# $$$

学会用精简的demo代码去解释面试题、基础word概念等

# 一 &二、 初始Python 和语言基础

Microsoft Visual Studio安装PTVS插件

Eclipse 安装PyDev插件

Python3.x中提供了中文注释，为了兼容旧的： #coding=utf-8

PEP8 = Python Enhancement proposal =增强建议书 =编码规范

顶级定义= 函数 或者类的定义

标识符 = **标识** 变量、函数、类、模块和其他对象的**名字**

（汉字也能当标识符，变量我的名字=’明日科技’，print(我的名字) ）

内置函数type()，id()变量所指的内存地址

**基本数据类型：**

数字Number：

整数int、浮点float、复数 complex

字符串string

转义字符=反斜杠 = \

\n=line=换行， \r=return=回车

布尔：

True 、False

储存多个元素的对象称为容器(container)

**数据类型转换：**  
 Python不用像C/Java在使用变量前先声明变量的类型

常用的类型转换函数P41

**input() 函数：**

tip=input(“请输入文字：”)

输入的都被当作字符串处理，可 int(tip)转为整数型

#三引号直接换行打印，不然要在每一个后\n

print('''i  
love  
you''')

# 三、运算符（5种）

1.算术：加减乘除，%余数，//整除，\*\*幂(次方)

2.赋值：= ，+= ，

3.比较：> ， >= ， == ，!=

4.逻辑： and or not

5.位运算：先转换为二进制 ， &与 |或 ^非 ~按位取反 <<左移位 >>右移位

## 条件表达式：

根据表达式的结果，有条件的赋值

r =a if a>b else b # a大于b时，r=a，else r=b

* **闰年== 能被4整除，但不能被100整除，或者能被400整除**

result = ’ yes ’ if (year%4==0 and year%100 !=0 ) or (year%100 ==0) else ‘no’

# 四、程序结构（3种）

顺序：

选择：

if

if …else（可省略）

if…elif …elif …else（可省略）

# if和elif 需要判断表达式的真假，而else不需要判断

循环：

## while循环：

while 真：

循环体

## for循环

for 迭代变量 in 对象：

循环体

#（一般用于次数已知，如枚举或遍历序列、迭代对象中的元素）

**range(start，end，step) 内置函数，用于生成一系列连续的整数**

start=可省略，省略代表从0开始

end=不包含

setp=可省略，省略代表步长=1

range

返回一系列连续增长的整数

工作方式类似于分片

可连续生成一个列表对象

for I in range(1,10,2)：

print(I , end = “ ”) #end 让print输出的语句在同一行显示， end = ‘分隔符’

循环嵌套：

break #跳出循环体

continue #跳过本次循环，再进入下I个循环

pass #不做任何操作，占位

# 五、元组和列表

## 序列= 列表 元组 集合 字典 字符串

**通用操作6个：**

索引+切片+相加+相乘（字典集合不支持） + in /not in + 内置函数（len max P84）

序列相加：类型需要相同（同为列表字符串元组等），序列里的元素可以不同。

## 列表：

列表的创建和删除：

1. =[ , , ]
2. =[ ]
3. list(range (10))

访问列表：

1.打印整个

2.索引访问 listname[0]

遍历列表：

for item in listname:

print(item)

for index ,item in enumerate (listname):

print (index ,item)

**# enumerate(sequence,[start=0]) 枚举函数**

将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。

增删改：

1. + ，慢
2. append()，列表末尾追加，快
3. extend()， listA.extend (listB) 把B全部添加到A中

修改：通过索引重新赋值

删除：

通过索引删， del listname[-1]

通过值删除， listname.remove(‘古道边’)

对列表统计计算：

listname.count(obj) ，obj出现次数

listname.index(obj) ，obj首次出现的位置/索引/下标

统计数值列表的和， sum( iterable [ ,start ] )

排序:

1.列表对象的方法listname.sort()

2.内置的sorted( )函数

# 1会改变原列表的排序， 2会新建副本，不改变原排序

列表推导式P98：

1.生成指定范围的数值列表

2.根据列表生成指定需求的列表

3.从列表选择符合条件的元素，组成新列表

二维列表：

列表的元素是列表

方式1.直接定义 方式2.用嵌套的for循环创建 方式3用列表推导式

list的常见函数和方法：

|  |  |
| --- | --- |
| **序号** | **函数 （与类和实例无绑定关系的function）** |
| 1 | [len(list)](https://www.runoob.com/python3/python3-att-list-len.html) |
| 列表元素个数 |
| 2 | [max(list)](https://www.runoob.com/python3/python3-att-list-max.html) |
| 返回列表元素最大值 |
| 3 | [min(list)](https://www.runoob.com/python3/python3-att-list-min.html) |
| 返回列表元素最小值 |
| 4 | [list(seq)](https://www.runoob.com/python3/python3-att-list-list.html) |
| 将元组转换为列表 |
|  | Python包含以下方法: |
| **序号** | **方法（用点调用的是方法）（与类和实例有绑定关系的function），所以用点** |
| 1 | [list.append(obj)](https://www.runoob.com/python3/python3-att-list-append.html) |
| 在列表末尾添加新的对象 |
| 2 | [list.count(obj)](https://www.runoob.com/python3/python3-att-list-count.html) |
| 统计某个元素在列表中出现的次数 |
| 3 | [list.extend(seq)](https://www.runoob.com/python3/python3-att-list-extend.html) |
| 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表） 如：listA.extend (listB) 把B全部添加到A中 |
| 4 | [list.index(obj)](https://www.runoob.com/python3/python3-att-list-index.html) |
| 从列表中找出某个值第一个匹配项的索引位置 |
| 5 | [list.insert(index, obj)](https://www.runoob.com/python3/python3-att-list-insert.html) |
| 将对象插入列表 |
| 6 | [list.pop([index=-1])](https://www.runoob.com/python3/python3-att-list-pop.html) |
| 移除列表中的一个元素（默认最后一个元素），并且返回该元素的值 |
| 7 | [list.remove(obj)](https://www.runoob.com/python3/python3-att-list-remove.html) |
| 移除列表中某个值的第一个匹配项 |
| 8 | [list.reverse()](https://www.runoob.com/python3/python3-att-list-reverse.html) |
| 反向列表中元素 |
| 9 | [list.sort( key=None, reverse=False)](https://www.runoob.com/python3/python3-att-list-sort.html) |
| 对原列表进行排序 |
| 10 | [list.clear()](https://www.runoob.com/python3/python3-att-list-clear.html) |
| 清空列表 |
| 11 | [list.copy()](https://www.runoob.com/python3/python3-att-list-copy.html) |
| 复制列表 |

## 元组：

“元”=“原”=不可变=“圆括号”

tuplename= ( , , )

用圆括号()，非中括号[]

元组中的元素不能单独修改，但可以给整个元组重新赋值元组可整体修改，因为新指了块内存。

## 元组和列表的其他区别：

元组处理速度快，只是访问时用元组 ； 列表不能作为字典的值，元组可以

# 六、字典和集合

相当于Java中的Map对象

键出现二次则覆盖第一个

键不可变，所以只能数字、字符串、元组，不能用列表

**创建：**

1、dictionary =[ ‘key1 ’:’value 1’, ‘key2’:’value 2, …]

2、空字典 dictionary= {} 或者 dictionary =dict()

3、通过**映射函数zip()**创建

dictionary= dict (zip (list1, list2))

#zip() 映射函数  
list1=[1,2,3]  
list2=['a','b','c']  
print(zip(list1,list2))  
print(list(zip(list1,list2)))  
print(tuple(zip(list1,list2)))  
print(dict(zip(list1,list2)))

<zip object at 0x00512378>

[(1, 'a'), (2, 'b'), (3, 'c')]

((1, 'a'), (2, 'b'), (3, 'c'))

{1: 'a', 2: 'b', 3: 'c'}

4、通过给定键值对创建

dictionary1 = dict( key1='value1' , key2='value2')

dict.fromkeys() #空字典

5、已存在的元组:列表

tuplekey=('1','2','3')  
listvalue=['a','b','c']  
dict1={tuplekey:listvalue}  
print(dict1)  
#结果：{('1', '2', '3'): ['a', 'b', 'c']}  
#key的tuplekey不能为列表；冒号；

**删除：**  
del dictionary #删除字典

dictionary.clear () #清空字典元素

**访问：**

初级：print (dictionary['key'])

中级：print ( dictionary['key'] if 'key' in dictionary else 'no this key')

高级：print (dictionary.get('key','可选default提示'))

**遍历：**

字典的items()方法，获取‘键-值’的元组列表

for item in dictionary.items():  
 print(item)

for key,value in dictionary.items():  
 print(key ,'的值是',value)

字典还有dict.values() 和 dict.key()方法

**修改：**

访问时赋值

**字典推导式：**

**集合**

主要作用：去重

通过 {}创建，或者用 set（）转换

如果元素重复，则只会保留一个，无序

setname.add() # 添加元素

del setname #删除整个

setname.clear()#清空

# 七、字符串

## 编码转换：

str.encode()  ->  bytes  
bytes.decode()->str

## 字符串操作：

拼接：  
+

字符串，拼接的接种方式(技巧)：<https://www.jb51.net/article/130867.htm>  
  
计算长度：  
len(string)  
  
截取（切片）：  
  
分割（分割为列表）：  
**str.spilt(分隔符，次数)**  
  
合并（列表合并为字符串）  
新字符串='分隔符'.join(列表)  
  
检索：  
str.count()  ,出现次数  
str.find()  ，存在：首次出现的索引，不存在：返回-1  
str.find()，从右侧开始找  
in  
str.index()  ，存在：首次出现的索引，不存在：报错  
str.startwith()，以指定字符串开头，是：返回True  
str.endwith()  
lower()，upper()  大小写  
**str.strip()  ,去掉空格和特殊字符（\r  \t  \n）**

格式化字符串：

1、%d %s

#直接传递：  
teststr = "i'm %s,%d years old "  
print(teststr%('chen',23))  
  
#用字典传递：  
teststr2 = "i'm %(name)s,%(age)d years old "%({'name':'chen','age':23})  
print(teststr2)

2、  str.format()

#不设置指定位置，按默认顺序  
teststr = "今天{}，{}".format("8号", "周六")   
print(teststr)  
  
#设置指定位置  
teststr2 = "今天{1}，{0}{1}".format("8号", "周六")   
print(teststr2)  
  
#设置参数：  
teststr3 = "今天{zhouji}，{data}{zhouji}".format(data="8号", zhouji="周六")   
print(teststr3)

## 字符串string内建函数（链接）：40余个，基本是str.方法名()调用

https://www.runoob.com/python3/python3-string.html

# 八、正则

用于检索和替换

^开始

$结束

\*零或多次

?零或一次

+一或多次

. 任意字符（数字、字母、各种符号）

(?!表达式):先检查字符串是否不满足表达式要求

元字符：  \w  \d  \s  \b  行定位符：   ^   $  
限定符：   ？  +  \*  {n}  {n,} {n,m}  
字符类集合：  []  
排除字符：  ^  (放在方括号中)  
选择字符：   |  
    身份证号  (^\d{15}$)|(^\d{18}$)|(^\d{17})(  \d|X|x)$  
转义字符+分组：  
    ip地址  [1-9]{1,3}(\.[0-9]){1,3}){3}   
    #斜杠用来转义点。ip第一位是1开始，后面可以0开始。小括号分组，重复三次。  
在Python中使用正则：    模式字符串(加引号+加转义)，错误  '\bm\w\*\b'，正确 '\\bm\\w\*\\b'  
    原生字符串（加引号+r即可，）  r'\bm\w\*\b'  
  
re模块：  
匹配字符串：  
match() #只在字符串开始位置匹配

search() #扫描整个字符串查找匹配

import re  
print(re.search('com', 'www.runoob.com'))

输出结果：

<re.Match object; span=(11, 14), match='com'>

匹配失败返回None

findall()  # match 和 search 是匹配一次 findall 匹配所有。

re.sub()  
  
分割：  
re.split()

string.split() 不允许多个分隔符

re.split() 允许为分隔符指定多个正则表达式

line='i am super man!'  
#分割空格m  
print(line.split(' m'))  
import re  
#分割空格或m  
print(re.split('[ m]',line))

输出：

['i am super man!11', '11']  
['i', 'a', '', 'super', '', 'an!11[', '', ']11']

基础语法远大于这些，如有需求还需要深入

# 九．函数

**形参**：定义或者函数时的参数，  叫形参，parameter，变量，'收  
**实参**：调用函数时的入参，        叫实参，argument，值，'给'  
  
**传值：**

将实际参数的值传给形参；

当实参为[不可变对象]时传值；

传值后，形参变，实参不变。

原理：**变量传递给函数后，函数会在内存中复制一个新的变量，从而不影响原来的变量。（我们称此为值传递）**

**传引用（传址）：**

将实际参数的引用传给形参；

当实参为[可变对象]时传引用；

传引用后，形参、实参变。

原理：**变量传递给函数的是一个指针，指针指向变量在内存中的位置，在函数中对变量的操作将在原有内存中进行，从而影响原有变量。（我们称此为指针传递）**

def canshu(x):  
 x+=x  
 print('函数执行后的x值：',x)  
 return  
 print('因为return了，所以这条不打印')  
str1='123'  
list1=['d','e','f']  
  
canshu(str1)  
print('调用函数后的实参str1的值：',str1)  
  
canshu(list1)  
print('调用函数后的实参list1的值：',list1)

函数执行后的x值： 123123

调用函数后的实参str1的值： 123

函数执行后的x值： ['d', 'e', 'f', 'd', 'e', 'f']

调用函数后的实参list1的值： ['d', 'e', 'f', 'd', 'e', 'f']

（Python有丰富的参数传递方式，还有关键字传递、表传递、字典传递等，

基础教程将只涉及位置传递）  
位置参数： 数量和位置要一致  
关键字参数： 调用函数时，直接给形参赋值

参数设置默认值；定义函数时，给参数一个默认值  
查看函数的参数默认值：  函数名.\_\_defaults\_\_  
  
可变参数：定义函数时，形参带\*号，接受实参放在元组中  
定义函数时，形参带\*\*号，接受实参放在字典中

动态参数列表=可变参数=参数带\*号

\*args可变参数列表

\*\*kwargs可变键值对参数列表

返回值：函数中有return时，直接结束函数的执行，即函数中return后的语句不执行  
  
局部变量：函数体内的  
  
全局变量：  
函数体外，或者函数体内+global  
  
匿名函数lambda

只是个表达式，比def简单

在 python 中没有显式的 private 和 public

限定符，如果要将一个方法声明为 private 的，只要在方法名前面加上“ \_\_ ”即可。

内建函数：<https://www.runoob.com/python3/python3-built-in-functions.html>

部分内建函数：

**map(function, iterable, ...)**会根据提供的函数对指定序列做映射。

map(square, [1,2,3,4,5]) # 计算列表各个元素的平方

**filter(function, iterable)** 函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。

tmplist = filter(is\_odd, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

**reduce()** 函数会对参数序列中元素进行累积。

dir()用来查询一个类或者对象所有属性。你可以尝试一下

>>>print dir(list)

help()用来查询的说明文档。你可以尝试一下

>>>print help(list)

# 十、类

\*/2pythoncode/practice/classlearn/

oop，程序设计思想，封装、继承、多态  
命名：驼峰，ClassName  
  
  
类里的\_\_init\_\_()方法，类似JAVA的构造函数

**self：**

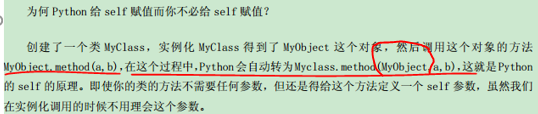
**self 是一个指向实例本身的引用**，用于访问类中的属性和方法.

python的类的方法和普通的函数有一个很明显的区别，在**类中的方法必须有个额外的第一个参数(self)。**

python的类的方法的这个特别的参数指代的是对象本身，而按照python的惯例，它用self来表示。（当然我们也可以用其他任何名称来代替，只是规范和标准在那建议我们一致使用self）。

self在python里不是关键字。**self代表当前对象的地址**。self能**避免非限定调用造成的全局变量（在类级别使用没有self的属性，是类属性，一般作为全局变量来用的）。**

**原理：**



类的成员：实例方法+数据成员（类中定义的变量/属性）  
类属性：**类中、函数体外**，通过类名.属性访问  
实例属性：定义在**类的方法中的属性**，只作用于当前实例，通过self.属性访问  
  
访问限制：保证类内部的属性和方法不被外部访问  
单下划线：

protected，类和子类访问；不能用  from  module  import  \*导入双下划线：private，类.属性访问，或者实例.类.属性访问

首尾双下划线：

一般是系统定义名字，如\_\_init()\_\_  
  
@property装饰器，将一个方法转为属性，实现用于计算的属性。用方法名来访问方法，不用加括号，更简洁。  
属性的其他操作，如可读不可改  
  
继承：class  ZiLei  (FuLei):  
子类可重写父类的方法  
子类通过spuer().\_\_init\_\_()调用父类的方法  
多实践思考积累才能真正弄明白

**list是python内置的一个类**

#动物类  
class Animal:  
 def run(self):  
 print('Animal is running..')  
  
#狗类继承动物类  
class Dog(Animal):  
 pass  
  
#创建狗类实体  
puppy =Dog()  
puppy.run()  
  
class Cat(Animal):  
 def \_\_init\_\_(self,name):  
 self.\_\_name = name  
  
 def getName(self):  
 print(self.\_\_name)  
  
 #多态，子类覆盖父类方法  
 def run(self):  
 print('cat:'+ self.\_\_name,'is running...')  
  
#lili是猫类的一个实例,根据初始化，这里要传入name  
lili =Cat('lili')  
lili.run()  
lili.getName()

Animal is running..

cat:lili is running...

lili

# 十一、模块

从本质上来说，它们都是为了更好的组织已经有的程序，以方便重复利用。

模块Modules 一个.py文件就是一个模块

import 模块

在执行时，需要在具体的变量、函数、类名前加上“模块名.”前缀

from 模块 improt ....

不用加前缀，直接访问具体的变量、函数、类名

**模块搜索目录：**

1、当前py文件所在的目录

2、到PythonPath环境变量下的每个目录查找

3、安装目录

sys.path是python的搜索模块的路径集，是一个list

一、临时添加：当前窗口有效，关闭失效

**sys.path.append("路径")**

二、增加.pth文件：当前版本有效，重新打开py文件生效

**在Lib/site-packages中增加 \*.pth，里面加入模块的路径**

**（在Lib/site-packages中放入 .py文件也可以）**

三、PythonPath下添加模块的目录：不同版本可共享，重新打开py文件生效

**包：**

**将功能相近的模块放在同一个目录，且存在\_\_init\_\_.py文件**

使用包：

import 包.模块

调用时 包.模块.变量

from 包 import 模块

调用时 模块.变量

from 包.模块 import \*

调用时 直接变量

显示每个模块的定义

print dir(模块)

记录模块名称的变量： \_\_name\_\_ ，

如果该模块不是被导入到其他程序中执行，则\_\_name\_\_= \_\_main\_\_

常用内置标准模块，

sys（环境操作） re os（操作系统功能） time json（序列化和反序列化） tkinter（GUI编程） logging shutil（高级文件操作，复制移动重命名）

第三方模块

​pip install

​pip list # 查看已经安装的第三方模块

​selenium xlrd xlwt

# 十二、异常

了解常见的异常类型：  
IndexError 索引超出序列范围

TypeError  类型不合适引发的错误

可能出错的放在try中，处理结果放在except中 （except中可同时处理多个异常）

将不直接抛出异常，而是友好的提示

没有异常时执行else中的语句

finally最终都执行，如清理环境等（关文件、释放资源）

**\*except(接受错误)和else是二选一，finally都执行**

raise 异常名('异常描述') 不想在当前函数中处理这个异常

idle程序调试

断点，go，step进入要执行的函数，

1、

try:

except:

2、

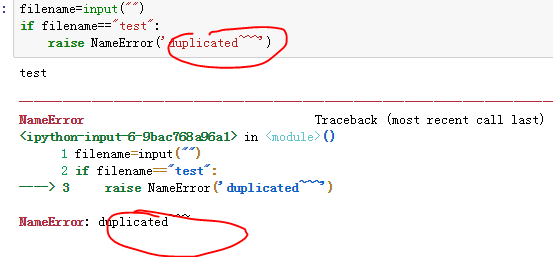
try:

finally: 最终都要执行

3、Python 使用 raise 语句抛出一个指定的异常

a、 raise自己定义一个异常，exceptions must derive from [BaseException](http://www.runoob.com/python/python-exceptions.html)

b、 如果你只想知道这是否抛出了一个异常，并不想去处理它，那么一个简单的 raise 语句就可以再次把它抛出。



# 十三、文件和目录操作

内置的open（）函数

**file = open ( filename ，mode )**

**mode：**

**r        只读，文件需存在**

**r+      读写，覆盖，文件需要存在**

**w       只写，存在则覆盖，不存在则创建**

**w+     读写，打开时清空，存在则覆盖，不存在则创建**

**a        追加，存在则追加，不存在则新建写入**

**a+      读写，存在则追加，不存在则新建读写**

带b的表示操作二进制，如rb+

file.close() 以免文件损坏，写入时先缓存然后close时才会存入磁盘

**with 表达式(一般open()函数) as 变量：  
    with-body**

file.write （str）

**file.writelines（）把字符串列表写入文件（无换行符）**

**file.read（[size]）打开的模式只能是 r或r+ ，前size个字符，省略代表全读**

**文件大时容易造成内存不足**

file.seek（）移动文件指针到新的位置，如可先file.seek(19)  再file.read(3)，读取第19个后的3个

**file.readline() 每次读1行，打开的模式只能是 r或r+**

**file.readlines()返回字符串列表，每个元素为一行的内容**

**文件大时会比较慢，此时可讲列表的内容逐行输出。**

OS模块

os模块操作目录的函数

getcwd()获取当前路径

mkdir()创建目录

rmdir()删除目录

**listdir()返回指定目录下的文件和信息**

**walk() 遍历目录，返回一个元组**

**（当前遍历的路径dirpath返回字符串，当前路径下的子目录dirnames返回列表，当前路径下的文件filenames返回列表）**

os.path模块操作与目录相关的函数

join(path ,name)拼接目录和文件名/目录

splitext() 分离文件名和扩展名

abspath (path)获取绝对路径

import os  
'''  
处理文件和目录  
os.name/getcwd/listdir/system'''  
print(os.name) #输出 nt  
print(os.getcwd())  
print(os.listdir())  
**print(os.system('ping www.baidu.com')) #执行操作系统的命令**

# 十五、wxpython

# 十八、进程和线程

spawn [spɔːn] v. n. 繁衍

多线程应用在符合并发机制的程序中，如网络程序。

**程序：**

本身是指令、数据及组织形式的描述。

**进程：**

程序运行实例。

独立数据空间，进程间不共享数据，系统调度。

如qq是一个程序，运行后，taskmgr中就有进程和pid。

**线程：**

是OS执行程序的最小单元，

线程内共享资源，系统调度。

如显示视频一个线程，播放音频一个线程。

进程、线程都是各自并行执行，哪个先完成就先结束。

**协程：**

**（Gevent是一种基于协程的Python网络库）**

Python特有，程序员调度，分解一个线程为多个“微线程”。高并发，一秒上千个请求。

创建进程：

方式1. os.fork() #只适合UNIX/Linux/Mac

方式2. multiprocessing 模块的Process类和Pool类（进程池）

p = Process( target=可调用对象，name=别名，args =传递给target的参数元组，kwargs=传递给target的参数字典)

常用的方法和属性：

p.start() #启动进程

p.join([timeout])  #等待进程实例执行结束，或者等timeout秒

p.terminate() #不管是否完成，立即终止

p.name  p.pid

from multiprocessing import Process  
def foo(i):  
 print('this is Process:',i)  
def main():  
 for i in range(3):  
 p =Process(target=foo,args=(i,))  
 p.start()  
 print('this p is num %d, pid:%d ,name:%s:'%(i,p.pid,p.name))  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

this p is num 0, pid:5508 ,name:Process-1:  
this p is num 1, pid:4812 ,name:Process-2:  
this p is num 2, pid:4908 ,name:Process-3:  
this is Process: 0  
this is Process: 1  
this is Process: 2

Process子类创建进程，一般用于处理复杂任务

先定义一个类，继承Process类

用进程池Pool创建进程，一般用于创建几十个或者上百个进程

了解即可；Pool类的常见方法，apply\_async()，阻塞（串行，一个退出后再下一个）、非阻塞（并行）。

进程间通信：

每个进程都有自己的地址空间、内存、数据栈、记录其运行状态的辅助数据，进程间不可共享。

multiprocessing模块的Queue （队列）、Pipes（管道）类等方式来交换数据。

？P392创建线程

#需要用的时候，需要深入

import threading  
def show(i):  
 print('this is Thread:',i)  
def main():  
 for i in range(3):  
 t =threading.Thread(target=show,args=(i,))  
 t.start()  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

使用Thread子类创建线程

**线程间通信：**

一个进程内的所有线程共享全局变量，互斥锁防止多线程造成全局变量混乱。

**threading模块的Lock类**，**有acquire() 和release()方法**，锁定后需要解锁，其他线程才能修改。

死锁：

两个线程都要对方已经占有的资源。用银行家算法解决（先检测资源数后分配）

threading模块的Queue队列实现线程间通信，通常用于生产者和消费者模式。

协程（微线程）

单线程高并发的，可以考虑用协程实现。#需要用的时候，需要深入

import gevent #第三方库  
def foo():  
 print('start foo')  
 gevent.sleep(0)  
 print('end foo')  
def bar():  
 print('start bar')  
 gevent.sleep(3)  
 print('end bar')  
gevent.joinall([gevent.spawn(foo),gevent.spawn(bar)])

# 其他章节：

## socket编程

*'''  
from，2的视频-4章-4.5,*

*socket=套接字  
应用程序通过套接字向网络发送请求，应答网络请求  
使主机间或者一台计算机上的进程间可以通信  
'''*import socket  
#创建socket对象  
s = socket.socket()  
  
#连接服务器  
s.connect(("127.0.0.1",6666))  
print(s.recv(1024))  
s.close()

import socket  
s =socket.socket()  
  
#绑定端口  
s.bind(("127.0.0.1",6666))  
  
#等待客户端连接  
s.listen(5)  
  
while True:  
 #建立客户端连接  
 c, addr = s.accept()  
 print('连接地址',addr)  
 c.send("wdlcome")  
 c.close()

## CGI，

Common Gateway Interface 通用网关接口 ,用来解释处理来自表单的输入信息

http server和机器上的程序进行交互的一种工具

程序需运行在网络服务器上，和语言无关。

WSGI，Python web Server Gateway interface，Python应用程序和web服务器的一种接口。

FastCGI，像是一个常驻型（long-live）的CGI，可以一直执行

## 迭代相关的内容

## 内存机制

## 序列化和反序列化：

json\_info = json.dumps(dict1)将字典转化为字符串

dict1 = json.loads(json\_info) 将字符串转化为字典

loads，d是dict，所以loads就是转换为字典

json.dump(json\_info, file)将json信息写进文件

info = json.load(file)将读取json信息

load，加载，所以load(FILE)，读取json信息