**Python学习**

**2022.3.17**

1. 字符串(String)

转义符 \。

反斜杠可以用来转义，使用 r 可以让反斜杠不发生转义。 如 r"this is a line with \n" 则 \n 会显示，并不是换行。

**2022.3.19**

1. Random

random.getstate得到生成随机数此时的系统种子，random.setstate重置种子

1. Decimal

定义一个十进制的数

1. and、or预算符遵循短路逻辑
2. Format

两个部分组成，字符串模板和模板数据内容组成，通过大括号{}，就可以把模板数据内容嵌到字符串模板对应的位置。

**2022.3.22**

1. List的增、删、改、查方法

列表末尾：append（单元素）extend（可迭代对象）指定位置：insert( ，)

remove（删除指定元素）pop（删除指定位置的元素）clear（删空）

sort（）排序 reverse（）翻转列表

count（）计数index（查找指定元素的位置）

浅拷贝和深拷贝——浅：copy（）和切片引用

深：copy模块，调用copy模块的deepcopy函数

1. ord函数将字符变成指定编码
2. 元组不可变的列表
3. 序列（字符串、列表，元组）**解包 x,y,z = tuple(1,2,3)**

**2022.3.23**

1. 字符串的方法

改变大小写的方法、调整字符串位置的方法、

查找方法：count、find、rfind、index

替换方法：expandtabs（替换tab为空格）、replace、translate（指定规则的替换）

判断方法：startswith、endswith（判断子字符串是否在主字符串的开头和结尾）

判断大小写、判断是否英文字母、判断是否数字、判断是否python保留关键字

截取方法：lstrip、rstrip、strip去除字符（或空白）、removeprefix、removesuffix去除指定字符

拆分和拼接方法：partition、rpartition以指定字符切分主字符串为三元组

split、rsplit切分字符串为列表splitlines以换行符作为拆分标志的拆分

join（可迭代对象）以分隔符为标志的拼接 效率远远大于+

格式化字符串的方法;format超多语法方法

1. 序列方法

id函数返回指向对象的唯一标识的整数值

del函数删除指定对象或对象指定位置的元素

sorted函数针对所有序列对象的排序、reversed函数返回一个反向可迭代对象

enumerate函数返回一个二元列表的可迭代对象

zip聚合两个可迭代对象的迭代器

map生成指定函数的一个或多个可迭代对象的值得迭代器

filter过滤器类似map只返回为正的

iter函数将可迭代对象转化为迭代器

**2022.3.25**

1. 操作字典的方法

增：fromkeys为不同的键创建一个值相同的字典

删：pop、popitem（删除字典的最后一个键值对）、del函数

改：update

查：get、setdefault（查找对应的键，如果不在字典中，增加新的键值对）

items、keys、values返回字典的键值对、键、值的视图对象

1. 操作集合的方法（集合中的元素是独一无二的、且是无序的）

集合用于检查列表中的元素是否唯一

isdisjoint（检查是否有交集）、issubset（检查是否是子集）、

issuperset（检查是否是超集）、union（形成并集）、intersection（找交集）

difference（找差集）、symmetric\_difference（找对称差集：排除交集以后的所有元素）以上方法均支持运算符操作

1. 仅适用于set的方法（不适用于frozenset）

update、intersection\_update、difference\_update、symmetric\_difference\_update、

add、remove、discard（抛出异常静默处理）

hash求哈希值、可哈希（不可变的对象）才能作为字典的键和集合的值

1. 函数的参数

形参、实参、位置参数、关键字参数

收集参数

myfunc(\*args):\*args在形参实现的元组打包//\*args在实参实现的元组解包

myfunc（\*\*kwargs）：\*\*kwargs实现的字典打包

1. 作用域

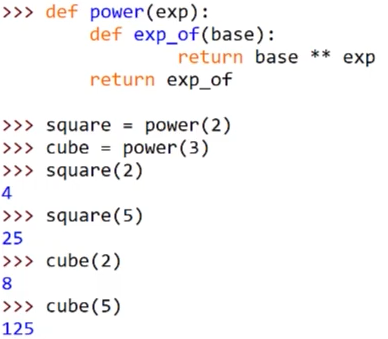
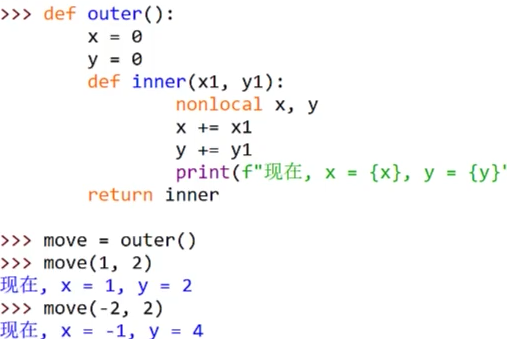
全局变量、局部变量 global语句

nonlocal语句实现覆盖外部函数值的作用

**2022.3.26**

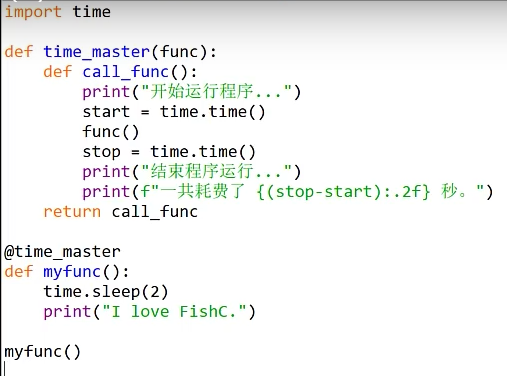
1. 函数的要点

闭包（工厂函数）用于记忆外层函数局部变量的特性，内层函数为参数返回

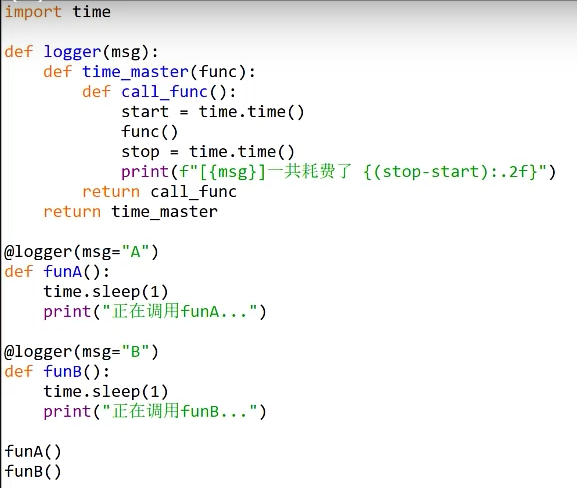


装饰器（@函数）——用于插入日志、添加性能测试、先进行权限验证

闭包和函数作为参数传递给另一个函数的联合实现



给装饰器传递参数



lambda表达式——lambda arg1，arg2,arg3,···argN:expression，

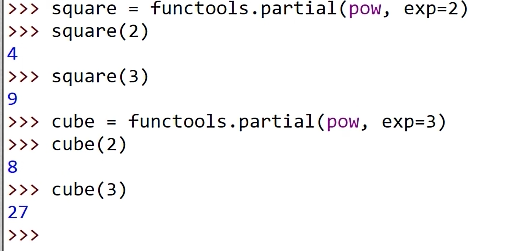
作用：出现在常规函数不能出现的地方

生成器（实现方法：将return替换成yield、生成器表达式）：每次产生一个数或者变量

函数文档、类型注释（Python检测Mypy）、内省（\_\_name\_\_、\_\_annotations\_\_）

高阶函数（funtools）——函数作为参数值传递给了另一个函数：

偏函数（partial）“创建一个调用另外一个部分参数或变量已经预置的函数”的函数的用法



@wraps装饰器



1. isinstance()函数用于检验数据类型

例子：isinstance(100, int) # ==> True

**2022.4.7**

1. Python的类和对象

区分类方法（@classmethod和第一个参数不是self而是cls）和实例方法、私有属性和实例属性

\_\_init\_\_方法重写可以传入参数

self引用的方法和属性相当于实例方法和实例属性

继承时为了防止子类覆盖父类的方法的两种措施：\_调用未绑定的父类方法、\_使用super函数

Python类的组合可以实现多了类的方法调用

1. 类和对象相关的BIF（内置函数）

issubclass（）检查是否是子类、isinstance（）检查是否是类的对象

检查对象中是否包含某个属性：hasattr、getattar、delattr

setattr （创建不存在的属性并赋值）

property通过属性调用方法：分别是get、set、del

**2022.4.7**

1. 理解Python的多态：同一个方法在不同子类中的表现形式的多样性
2. 操作类的方法：type（查看属性）dir（获取变量的所有属性）

例题：希望除了name和gender外，可以提供任意额外的关键字参数，并绑定到实例，请修改 Person 的\_\_init\_\_()定义，完成该功能。

提示：结合在入门课程学习的向函数传递任意关键字参数的方法以及刚刚学习的setattr()方法。

参考答案：

class Person(object):

def \_\_init\_\_(self, name, gender, \*\*kw):

self.name = name

self.gender = gender

for k, v in kw.items():

setattr(self, k, v)

p = Person('Bob', 'Male', age=18, course='Python')

print(p.age)

print(p.course)

1. Python里的特殊方法（魔法方法）——有些可以有返回值，有些不可以

\_\_new\_\_（一般不重写，调用不可变的类时需要重写）开头第一个变量是cls

\_\_init\_\_、\_\_del\_\_（所有的引用全部结束才会调用）：构造方法和析构方法

Python里为每一个类内置了算法运算（加减乘除取余等等）、反运算（a,b位置互换）、增量运算（+=）、一元操作符的魔法方法

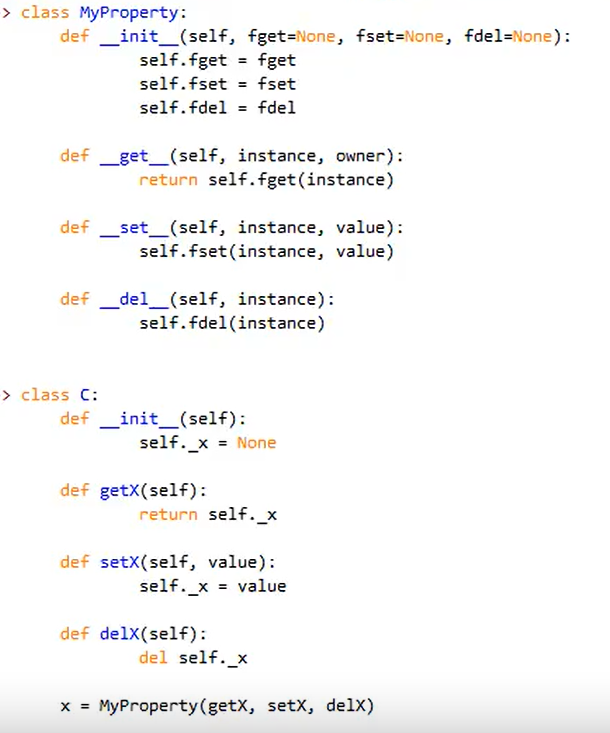
\_\_src\_\_和\_\_repr：只能返回字符串类型

属性访问的魔法方法：

\_\_getattr\_\_(用户访问一个不存在的属性时)、\_\_getattribute\_\_（用户访问一属性时）,\_\_setattr\_\_（赋值时调用，name、value两个属性）、\_\_delattr\_\_

描述符（property的原理）：将特殊类型的类的实例指派给另一个类的属性

\_\_get\_\_(属性被访问时，返回属性的值)、\_\_set\_\_、\_\_delete\_\_



定制容器的协议：

容器是不可变的——\_\_len\_\_、\_\_getitem\_\_

容器是可变的——\_\_len\_\_、\_\_getitem\_\_、\_\_setitem\_\_、\_\_delitem

迭代器： iter（）、next（）迭代器的魔法方法： \_\_iter\_\_（）、\_\_next\_\_（）

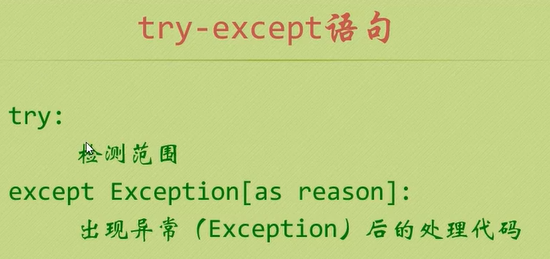
\_\_slots\_\_:限制类的属性的个数

\_\_call\_\_()把一个类实例也变成一个可调用对象

**2022.4.8**

1. **self**指向对象的实例属性
2. 函数之间可以赋值，指向相同的内容
3. 关于异常处理：

检测异常



可以一个try和多个except叠加使用



引出异常：raise

**2022.4.10**

1. 模块——程序的封装

导入模块的方法

import 模块名

from 模块名 import 函数名（\*所有的函数）

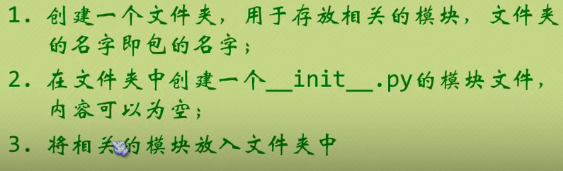
import 模块名 as 新名字

if \_\_name\_\_ == ‘\_\_main\_\_’ 放在模块中，用于测试代码，保证主程序执行时不执行模块中的测试代码

搜索路径：

添加路径 导入模块import sys sys.path.append(新的路径)

包（package）



1. 函数的文档在函数的最顶部以注释的形式展示

**2022.4.11**

1. Python读写文件

打开文件 open（）函数

写入文件 write（）、writelines（）

关闭文件 close()

查看文件指正 tell（）、设置文件指针的位置seek（）

文件的截取（删除截取位置后的内容）truncate（）

1. 路径处理

pathlib模块的Path

cwd（获取当前目录的路径）、is\_dir(判断是否是文件夹)、is\_file(判断是否是文件)、exists（判断是否存在）、name属性（获取路径的最后一部分）、stem(获取文件名) 、suffix（获取文件后缀）、parent（获取父级目录）、parents（获取父级祖先构成的不可变序列）、parts（路径的各个部分以元组的形式存放）、stat（查询文件夹或文件的信息）

相对路径和绝对路径：

相对路径—— 以当前目录为基准的路径推到.表示当前路径..表示上一级路径

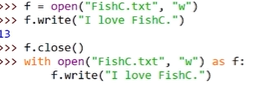
resolve 相对路径转化为绝对路径

mkdir（parents参数生成多个文件夹，exist\_ok允许存在文件夹 ）创建文件

rmdir（删除文件夹）、unlink（删除文件）

glob（查找功能）glob（\*/\*.py——查找当前目录下一级目录中所有的py文件）、glob（\*\*/\*.py——迭代查找当前目录下一级目录中所有的py文件）

with语句和上下文管理器



Pickle 将源文件转化为二进制代码：dump（）、load（）

**2022.4.15**

1. Python与网路编程

* socket（套接字）——实现服务端和客户端之间的网络通信

服务端建立四个步骤：新建socket、绑定IP和端口(bind)、监听连接(listen)、接受连接(accept)。

客户端两个步骤：新建socket、连接服务端(connect)。

当网络连接上以后，客户端和服务端就可以进行数据通信，通过send()函数发送数据，通过recv()函数接收数据。

服务端，新建server.py的文件：

import socket

server = socket.socket() # 1. 新建socket

server.bind(('127.0.0.1', 8999)) # 2. 绑定IP和端口（其中127.0.0.1为本机回环IP）

server.listen(5) # 3. 监听连接

s, addr = server.accept() # 4. 接受连接

print('connect addr：{}'.format(addr))

content =s.recv(1024)

print(str(content, encoding='utf-8')) # 接受来自客户端的消息，并编码打印出来

s.close()

客户端，新建client.py文件：

import socket

client = socket.socket() # 1. 新建socket

client.connect(('127.0.0.1', 8999)) # 2. 连接服务端（注意，IP和端口要和服务端一致）

client.send(bytes('Hello World. Hello Socket', encoding='utf-8')) # 发送内容，注意发送的是字节字符串。

client.close()

在一个终端运行服务端：python server.py

在另外一个终端运行客户端：python client.py

在服务端的终端，将会输出以下信息：

connect addr：('127.0.0.1', 50382)

b'Hello World. Hello Socket'

* Python自带的HTTP服务器

python提供了简易的HTTP服务器，可以直接运行

在终端，输入python -m http.server就启动了HTTP服务器。

（python -m http.server 9999（设置端口） -d /user（指定文件夹） ）

启动成功后，会输出以下信息：

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

提示HTTP服务器在本机8000端口运行，接着就可以在浏览器输入http://127.0.0.1:8000看到由这个服务器提供的网页。

* Python发送HTTP请求

使用Python自带的功能启动一个HTTP服务器，并通过浏览器浏览到了这个HTTP服务器提供的页面。在浏览的过程中，实际上是浏览器向HTTP服务器发送了一个HTTP请求。

除了使用浏览器发送HTTP请求，通过代码也可以向HTTP服务器发送请求，Python提供了相关的库urllib，通过urllib包里面的request，可以向其他HTTP服务器发起请求。

from urllib import request

response = request.urlopen('https://www.imooc.com') # 向慕课网官网发出请求

print(response) # ==> <http.client.HTTPResponse object at 0x000001377D631948>

请求成功的话，会得到一个HTTPResponse，它是来自HTTP服务器的一个回应，可以把这个回应的一些信息打印出来。

状态码：

print(response.status) # ==> 200

状态码是一个三位整数，在HTTP协议的标准里面，定义了很多状态码，其中200表示请求是正常的。

响应头：

​for k, v in response.getheaders():

print('{}: {}'.format(k, v))

将会输出以下信息，这是HTTPResponse附带的一些信息，包括服务端的服务器是什么、请求时间、内容类型、内容长度等等。

Server: openresty

Date: Thu, 20 Aug 2020 08:16:07 GMT

Content-Type: text/html; charset=UTF-8

Content-Length: 376639

Connection: close

Vary: Accept-Encoding

Vary: Accept-Encoding

X-Varnish: 280516221 281284036

Age: 29

Via: 1.1 varnish (Varnish/6.0)

X-Cache: HIT from CS42

Accept-Ranges: bytes

* Python的requests库

Python官方提供的urllib库可以满足一般情况下的HTTP操作，但是urllib这个库是用来处理url地址的，并不是专门处理HTTP操作的包。因此在很多场景下一般会使用requests库来进行HTTP请求。

requests库是著名的Python第三方库，使用requests库，可以定制化你的HTTP请求，包括请求方法，请求参数等等。

由于requests是第三方库，因此在使用前，需要安装。

pip install requests

安装完成后，使用requests库来请求慕课网官网。

response = requests.get('https://www.imooc.com')

# 打印状态码

print(response.status\_code)

# 打印回应头

print(response.headers)

在一般的使用上，requests和urllib没有太大区别，但是在复杂的场景中，requests可以提供urllib无法提供的强大功能。因此，在使用上，建议使用requests库代替urllib库来进行HTTP请求等的操作。

* HTTP响应的内容

通过urllib或者requests请求后，会得到一个HTTPResponse，HTTPResponse拥有状态码、回应头等的信息。

对于页面，通常是由文字、图片等信息组成的，这些属于HTTPResponse的内容。

import requests

response = requests.get('https://www.imooc.com')

content = str(response.content, encoding='utf-8') # ==> 打印具体内容

打印的结果是一个很长的字符串，显得杂乱无章，但其实它是由结构的，它是一个标准的HTML页面，可以从页面内容里面获取很多有用的数据。

网络爬虫是典型的应用程序，它的工作原理就是通过不断的请求互联网的页面，并从回应中解析获取出有用的数据；数据积累后，可以有很多用处。

通过requests获得网页的内容后，我们可以尝试使用一些简单的方法获取页面的内容。

content\_list = content.split('\n') # 分行

len(content\_list) # 打印页面内容的行数

在网页中，页面内部链接其他资源的信息是通过href提供的，通过字符串匹配的方式可以过滤出包含链接的行。

for line in content\_list:

if 'href' in line:

print(line.strip())

1. python函数编程

map函数：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，map()函数会返回一个迭代器，可以依次迭代得到原来list的元素被函数f处理后的结果。

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python2中，reduce()函数和map()函数一样，可以直接使用，但是在python3中，reduce()函数被收录到functools包内，需要引入functools才可以使用。

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，并返回一个迭代器，可以迭代出所有符合条件的元素。

sorted（）函数：sorted接受key参数，用来指定排序的字段，key的值是一个函数，接受待排序列表的元素作为参数，并返回对应需要排序的字段。如果需要倒序，指定reverse参数即可。