大类	项目	语法	举例	特殊要求/备注
基础	注释	# 注释内容	# 这是一条注释	
	打印	print()		
	变量	param = ''	message = 'good job'	1.变量名只能包含字母、数字和下划线；可以字母或下划线打头，不能以数字打头； 2.变量名不能包含空格； 3.不要将python关键字和函数名用做变量名； 4.变量名应该既简短又具有描述性； 5.慎用小写l和大写O，因为可能被看成1和0。
	代码规范：缩进	四个空格		遵循PEP8规范
	代码规范：行长	不超过80个字符		
	代码规范：空行	不同部分用空行隔开		
	表示空	none	return none	
字符串	是什么样	单引号或双引号	'good job'	
	首字母大写	str.title()	name = 'steven curry' print(name.title()) 输出：Steven Curry	
	全部大写	str.upper()	name = 'steven curry' print(name.upper()) 输出：STEVEN CURRY	
	全部小写	str.lower()	name = 'Steven Curry' print(name.lower()) 输出：steven curry	
	拼接字符串	str1 + str2	str1 = 'Steven' str2 = 'Curry' print(str1 + ' ' + str2) 输出：Steven Curry	

	删除空白	str.rstrip()	name = 'Steven Curry' print(name.rstrip()) 输出：StevenCurry	
数字	加减乘除	同数学		
	乘方	**		
	浮点数	带小数点的		
	数字转字符串	str()	str = str(5) print(str) 输出：'5'	字符串和字符串相加会变成一个字符串，数字和字符串相加会报错，需要先把数字转成字符串
	怎么表示	[]		
	访问元素	arr[index]		索引从0开始
	修改元素	arr[index] = ''	arr = ['James', 'Wade', 'Bosh'] arr[0] = 'Rose' print(arr) 输出：['Rose', 'Wade', 'Bosh']	
	在末尾添加元素	arr.append(param)		
	指定位置添加元素	arr.insert(index, param)		
	删除指定位置元素	del arr[index]		
	在末尾删除元素	arr.pop(param)		
	删除指定位置的元素并获取被删除的元素	arr.pop(index)		
	根据值删除元素	arr.remove(param)		
	永久修改列表排序：按大小顺序正序	arr.sort()		
	永久修改列表排序：按大小顺序倒序	arr.sort(reverse=True)		
	临时修改列表排序：按大小顺序正序	sorted(arr)		
	临时修改列表排序：按大小顺序倒序	sorted(arr, reverse=True)		
	列表反过来排序（不是按大小排序）	arr.reverse()		
	列表长度	len(arr)		

列表	遍历	for… in …	names = ['James', 'Wade', 'Bosh'] for name in names: print(name)	要求：for循环要缩进；不要遗漏冒号 规范：用单数和复数区分元素和列表
	数字列表：生成一系列数字	range(number1, number2, step)	range(1, 5)	number1要小于number2; 生成的数字包含number1，不包含number2
	数字列表：创建数字列表	list(range(number1, number2)	numbers = list(range(1, 5)) print(numbers) 输出：[1, 2, 3, 4]	
	数字列表：最小值	min(arr)		
	数字列表：最大值	max(arr)		
	数字列表：求和	sum(arr)		
	列表切片	arr[index1:index2]	arr = ['James', 'Wade', 'Bosh'] arr1 = arr[0:2] print(arr1) 输出：['James', 'Wade']	index从0开始; 切片获取的列表包含index1，不包含index2; 如果没有指定index1，则从列表开头开始 如果没有指定index2，则到列表末尾结束 如果index1和index2都没指定; 如果index指定为复数，则表示从末尾计数
	复制列表	arr[:]		
	元组：不可变的列表	(param1, param2, ...)	使用方法同普通列表	
	检查是否相等	==		
	检查是否不相等	!=		
	检查多个条件	and	age1 == 5 and age2 == 7	
	检查多个条件	or		
	检查特定值是否包含在列表中	param in params	arr = ['James', 'Wade', 'Bosh'] >>> 'James' in arr True	
	检查特定值不包含在列表中	if param not in params:	arr = ['James', 'Wade', 'Bosh'] >>> if 'Rose' not in arr: True	不要遗漏了冒号
	if	if 表达式:	if age <= 20 print('年纪小，头发多')	不要遗漏了冒号

if语句	if-else	if 表达式:
		  do something
		else:
		  do something else
				不要遗漏了冒号

	if-elif-else	if 表达式:
		  do something1
		elif 表达式:
		  do something2
		else:
		  do something else
				不要遗漏了冒号

	确定列表是不是空的	if 列表名	arr = [] if arr: do something	
字典	字典	{}		
	使用字典中的值	obj[key]	player = {'team': 'HEAT', 'name': 'Wade'} player['team']	
	添加键值对	obj[key] = param（key为未有的）		
	修改字典中的值	obj[key] = param（key为已有的）		
	删除键值对	del obj[key]		

	遍历所有键值对	for key, value in obj.items():	obj = {
			  'name': 'Wade',
			  'team': 'HEAT',
			  'age': 35,
			}
			for key, value in obj.items():
			  do something
				不要遗漏了冒号

	遍历所有的键，并返回一个列表	for key in obj.keys():		
	按特定顺序遍历所有的键，并返回一个列表	for key in sorted(obj.keys()):		
	遍历所有的值，并返回一个列表	for value in obj.values():		
	遍历所有的值，并返回一个去重列表	for value in set(obj.values());		
用户输入	输入	input(message)		
	字符串转数字	int(string number)		
	求模运算	%	>>> 4 % 3 1	

while	while循环基础	while 表达式 　　do something		

	break退出	while 表达式:
		　　do something
		　　if 表达式:
		　　　break		

	continue返回循环开头	while 表达式:
		　　do something
		　　if 表达式:
		　　　continue
		　　print('执行continue就不执行这句')		

函数	定义函数	def fun_name(): 　　do something 调用：fun_name()		
	返回值	def fun_name(): 　　return something		
	函数要调用列表又不想让列表被修改	fun_name(list_name[:])		
	传递任意数量的实参	fun_name(*list_name)		形参带星号，会创建一个名为list_name的空元组，函数收到的所有实参都会封装到这个元组中
	导入整个模块	import module_name 调用：module_name.fun_name()		
	导入模块中的特定函数	from module_name import fun_name1, fun_name2,…		
	使用as给模块指定别名	import module_name as other_name		
	导入模块中所有的函数（不建议使用）	form module_name import *		

	创建类	class	class Dog():
			"""定义一个狗的类"""
			
			def __init__(self, name, age):
			self.name = name
			self.age = age
			self.color = 'blue'
			
			def sit(self):
			do something
			
			def update_color(self, color):
			self.color = 'yellow'
			调用:
			my_dog = Dog('xiaotianquan', 6)
			print(my_dog.name)
			输出：xiaotianquan
				1.类名首字母大写;
				2.__init__，前后个两个下划线，类创建后自动执行这个函数
				3.__init__中的self必不可少，self是自动传递，不需要手动写
				4.以self为前缀的变量都可供类中所有方法使用

	访问类的属性	class_name.attr	my_dog.name	
	调用类中的方法	class_name.fun_name()	my_dog.sit()	
	修改类的属性的值：直接修改	class_name.attr = newValue	my_dog.color = 'red'	
	修改类的属性的值：通过方法修改	类中写一个方法，方法中修改了属性，外部调用这个方法	my_dog.update_color('white')	

类	继承	class	class Dog():
			"""定义一个狗的类"""
			
			def __init__(self, name, age):
			self.name = name
			self.age = age
			self.color = 'blue'
			
			def sit(self):
			do something
			
			class MaleDog(Dog):
			"""定义一个公狗的类"""
			
			def __init__(self, name, age):
			super().__init__(name, age)
			self.color = 'red'
			
			调用:
			my_male_dog = MaleDog('Jack', 3)
			my_male_dog.sit()
				1.父类必须在当前文件前
				2.定义子类时，必须在括号内指定父类名称
				3.super()可以把父类和子类关联起来
				4.子类定义属性要写在super()下面
				5.子类可以定义自己的方法
				6.子类定义和父类同名的方法，则使用子类里的方法

	导入单个类	from filename import Class_name		filename不含后缀
	导入多个类	from filename import Class_name1, Class_name2, …		filename不含后缀
	导入整个模块	import filename		filename不含后缀
	导入模块中的所有类	form filename import *		filename不含后缀
	类编码风格			1.类名才用驼峰命名法 2.实例名和模块名才用小写格式 3.每个类，在类定义后面都应该包含一个文档字符串 4.类中用一个空行分隔方法，模块中用两个空行 5.导入模块时，外部模块先写，自己写的模块放后面
	从文件中读取数据	with open(filename) as file_object: contents = file_object.read() print(contents)		1.用with，使用完文件后会自动关闭 2.finename带后缀

文件操作	文件路径	Linux和OSX: with open('text_files/filename.txt') as file_object Windows: with open(r'text_files\filename.txt') as file_object		windows中用反斜杠，且建议在单引号前面加r
	逐行读取	with open(filename) as file_object: for line in file_object: print(line)		
	创建一个包含文件各行内容的列表	with open(filename) as file_object: lines = file_object.redlines()		
	写入空文件	with open(filename, 'w') as file_object: file_object.write(message)		写入文件的内容只能为字符串
	写入多行	多个file_object.write(message)，每个message末尾加'\n'		
	给文件添加内容	with open(filename, 'a') as file_object: file_object.write(message)		
	存储数据	json.dump()	import json numbers = [1, 2, 5, 7, 11] filename = 'numbers.json' with open(filename, 'w') as f_obj: json.dump(numbers, f_obj)	1.先导入json模块 2.指定要存储到的文件的名称，通常用.json文件扩展名 3.使用dump()存储到文件中
	读取文件	json.load()	import json filename = 'numbers.json' with open(filename, 'w') as f_obj: json.load(f_obj)	

异常处理	用try-except	try:
		do something
		except 异常对象名称:
		do something提示/pass
		else:
		do something
				1.try里面写可能异常的代码
				2.except写异常的时候给用户看的东西，如果不想让用户知道，则写个pass，表示啥也不做
				3.依赖于try中代码执行成功后执行的代码