for key,value in dict.items():

print("key=%s,value=%s" % (key, value))

1. 集合是可变类型，但是只能添加和删除，不能修改。

add() 只能添加一个元素；update 添加多个数据；pop() 随机移除一个元素；

40.

*使用get方法也可以获取一个数据  
# # 使用get方法来获取数据，如果key不存在，不会报错，会返回None  
# print(person.get('name')) # get参数需要传入要查询的key  
# print(person.get('xxxx')) # None  
# # #  
# # # # get方法可以设置一个默认值  
# # # # 如果key存在，就获取key对应的值；如果key不存在，使用默认值  
# # # print(person.get('score', 89)) # 89  
# # # print(person.get('age', 34)) # 18  
# # # # get方法的默认值，不会改变原有的字典  
# # # print(person)  
# # #  
# # # # 修改元素和增加元素的语法是一样的  
# # # # 如果key在字典里已经存在了，是修改；如果key在字典里不存在，是新增  
# # # # 修改数据  
# # # person['name'] = 'lisi'  
# # # print(person)  
# # #  
# # # # 增加数据  
# # # person['address'] = '上海'  
# # # print(person)  
# # #  
# # # # 删除数据  
# # # # 使用pop(key) 删除指定的key,并且返回对应的value  
# # # # print(person.pop('address')) # 上海  
# # # # print(person)  
# # #  
# # # # popitem 删除一个元素，并且返回元组。元组里有两个数据，第0个是被删除的key,第一个是value  
# # # # print(person.popitem()) # ('address', '上海')  
# # # # print(person)  
# # #  
# # # # del 运算符也可以删除一个指定的元素  
# # # # del person['address']  
# # # # print(person)  
# # # temp={'name':'zhangsan','sex':'man','age':23}  
# # # del temp['sex']  
# # # print(temp)  
# # # person = {'name': 'zhangsan', 'age': 18, 'gender': 'male'}  
# # # for x in person:  
# # # print('%s的值是：%s'%(x,person[x]))  
# # # for x, y in enumerate(person):  
# # # print('%s的值是%s' % (x, y))  
# # # its = person.items()  
# # # print(its)  
# # # for it in its:  
# # # print('%s的值是%s' % (it[0], it[1]))  
# # # print('-'\*20)  
# # # for k, v in person.items():  
# # # print('%s的值是%s' % (k, v))  
# # #  
# # # temp={'name':'zhangsan','sex':'man','age':'23','hobby':'ball'}  
# # # for i,j in temp.items():  
# # # print('%s的值是：%s'%(i,j))  
# # # sss={}  
# # # xxx=set()  
# # # print(type(sss))  
# # # print(type(xxx))  
# # # company={'百度','阿里','腾讯','小米','恒大','vivo','oppo'}  
# # # company.add(('华为','中兴'))  
# # # print(company)  
# # # company.update(('魅族','锤子'))  
# # # print(company)  
# # # company.pop()  
# # # print(company)  
# # # company.remove('oppo')  
# # # print(company)  
# # # s5 = '你好，我的名字是{1},我今年{age}岁了.他的名字是{0}'.format('tom', 'jerry', age=12)  
# # # print(s5)  
# # s='你好。我的名字是{},我今年已经{}岁了，我的哥哥比我大{}岁'.format('jack',23,3)  
# # print(s)  
# # print('我的名字是{1}，我今年{0}岁了，我哥哥的名字是{name}'.format(23,'mike',name='john'))  
# # ss=('jerry',23,'jack')  
# # print('我的名字是{}，我今年{}岁了，我哥哥的名字是{}'.format(\*ss))  
# # ss={'name':'jerry','age':23,'new\_name':'jack'}  
# # print('我的名字是{name}，我今年{age}岁了，我哥哥的名字是{new\_name}'.format(\*\*ss))  
# def sum(a,b,c):  
# x=a+b+c  
# y=x/3  
# return x,y  
# result=sum(1,2,3)  
# print(result)  
#  
# def get\_sum(a, b, \*args, xxx='hehehe'):  
# print('xxx====>{}'.format(xxx))  
# result = a + b  
# for x in args:  
# result += x  
# return result  
#  
#  
# # print(get\_sum(xxx='hahaha', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10))  
# # 实参在传值的时候，要按照位置参数，可变参数，关键字参数的顺序来传递  
# print(get\_sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, xxx='haha'))  
# # 实参在传值的时候，一定要按照位置参数，可变参数，关键字参数的顺序来执行  
# 定一个列表，用来存储所有的名片信息(每个名片是一个字典)*info\_list = []

1. 元组，列表，字典三者的增删改查对比：

a=('A','B','C','D','E','F')  
b=['g','h','i','j','k','l']  
c={'name':'谢正东','sex':'男','age':22}  
*print*(a+('G','H'))*# 元组的增加方法*b.append('m') *# 列表的增加方法  
print*(b)  
b.insert(5,'q') *# 列表的增加方法  
print*(b)  
b.extend(('QQ','wechat')) *# 列表的增加方法  
print*(b)  
c['hobby']='basketball'*# 字典的增加方法  
print*(c)

b.pop(2) *# 列表的删除方法，pop()括号里不加下标时，默认删除最后一个  
print*(b)  
b.remove('l')*# 列表的删除方法，指定元素删除  
print*(b)  
*del* b[1] *# 列表的删除方法，根据下标来进行删除。  
print*(b)  
*# del b # 列表的删除方法，可以直接删除整个变量。  
# print(b)*c.pop('age') *# 字典的删除方法，pop()括号里加键值  
print*(c)  
*del* c['sex'] *# 字典的删除方法  
print*(c)

b[5]='QQ' *# 列表的修改方法  
print*(b)  
c['age']=18 *# 字典的修改方法  
print*(c)  
*print*(a[2]) *# 元组的查询  
print*(b[5]) *# 列表的查询  
print*(c['name']) *# 字典的查询*

1. 列表推导式：

*# 列表推导式*a=[x *for* x *in range*(0,10) *if not* x%2]  
*print*(a)  
b=[(x,y) *for* x *in range*(5) *for* y *in range*(5)]  
*print*(b)  
c=[z *for* z *in range*(10)]  
*print*(c)  
*# 请写出一段 Python 代码实现分组一个 list 里面的元素,  
# 比如 [1,2,3,...100]变成 [[1,2,3],[4,5,6]....]*num=[x *for* x *in range*(1,101)]  
a=[num[i:i+3] *for* i *in range*(0,100,3)]  
*print*(a)

42 递归函数：

*def* fib(n):  
 *# if n == 10:  
 # return 89  
 # if n == 11:  
 # return 144  
 if* n == 0 *or* n == 1:  
 *return* 1  
 *return* fib(n - 2) + fib(n - 1) *# fib(4)+fib(5)*

1. 私有方法和私有属性：在方法和属性名之前加上两个\_,表示是私有属性和私有方法。从外部直接访问私有属性的话，可以使用：对象名.\_类名+私有属性。但是这种方法强烈不建议使用，从外部访问私有方法的话，可以使用：对象名.\_类名+私有方法名。

**class Person(object):  
 def \_\_init\_\_(self,name,age):  
 self.name=name  
 self.age=age  
 self.\_\_money=2000  
 def \_\_shopping(self,cost):  
 self.\_\_money-=cost  
 print('你的余额为：{}'.format(self.\_\_money))  
 def test(self):  
 return self.\_\_money  
 def test1(self,money):  
 self.\_\_money=money  
p1=Person('张三',20)  
print(p1.test())  
p1.test1(20000)  
print(p1.test())  
p1.\_Person\_\_shopping(1000)**

1. 继承，多继承：

# 子类可以继承父类的属性和方法

class Animal(object):

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

self.name=name

def sleep(self):

print('{}睡觉了'.format(self.name))

class Dog(Animal):

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

self.name=name

class Cat(Animal):

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

self.name=name

dog=Dog('狗')

cat=Cat('猫')

dog.sleep()

cat.sleep()

'''

'''

# 多继承,子类继承父类，父类又继承一个父类。

class Animal(object):

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

self.name=name

def sleep(self):

print('{}快睡觉了'.format(self.name))

class Dog(Animal):

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

self.name=name

def sleep(self):

print('{}已经睡觉了'.format(self.name))

def fark(self):

print('{}在叫'.format(self.name))

class XiaoQi(Dog):

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

self.name=name

a=XiaoQi('小七')

a.fark()

a.sleep()

print(XiaoQi.\_\_mro\_\_)

'''

'''

# 一个子类，继承不同的父类

class Father(object):

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

self.name=name

def sleep(self):

print('{}已经睡觉了'.format(self.name))

class Mother(object):

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

self.name=name

def sleep(self):

print('{}已经睡着了'.format(self.name))

class Son(Mother,Father):

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

self.name=name

def test(self):

self.sleep()

a=Son('张三')

a.sleep()

a.test()

print(Son.\_\_mro\_\_)

'''

'''

# 子类覆盖父类的方法，在子类中重新定义一个和父类中方法同名的方法，这样的话，子类在调用时，会先调用自己的方法

class Father(object):

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

pass

def eat(self):

print('father在吃东西')

class Son(Father):

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

pass

def eat(self):

print('son在吃东西')

a=Son()

a.eat()

# 子类扩展父类的方法

class Father(object):

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

pass

def eat(self):

print('father在吃东西')

class Son(Father):

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

pass

def eat(self):

print('son在吃东西')

super().eat()

a=Son()

a.eat()

44.多态方法的使用：

# 多态方法的使用

class Dog(object):

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

pass

def work(self,dog):

dog.work()

class Adog(Dog):

def work(self):

print('在搜毒品')

class Bdog(Dog):

def work(self):

print('在攻击假人')

class Cdog(Dog):

def work(self):

print('在玩耍')

a=Dog()

b=Adog()

c=Bdog()

d=Cdog()

a.work(b)

a.work(d)

45.导入模块的方法有以下几种：

* import 模块名
* from 模块名 import 功能名
* from 模块名 import \*
* import 模块名 as 别名
* from 模块名 import 功能名 as 别名

1. 面向过程关注点在于怎么样做，面向对象关注点在于谁来做。

面向对象的特点：封装，继承，多态。

47.类属性只能通过类对象修改，不能通过实例对象修改

尽量避免类属性和实例属性同名。如果有同名实例属性，实例对象会优先访问实例属性。

* ****第一个形参是类对象的方法****
* 需要用装饰器@classmethod来标识其为类方法，对于类方法，****第一个参数必须是类对象****，一般以cls作为第一个参数。
* 当方法中 ****需要使用类对象**** (如访问私有类属性等)时，定义类方法
* 类方法一般和类属性配合使用
* 需要通过装饰器@staticmethod来进行修饰，****静态方法既不需要传递类对象也不需要传递实例对象（形参没有self/cls）****。
* 静态方法 也能够通过 ****实例对象**** 和 ****类对象**** 去访问。
* 当方法中 ****既不需要使用实例对象****(如实例对象，实例属性)，****也不需要使用类对象**** (如类属性、类方法、创建实例等)时，定义静态方法
* ****取消不需要的参数传递****，有利于 ****减少不必要的内存占用和性能消耗****

### **注意点：**

* 类中定义了同名的方法时，调用方法会执行最后定义的方法

1. 自定义异常：

# 让用户输入内容，如果是长度不在 6~12位，直接报错  
class LengthError(Exception):  
 def \_\_init\_\_(self,min,max):  
 self.min=min  
 self.max=max  
 def \_\_str\_\_(self):  
 print(**'密码长度要在{}和{}之间'**.format(self.min,self.max))  
b=input(**'输入一个密码：'**)  
def check(a):  
 if not (12>=len(a)>=6):  
 raise LengthError(6,12)  
 else:  
 print(**'你输入的密码合法，是{}'**.format(a))  
check(b)

48：自定义一个可迭代对象：

from collections.abc import Iterable, Iterator  
  
  
# 如果一个类要成为一个迭代器，要重写 \_\_next\_\_方法  
class MyIterator(object):  
 def \_\_init\_\_(self, n):  
 self.n = n  
 self.current = 0  
  
 def \_\_next\_\_(self):  
 if self.current < self.n:  
 self.current += 1  
 return 100  
  
 # 当不满足条件的时候，raise StopIteration异常  
 raise StopIteration  
  
  
# my\_iter = MyIterator()  
# print(isinstance(my\_iter, Iterator))  
  
  
class Demo(object):  
 def \_\_init\_\_(self, n):  
 self.n = n  
  
 # 重写了 \_\_iter\_\_ 方法，它就是一个可迭代对象  
 def \_\_iter\_\_(self):  
 # 如果要让一个可迭代对象能够for...in循环  
 # \_\_iter\_\_方法，必须要返回一个迭代器  
 return MyIterator(self.n)  
  
  
d = Demo(10000)  
# print(isinstance(d, Iterable))  
  
# for ... in 就是不断的调用 迭代器的 \_\_next\_\_方法  
for m in d:  
 print(m)  
# print(d.\_\_iter\_\_().\_\_next\_\_())  
# print(d.\_\_iter\_\_().\_\_next\_\_())  
# print(d.\_\_iter\_\_().\_\_next\_\_())

49.# 打印九九乘法表  
for i in range(9):  
 i+=1  
 for j in range(i):  
 j+=1  
 print(**'%d\*%d=%-3d'**%(j,i,j\*i),end=**' '**)  
 print()

# 闰年: 能被4整除，但是不能被100整除，或者能被400整除  
# 能被400整除或者能被4整除但是不能被100整除  
year=int(input(**'请输入一个年份：'**))  
if (year%4==0 and year%100!=0) or year%400==0:  
 print(**'这一年是闰年'**)  
else:  
 print(**'这一年不是闰年'**)

color = 0xF0384E  
red=(color>>4)&0xff  
green=(color>>2)&0x00ff  
blue=color&0x0000ff  
print(red,green,blue)

# 求1~100所有整数的和  
i=0  
sum=0  
while i<101:  
 sum+=i  
 i+=1  
print(sum)  
  
result=0  
for i in range(101):  
 result+=i  
print(result)

#打印1~10的所有偶数  
for i in range(11):  
 if not i%2:  
 print(i,end=**' '**)  
i=0  
while i<11:  
 if i%2==0:  
 print(i,end=**' '**)  
 i+=1

# 打印[200,300]，第一个能被23整除的数  
# 拿到第一个能被23整除的数字以后，就直接把循环整个给停掉  
for i in range(200,301):  
 if not i%23:  
 print(i)  
 break  
i=200  
while i<301:  
 if i%23==0:  
 print(i)  
 break  
 i+=1

# 1. 不断的询问用户，"我爱你，你爱我吗?"，只有用户回答"爱"时，结束循环  
while True:  
 answer=input(**'我爱你，你爱我吗?'**)  
 if answer==**'爱'**:  
 break  
#2. 不断的提示用户输入用户名和密码，只有用户名是zhangsan,并且密码是123才停止  
while True:  
 name=input(**'请输入用户名：'**)  
 password=input(**'请输入密码：'**)  
 if name==**'zhangsan'**and password==**'123'**:  
 break

# 1. 有一个列表保存了所有的成绩。计算总成绩和平均成绩  
scores = [56, 78, 93, 65, 88, 42, 97]  
sum=0  
for i in scores:  
 sum+=i  
print(**'总成绩是%d'**%sum)  
print(**'平均成绩是%d'**%(sum/7))  
#2. 有一个列表保存了一组数字，求最大数（的下标）  
arr = [4, 100, 80, 2, 6, 104, 10, 23]  
m\_index=0  
m=arr[m\_index]  
for i ,j in enumerate(arr):  
 if j>m:  
 m=j  
 m\_index=i  
print(m\_index,m)

# 冒泡排序  
nums = [6, 5, 23, 1, 8, 7, 12, 4,9,11]  
for i in range(len(nums)-1):  
 count=0  
 for j in range(len(nums)-1-i):  
 if nums[j]>nums[j+1]:  
 count=1  
 nums[j],nums[j+1]=nums[j+1],nums[j]  
 print(nums)  
 if count==0:  
 break

# 1. 求1~100以内所有的素数(质数，除了1和它本身以外，不能再被任何数整除)  
# 2. 求水仙花数（一个三位数，如果它每个位上的数字的三次方之和等于它本身，我们称它为水仙花数）  
for i in range(2,101):  
 count=0  
 for j in range(2,i):  
 if i%j==0:  
 count=1  
 break  
 if count==0:  
 print(i,end=**' '**)  
  
  
num=int(input(**'请输入一个三位数：'**))  
a=num//100  
b=num%100//10  
c=num%10  
if pow(a,3)+pow(b,3)+pow(c,3)==num:  
 print(**'这个数是水仙花数'**)  
else:  
 print(**'这个数不是水仙花数'**)

# 一个学校，有3个办公室，现在有8位老师等待工位的分配，请编写程序，完成随机的分配  
import random  
num=[[],[],[]]  
teachers=[**'A'**,**'B'**,**'C'**,**'D'**,**'E'**,**'F'**,**'G'**,**'H'**]  
for teacher in teachers:  
 index=random.randint(0,2)  
 num[index].append(teacher)  
for i, j in enumerate(num):  
 print(**'第%d个办公室有%d个人：'**%(i,len(num[i])),end=**' '**)  
 for k in j:  
 print(k,end=**' '**)  
 print()

# 5. 写一个函数求任意个数数字之和  
def add(n):  
 sum=0  
 for i in range(1,n+1):  
 sum+=i  
 print(sum)  
add(100)

def test(a,b,\*args,xxx=**'hehehe'**):  
 print(**'xxx====>{}'**.format(xxx))  
 print(a+b)  
 print(args)  
 for i in args:  
 print(i,end=**' '**)  
test(1,2,1,2,3,4,5,6,7,8,9,10,xxx=**'xiaxiaxia'**)  
#实参在传值时，要按照位置参数，可变参数，关键字参数的顺序来传递

查找一个元素是否在一个列表中，在的话，返回下标：

def print\_index(value,values):  
 if value not in values:  
 print(**'你要查找的数据不存在'**)  
 return  
 for i,j in enumerate(values):  
 if value==j:  
 print(i)  
 break  
print\_index(**'hi'**,[**'hello'**,**'hi'**,**'good'**])

# 1. 定义一个函数，实现冒泡排序  
def bubble\_sort(num):  
 for i in range(len(num)-1):  
 count=0  
 for j in range(len(num)-1-i):  
 if num[j]>num[j+1]:  
 count=1  
 num[j],num[j+1]=num[j+1],num[j]  
 print(num)  
 if count==0:  
 break  
scores=[90,78,65,87,56,87,66]  
bubble\_sort(scores)  
  
# 2. 定义一个函数，求斐波那契数列的第n个数 1,1,2,3,5,8,13,21,34,55,89,144 ... ...  
def fib(n):  
 if n==0 or n==1:  
 return 1  
 return fib(n-1)+fib(n-2)  
  
print(fib(11))

# 使用递归函数，求1~n的数字之和  
def b(n):  
 if n==1:  
 return 1  
 return n+b(n-1)  
  
print(b(100))

# 匿名函数  
test=lambda x,y:x+y  
print(test(2, 3))  
num=lambda x,y:x\*y  
print(num(8, 9))  
print(list(filter(lambda x: x % 2 == 0, [1, 2, 3, 4, 5, 6])))

# 列表推导式的使用  
a=[x for x in range(10) if x%2]  
print(a)  
b=[(x,y) for x in range(5) for y in range(5)]  
print(b)  
# 请写出一段 Python 代码实现分组一个 list 里面的元素,  
# 比如 [1,2,3,...100]变成 [[1,2,3],[4,5,6]....]  
nums=[x for x in range(1,101)]  
c=[nums[i:i+3] for i in range(0,100,3)]  
print(c)

def outer(a):  
 a=10  
 print(**'我是outer函数'**)  
 print(**'我是outer函数里面的a:%d'**%a)  
  
 def inner():  
 nonlocal a  
 a+=2  
 print(**'我是outer函数里面的inner函数'**)  
 print(**'我是inner函数里面的a:%d'**%a)  
 return inner  
outer(14)()  
  
import time  
def cal\_time(fn):  
 start=time.time()  
 fn()  
 end=time.time()  
 print(end-start)  
def test():  
 for i in range(2000):  
 print(i)  
cal\_time(test)

# 简单装饰器的使用

import time

def cal\_time(fn):

def do\_action():

start=time.time()

fn()

end=time.time()

print(end-start)

return do\_action

@cal\_time

def test():

for i in range(10000):

print(i)

test()

a=11  
def outer(clock):  
 def can\_play(fn):  
 def do\_action(name):  
 if clock<12:  
 fn(name)  
 else:  
 print(**'太晚了，不能打了'**)  
 return do\_action  
 return can\_play  
@outer(a)  
def test(name):  
 print(**'%s还在打游戏'**%name)  
test(**'张三'**)

使用装饰器实现权限验证：

user\_permission=7  
read\_permission=4  
write\_permission=2  
execute\_permission=1  
def check\_permission(permission):  
 def handle\_action(fn):  
 def do\_action():  
 if user\_permission & permission !=0:  
 fn()  
 else:  
 print(**'对不起，你没有此权限'**)  
 return do\_action  
 return handle\_action  
@check\_permission(read\_permission)  
def read():  
 print(**'正在读取内容'**)  
@check\_permission(write\_permission)  
def write():  
 print(**'正在写内容'**)  
@check\_permission(execute\_permission)  
def execute():  
 print(**'正在执行命令'**)  
read()  
write()  
execute()

# time模块相关知识  
import time  
import datetime  
import calendar  
print(time.time())  
print(time.strftime(**'%Y-%m-%d %H:%M:%S'**))  
print(**'hello'**)  
time.sleep(5)  
print(**'world'**)  
print(datetime.datetime.now())  
print(datetime.datetime.now().replace(year=2020,month=11))  
print(datetime.datetime.now() + datetime.timedelta(days=20))

import time  
print(time.strftime(**'%Y-%m-%d %H:%M:%S'**))  
temp=[{**'name'**:**'谢正东'**,**'age'**:22,**'tel'**:18055477325},  
 {**'name'**:**'未方成'**,**'age'**:27,**'tel'**:18256474202},  
 {**'name'**:**'张长庚'**,**'age'**:22,**'tel'**:13773422907}]  
def print\_menu():  
 print(**'='**\*30)  
 print(**' 名片管理系统 '**)  
 print(**'='**\*30)  
 print(**'1.添加名片'**)  
 print(**'2.修改名片'**)  
 print(**'3.删除名片'**)  
 print(**'4.查询名片'**)  
 print(**'5.显示所有名片'**)  
 print(**'6.退出系统'**)  
def add():  
 new\_name=input(**'请输入姓名：'**)  
 for str in temp:  
 if str[**'name'**]==new\_name:  
 print(**'对不起，该用户名已经被注册，请重新输入'**)  
 return  
 new\_age=input(**'请输入年龄：'**)  
 new\_tel=input(**'请输入电话号码：'**)  
 num={}  
 num[**'name'**]=new\_name  
 num[**'age'**]=new\_age  
 num[**'tel'**]=new\_tel  
 temp.append(num)  
 return  
def delete():  
 number=int(input(**'请输入你要删除名片的编号：'**))  
 if not (len(temp)>=number>=0):  
 print(**'对不起，你输入的编号有误，请确认后再次输入'**)  
 return  
 del temp[number]  
 return  
def change():  
 number=int(input(**'请输入要修改的名片编号：'**))  
 if not (len(temp)>=number>=0):  
 print(**'对不起，你输入的编号有误，请确认后再次输入'**)  
 return  
 print(**'name:{name} age:{age} tel:{tel}'**.format(\*\*temp[number]))  
 name=input(**'请输入新姓名：'**)  
 age=int(input(**'请输入新年龄:'**))  
 tel=input(**'请输入新电话号码：'**)  
 temp[number][**'name'**]=name  
 temp[number][**'age'**]=age  
 temp[number][**'tel'**]=tel  
 return  
def find\_menu():  
 number=int(input(**'请输入你要查询名片的编号:'**))  
 if not (len(temp)>=number>=0):  
 print(**'对不起，你输入的编号有误，请确认后再次输入'**)  
 return  
 print(**'name:{name} age:{age} tel:{tel}'**.format(\*\*temp[number]))  
 return  
def all\_menu():  
 for i in temp:  
 print(**'name:{name} age:{age} tel:{tel}'**.format(\*\*i))  
def exit():  
 answer=input((**'亲，你真的要退出吗？yes or no'**))  
 if answer.lower()==**'yes'**:  
 return True  
 return False  
def main():  
 while True:  
 print(**'='** \* 30)  
 print(**' 名片管理系统 '**)  
 print(**'='** \* 30)  
 print(**'1.添加名片'**)  
 print(**'2.修改名片'**)  
 print(**'3.删除名片'**)  
 print(**'4.查询名片'**)  
 print(**'5.显示所有名片'**)  
 print(**'6.退出系统'**)  
 number=int(input(**'请输入你要操作的序号:'**))  
 if number==1:  
 add()  
 elif number==2:  
 change()  
 elif number==3:  
 delete()  
 elif number==4:  
 find\_menu()  
 elif number==5:  
 all\_menu()  
 elif number==6:  
 is\_sure=exit()  
 if is\_sure:  
 break  
 else:  
 print(**'你输入的不合法，请重新输入'**)  
main()

# 小明体重 75.0 公斤  
# 小明每次跑步会减肥 0.5 公斤  
# 小明每次吃东西体重增加 1 公斤  
# 在打印小明这个对象时，输出小明的姓名，体重信息  
class Person(object):  
 def \_\_init\_\_(self,name,weight):  
 self.name=name  
 self.weight=weight  
 def run(self):  
 self.weight-=0.5  
 print(**'{}跑完步'**.format(self.name))  
 def eat(self):  
 self.weight+=1  
 print(**'{}吃完饭'**.format(self.name))  
 def \_\_str\_\_(self):  
 return(**'{}的体重为：{}'**.format(self.name,self.weight))  
p=Person(**'小明'**,75)  
p.eat()  
print(p)  
p.run()  
print(p)

# 房子(House) 有 户型、总面积 、剩余面积 和 家具名称列表 属性  
# 新房子没有任何的家具  
# 将 家具的名称 追加到 家具名称列表 中  
# 判断 家具的面积 是否 超过剩余面积，如果超过，提示不能添加这件家具  
# 家具(HouseItem) 有 名字 和 占地面积属性，其中  
# 席梦思(bed) 占地 4 平米  
# 衣柜(chest) 占地 2 平米  
# 餐桌(table) 占地 1.5 平米  
# 将以上三件 家具 添加 到 房子 中  
# 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(object):  
 def \_\_init\_\_(self,huxing,allarea):  
 self.huxing=huxing  
 self.allarea=allarea  
 self.namelist=[]  
 self.leftarea=allarea\*0.7  
 def add(self,name):  
 if self.leftarea<name.area:  
 print(**'房间剩余空间不足，无法放下{}'**.format(name.name))  
 return  
 self.namelist.append(name.name)  
 self.leftarea-=name.area  
 def \_\_str\_\_(self):  
 return **'户型是：{} 总面积是：{} 剩余面积是：{} 家具列表是：{}'**.format(self.huxing,self.allarea,self.leftarea,self.namelist)  
  
class HouseItem(object):  
 def \_\_init\_\_(self,name,area):  
 self.name=name  
 self.area=area  
h=House(**'次卧'**,20)  
bed=HouseItem(**'席梦思'**,4)  
h.add(bed)  
chest=HouseItem(**'衣柜'**,2)  
h.add(chest)  
table=HouseItem(**'餐桌'**,1.5)  
h.add(table)  
computer=HouseItem(**'电脑'**,10)  
h.add(computer)  
print(h)