# 数据挖掘和机器学习步骤：（1）数据预处理；（2）数据建模；（3）数据验证

一、#python实现决策树功能

import numpy as np

import pandas as pd

def main():

from sklearn.datasets import load\_iris ...数据预处理

iris=load\_iris()

print(iris)

print(len(iris["data"]))

from sklearn.eross\_validation import train\_test\_split

from sklearn import tree ...数据建模（决策树，分类器）

from sklearn import metrics ...验证（准确率，混淆矩阵）

二、Python 人工网络---keras/ tensorflow(谷歌的开源框架，windows环境下不太成熟)

keras安装

1.Anaconda CMD--Anaconda 命令窗口

2.conda install mingw libpython

3.pip install keras

机器学习

1.算法

分类1） 有监督学习

无监督学习

半监督学习

2） 分类与回归

聚类

标注

3） 生成模型

判别模型

分类 C4.5

聚类 K-Means

统计学习 SVM EM

关联分析 Apriori

链接挖掘 PageRank

集装与推进 AdaBoost

分类 KNN Naive Bayes CART

2.应用：（1）关联规则；（2）聚类（用户细分，精准营销）；（3）朴素贝叶斯（垃圾邮件识别）和决策树；（4）ctr预估（搜索排序）和协同过滤（商品推荐）

3.框架

4.案例

python案例

案例1：有4个数字1，2，3，4能够组成多少个互不相同且无重复数字的三位数

分析1：可填在百位，十位，各位的数字都是1，2，3，4.组成所有的排列后去掉不满足条件的数字（数字有相同或重复的数字）

代码实现:

Num=[1,2,3,4]

i=0

For a in num:

For b in num:

For c in num:

If(a!=b) and (b!=c) and (a!=c):

i+=1

print(a,b,c)

print(“总数是:”,i)

减少冗余循环的写法：

Num=[1,2,3,4]

i=0

for a in num:

for b in num:

if(a==b):

continue;

for c in num:

if(c==b or c==a):

continue;

i+=1

print(a,b,c)

print(“总数是：”i)

python自带函数的处理办法

from itertools import permutations

for i in permutations([1,2,3,4],3):

print(i)

案例二：企业发放奖金根据利润。利润（i）低于或等于10万元时，奖金可提10%；利润高于10万元低于20万元时，低于10万元部分按照10%提成，高于10万元的部分，提成7.5%；20万到40万之间时，高于20万的可提成5%；40万到60万之间高于40万元的部分可提成3%；60万到100万之间高于60万元的部分可提成1.5%；高于100万元时，超过100万元的部分按照1%提成，键盘输入利润I,求应发奖金总数。

代码实现：

直接采用 if else方法来实现：

Import sys

‘’’

From imp import reload #重载，目的是设置python 解释器的编码规则、

Reload(sys)

Sys.setdefaultencoding(‘utf-8’)

‘’’

X=int(input(“净利润：”)) #转换成长整型

If(x<=100000):

Bonus=x\*0.1

Print(‘奖金：’,bonus,’元’)

elif(100000<x<=2000000):

Bonus=100000\*0.1+(x-100000)\*0.075

Print(‘奖金：’,bonus,’元’)

elif(200000<x<=4000000):

bonus=10000+7500+(x-200000)\*0.05

Print(‘奖金：’,bonus,’元’)

elif(400000<x<=6000000):

bonus=10000+7500+10000+(x-400000)\*0.03

Print(‘奖金：’,bonus,’元’)

elif(600000<x<=10000000):

bonus=10000+7500+10000+6000+(x-600000)\*0.015

Print(‘奖金：’,bonus,’元’)

else:

bonus=10000+7500+10000+6000+6000+(x-1000000)\*0.01

Print(‘奖金：’,bonus,’元’)

使用python函数方法实现：

Def get\_reward(i):

Rewards=0

If(i<=10):

Rewards=i\*0.1

elif(10<i<=20):

rewards=(i-10)\*0.075+ get\_reward(10)

elif(20<i<=40):

rewards=(i-20)\*0.05+ get\_reward(20)

elif(40<i<=60):

rewards=(i-40)\*0.03+ get\_reward(40)

elif(60<i<=100):

rewards=(i-60)\*0.015+ get\_reward(60)

else

rewards=(i-100)\*0.01+ get\_reward(100)

if \_\_name\_\_==’\_\_main\_\_’:

i=int(input(“净利润：”))

print(“发放的奖金为：”,get\_reward(i/10000)\*10000)

案例三：一个整数，他加上100是一个完全平方数，再加上168又是一个完全平方数，请问该数是多上？

分析：设该整数为x

X+100=n平方 x+100+168=m平方=>m平方-n平方=168 =>(m+n)\*(m-n)=168

令m+n=i，m-n=j

则 i\*j=168

由 i>0 j>0 推出 i%2=0 j%2=0

由 168=2\*2\*2\*3\*7

上面两个条件推出i与j值的范围[2,4,6,12,14,28,42,84]

反推：m=(i+j)/2和n=(i-j)/2 并且 n>0 推得 i>j

则 i=[14,28,42,84]

j=[12,6,4,2]

代码实现：

list1=[14,28,42,84]

list2=[12,6,4,2]

for i in range(0,4):

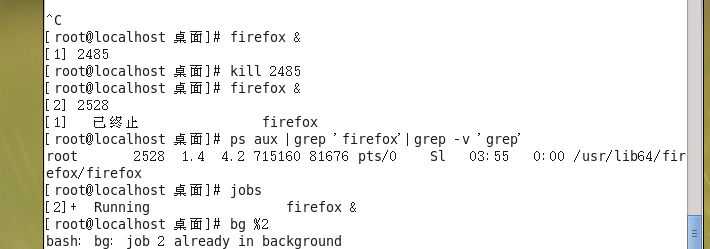
m=(list1[i]+list2[i])/2

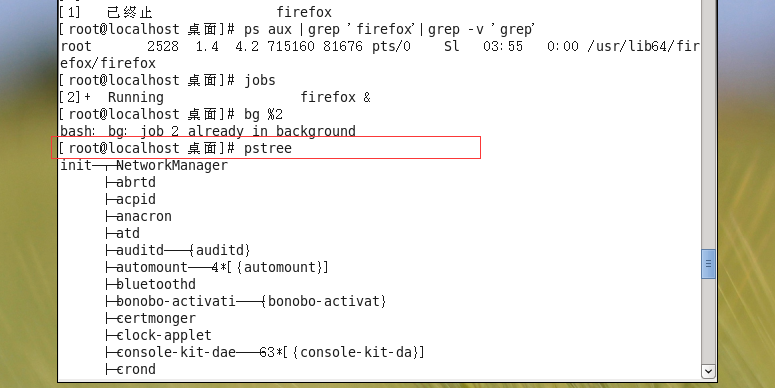
n=(list1[i]-list2[i])/2

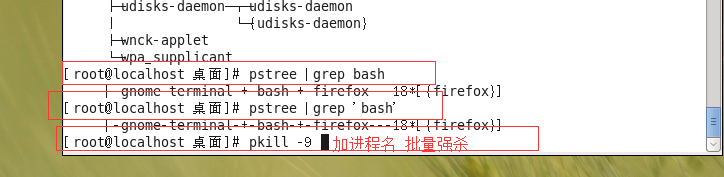
x=n\*n-100

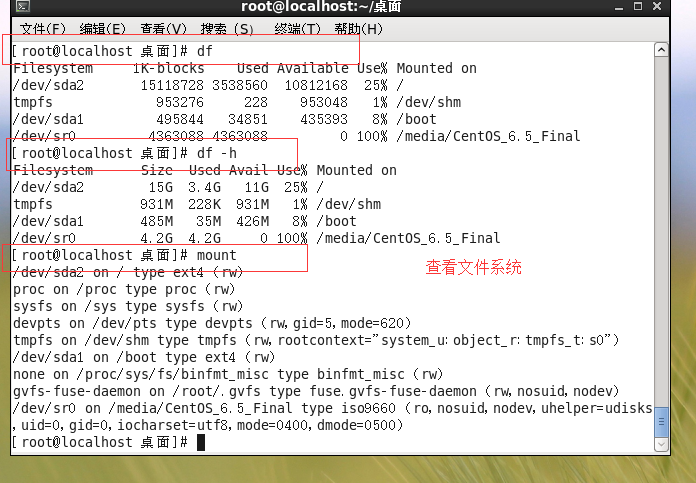
print(‘符合条件整数：’,x)

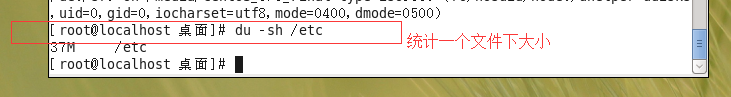
LINUX操作：

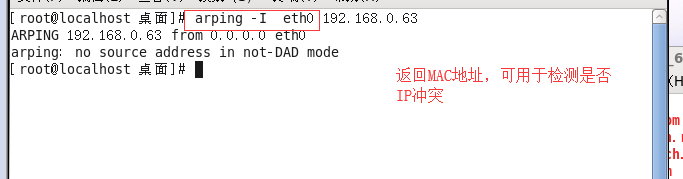


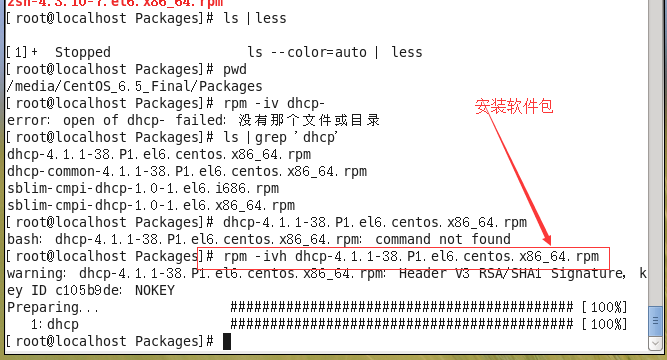


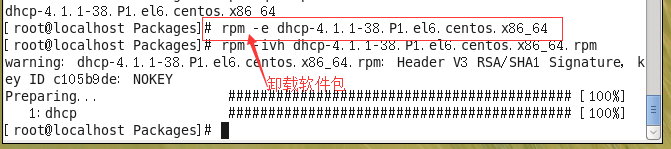


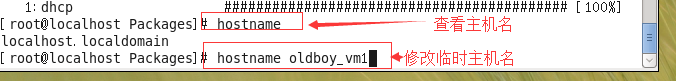












**一、python开发基础**

**1.python基础**

**1）基础**

**（1）文件后缀名为.py**

**（2）两种执行方式：py文件路径；进入解释器实时输入并获取执行结果；**

**（3）变量：只能由字母，数字，下滑线组成，但是变量不能用数字开头，关键字不能作为变量使用；**

**2）基本数据类型**

**（1）运算符：+ - \* / % // ； in , not in：判断字符是否在某个字符串中，但必须连续（子字符串，子序列）**

**（2） 数字：在python中不管数字有多大都是int ：（num,base=16）--按照进制转换；bit\_length()—至少用 多少二进制位表示；**

**字符串：魔法函数：capitalize()---首字母大写 ；casefold()/lower—转换小写，casefole()可转换特殊字符；format()—将字符串中的占位符替换为指定的值；partition()做‘分割，只能分三份；split()可以全部分割，可以指定分割几个**

**列表:可以是数字，字符串，列表，布尔值---所有的都能放进去。列表的存储形式像链表，存值的同时，存了下一个元素的地址，非连续存储，可以修改；列表有序；**

**元组：元素不可被修改（元组的一级元素），不能被增加或者删除；元组也有序；**

**字典：字典的value可以是任意值，字典的key值---------布尔值，列表，字典，集合不可以成为字典的键值；字典是无序的；不可切片；keys()，values()，items()。**

**布尔值：bool(…) None、””、0、（）、{}、[]、set ---------------🡪结果为None**

**集合：不同元素组成，元素全部无序， 可哈希（不可变数据类型）**

**求交集 &（intersection） ；求并集 |（union） ； 求差 – （difference） ；**

**交叉补集 ^（symmetric\_difference） 除交集之外的部分；**

**数字：int(…)**

**字符串：replace、find、join、strip、startswith、split、upper、lower、format**

**列表：append、extend、insert | 索引，切片，循环**

**元组：索引，切片，循环 元素不能被修改**

**字典：get、update、keys、values、items | for 索引**

**字符串格式化（%，format）：%s：传一个字符串（可以接收任何值）；%d：传一个数字；%f（%.nf）：传带有n位小数的浮点数；%.nf %%：打印百分比；**

**“I am {0},age{1},{2}”.format（”alex”,18,”seven”）：依元组下标匹配传值；**

**“I am {name},age {age},really {name}”.format(name=”seven”,age=18)：依键值传值；**

**.format(\*\*{“key”:values})**

**3）函数：逻辑结果化和过程化的一种编程方法；过程是简单特殊没有返回值的函数。**

**函数参数：（1）形参：形参变量只有在被调用时才分配内存单元，调用结束即刻释放分配的内存单元，因此形参只在函数内部有效。**

**（2）实参：可以是常量，变量，表达式，函数等，**

**（3）位置参数，关键字参数，位置参数必须在关键字参数左边；**

**默认参数def func(x,y=None)**

**（4）参数组：\*\* 字典，\* 列表**

**（5）全局变量：文件开头，没有缩进，整个文件都可以使用。**

**（6）局部变量：子程序中定义的变量就是局部变量，前方有缩进。**

**（7）global：全局变量，找到最外层的变量，nonlocal 指定上一级变量。**

**（8）前向引用：（函数即变量）**

**4）函数递归：不断调用自身，直到返回一个结果，必须要有结束条件否则会形成死循环，递归的效率不高，递归过多容易形成栈溢出。**

**5）作用域：？？？**

**6）匿名函数：lambda 定义：lambda x（形参）:x+1（处理的表达式）🡪相当于函数中return的值。如果要打印需要一个值去接收，返回一个函数地址，加（）传参即可得到结果。**

**7）函数式编程：**

**面向过程：一步一步执行，中间参与处理过程；**

**函数式编程：编程语言定义的函数 + 数学意义的函数。（1）不可变：不用变量保存状态，不修改变量；（2）第一类对象，函数即变量；（3）高阶函数 ：把函数当做参数传给另一个函数名；返回值中包含函数；（4）尾调用：在函数的最后一步调用另一个函数。**

**7.1）map函数：map（lambda x:x+1,num\_1）返回一个迭代器（只迭代一次），打印时可list化一下，处理序列中的每个元素，得到一个顺序和原有列表形式相似的’列表’。**

**7.2）filter函数：filter（函数，可迭代对象）返回的也是一个迭代器，打印时需要list化一下，该函数用于筛选，去重等,遍历序列中的每个元素，判断每个元素得到布尔值，如果是True则 留下来。**

**7.3）reduce函数： 合并一个系列，得到一个最终的值，返回一个确切的值。**

**7.4）文件处理：打开文件--🡪通过文件句柄对文件进行操作-🡪关闭文件**

**f.seek（10）🡪指定光标的位置为10，默认从文件头开始；**

**f.tell（）🡪显示当前光标的位置；**

**f.flush（）🡪将内容从内存中刷新到存储中；**

**with open（’文件名’，’打开方式’，encoding=’utf-8’，newline=’’）as f ；**

**for i in f: 循环文件，要一行取一行，节省内存；**

**8.迭代器和生成器**

**迭代器：**

**8.1）迭代器协议：对象必须提供一个next方法，执行该方法要么返回迭代中的下一项，要么就引起一个stopiteration异常，以终止迭代；**

**8.2）可迭代对象：实现了迭代器协议的对象；for调用\_\_iter\_\_()方法将对象转换成遵循迭代器协议的对象；再做一个错误处理机制；**

**生成器：可以理解为一种数据类型，这种数据类型自动实现了迭代器协议，生成器就是可迭代对象。**

**8.3）生成器函数：常规函数定义，但是用yield语句而不是用return语句返回结果，yield一次返回一个结果，自动实现迭代器协议（自动实现next（）方法，无需再去调用\_\_iter\_\_（）方法）；**

**生成器的好处是延迟计算，一次返回一个结果，生成器只能遍历一次。**

**8.4）三元表达式：’SB’ if name == ‘alex’ else ‘帅哥’：if后为True返回’SB’，’False’返回else后的结果；**

**L=[‘a’ for I in rang(10)]🡪列表解析式（列表推导式）**

**g\_l=（‘a’ for I in rang(10)）🡪生成器表达式**

**9.深浅拷贝**

**9.1）深拷贝：deepcopy（）克隆一份，需要单独模块执行copy🡪copy.deepcopy（list）；**

**9.2）浅拷贝：copy（）只拷贝第一层，更深层共享，如 l=[[1,2],3,4]，b=l.copy（）🡪重新存3,4，共享[1,2]；**

**10.装饰器：**

**本质就是函数，为其他函数添加附加功能。原则：不修改被装饰函数的源代码；不修改被装饰函数的调用方式。**

**三要素：基本实现 | 参数 | 返回值**

**10.1）装饰器的知识储备：装饰器=高阶函数+函数嵌套+闭包**

**高阶函数：函数的参数是一个函数名；函数的返回值是一个函数名；满足上述任一条件就是高阶函数。**

**def foo():**

**print(‘你好凌师傅！’)**

**def test(func):**

**print(func)**

**func()**

**test(foo)🡪test就是一个高阶函数，实参是一个函数**

**10.2） 函数嵌套：在函数内部再定义函数 def**

**10.3） 闭包：跟作用域性质类似，定义函数嵌套，优先寻找本层定义变量，再一层层往外找，最外层变量可以渗透到最内层。**

**10.4）@timmer 相当于test = timmer(test) test为被装饰函数。**

**10.5）解压序列 ：l=[1,2,3,4] a.\*d,c=l a=1,d=[2,3],c=4**

**10.6）已经写好的装饰器加参数：再做函数嵌套然后传参（闭包可以将最外层参数传到最里层）。**

**11.模块**

**11.1）time ：**

**time.time() 时间戳🡪1970至今的时间秒数，用于做计算；**

**t=time.localtime() 🡪 print(t.tm\_year) 打印具体的年月日等；**

**time.gmtime() 🡪 结构化时间 ；**

**time.mktime() 🡪 将结构化时间转换为时间戳；**

**time.strftime(“%Y-%m-%d %X”,time.localtime())--🡪将结构化时间转换为字符串时间；**

**time.strftime(“2018:07:30:17:50:36”,”%Y:%m:%d:%X”)将字符串时间转换为结构化时间；**

**time.asctime()/time.ctime() -🡪设定好格式的时间表示（星期 月份 日期 时间 年）；**

**datetime.datetime.now() 🡪显示年月日时分秒的样子；**

**11.2）random：**

**random.random() 🡪打印0-1的浮点数；**

**random.randint(1,3)🡪 1,2,3**

**random.rangrange(1,3) 🡪1,2**

**random.choice([11,22,33])🡪随机取出一个元素；**

**random.sample([11,22,33],2)🡪随机取出2个元素；**

**random.uniform(1,3) 🡪生成1-3中任意的浮点数；**

**ramdom.shuffle(item) 🡪打乱（洗牌）；**

**11.3）os：**

**os.getcwd()🡪获取当前工作目录；**

**os.chdir()🡪改变当前工作目录；**

**os.makedirs(‘name1/name2’🡪建立多层目录)；**

**os.removedirs(“name1/name2”)🡪删除文件（只删除空文件）；**

**os.listdir(“name1”)🡪显示指定文件下的文件或目录；**

**os.stat(“文件”)🡪文件相关信息介绍；**

**os.system(“dir”)🡪显示当前文件信息；**

**os.path.join(a,b)🡪默认找出操作系统的拼接符，对a,b路径进行拼接；**

**os.path.getmtime(path)🡪获取path路径文件修改时间；**

**11.4） sys：**

**sys.path 🡪返回模块的搜索路径，初始化时使用pythonpath环境变量的值；**

**sys.argv 🡪命令行参数list，可以获取命令行输入的内容；**

**sys.stdout.write(‘#’)🡪往屏幕上显示 #####；**

可用于制作进度条。

**sys.stdout.flush()🡪刷新屏幕；**

**11.5）json/pickle （序列化）： 数据交换的桥梁（适用于各种语言），json返回str，pickle返回bytes，pickle可以序列化的类型更多。**

**json.dumps(dic)🡪其他数据类型转化为字符串(“ ”)；**

**# import json**

**# dic = {'name':'alex'} ----->{"name":"alex"}----->'{"name":"alex"}'**

**# i = 8 --------> '8'**

**# s = 'hello' -------->"hello"----->'"hello"'**

**# l = [11,22] --------"[11,22]"**

**# data = json.dumps(dic)**

**# print(data)**

**json.loads() 🡪将dumps()转换的数据类型恢复成之前的形式；**

**json.dump(dict,f) 🡪在文件操作过程中直接将文件写到文件中，不用f.write();**

**json.load() 🡪直接读出来，不用f.read()操作，只能用于文件操作；**

**11.6）shelve:转换后的类型是一个字典；**

**11.7）XML：也是做数据交换，标签功能可以用字典表示出来；**

**Import xml.etree.ElementTree as ET**

**tree= ET.parse(“文件名”)🡪拿到根标签；**

**root=tree.getroot()🡪根节点;**

**root.tag 🡪标签名；**

**attrib🡪将属性组成键值对；**

**text 🡪拿文本；**

**11.8）re模块：正则表达式**

**正则：元字符🡪 . ^ $ \* + ? {}**

**. :通配符，所有字符都可以匹配上，除了 \n；**

**^：以什么开头的意思，必须在字符串的开头匹配成功，只能在开头匹配；**

**$：必须在字符串的末尾匹配，以匹配字符结尾；**

**\*：d\*🡪匹配紧挨字符的重复，0-无穷，紧挨着的字符可以出现0次；🡪贪婪匹配**

**+：匹配1-无穷次，即它挨着的字符至少出现一次；🡪贪婪匹配**

**？：匹配0次或1次；🡪惰性匹配**

**{ }：万能，，，{0，} == \***

**{1，}== +**

**{0,1} == ？**

**{6} 🡪 重复6次**

**{1,6}🡪 重复1到6中任意次数**

**[]：字符集；”x[yz]p” 🡪[‘xyp’,’xzp’]，字符集[]中没有特殊符号,除外的 -（a-z表示范围）；**

**X[^a-z]🡪非a-z的范围；\**

**\：转义字符（元字符变成普通字符，普通字符变得有意义）；**

**\d :匹配任何十进制数，相当于[0-9]**

**\D：匹配任意非数字字符，相当于[^0-9]**

**\s：匹配任何空白字符，相当于[\+\n\r\f\n]**

**\S:匹配任何非空白字符，相当于[^ \+\n\r\f\n]**

**\w：匹配任何字母数字字符，相当于[a-zA-Z0-9\_]**

**\W：匹配任何非字母数字字符，相当于[^ a-zA-Z0-9\_]**

**\b：匹配一个特殊字符边界，比如空格 & #等；**

**\特殊符号：将特殊符号转换成一般符号。**

**|：或的意思。匹配 | 前面的或者后面的；**

**（）：分组，（abc）匹配abc为组的信息；**

**re.findall（‘匹配规则’，‘被匹配字符串’），把字符串里面所有的能匹配到的结果，用列表展示出来（放到列表里）；**

**re.search()：匹配字符串中的结果，只要能找到一个就不再匹配；匹配成功返回一个对象，失败返回空；**

**re.search().group()-🡪打印出匹配结果；**

**re.match()：匹配成功返回一个对象，失败也是空；只从开始匹配；**

**re.split()：根据特定字符 分割 ，返回列表；此处容易分出空的，是个坑。**

**re.sub(匹配规则，替换内容，被替换内容，匹配次数)，做替换，结果是字符串；**

**re.subn(匹配规则，替换内容，被替换内容，匹配次数)，做替换，结果放在一个元组里，并显示匹配次数；**

**com = re.compile(‘\d+’) 🡪com.findall(字符串)🡪不用写匹配规则；编译匹配规则；**

**re.finditer()：将所有结果封装到一个迭代器，返回一个迭代器，不会将所有数据放到内存中，next()用一条取一条,取出来的也是一个对象；**

**11.9）logging模块**

**logging.basicConfig(level = logging.DEBUG， 🡪设定日志级别**

**filename = “log.log”， 🡪日志存储位置**

**filemode = “w”， 🡪日志文件添加格式**

**format = “%(asctime)s [%(lineno)d] %(message)s %(filename)s” 🡪显示格式(时间 行号 信息 文件名称)**

**)**

**logging.debug(“debug”)**

**logging.info(“info”)**

**logging.warning(“waring”)**

**logging.error(“error”)**

**logging.critical(“critical”)**

**logger:**

**----------------------------------------------------------------------------设置logger格式-----------------------------------------**

**logger = logging.getLogger() :括号不加参数相当于根，加参数的话，相当于子用户，logger是一棵树，创建对象，子对象打印时会将父节点也执行一遍（当父节点在工作时）**

**fh = logging.FileHandler()**

**ch = logging.StreamHandler()**

**fm = logger.Formatter(“%(asctime)s [%(lineno)d]”)**

**fh.setFomatter(fm) 🡪将内容格式放进来**

**ch.setFormatter(fm) 🡪将内容格式放进来**

**logger.addHandler(fh) 🡪往文件发送内容**

**logger.addHandler(ch) 🡪往屏幕发送内容**

**logger.setLeval(“DEBUG”) 🡪设定logger 日志级别**

**-----------------------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------调用logger------------------------------------------------------------------------------**

**logger.debug(“hello”)**

**logger.info(“hello”)**

**logger.warning(“hello”)**

**logger.error(“hello”)**

**logger.critical(“helllo”)**

**-----------------------------------------------------------------------------------------------------------------------------------------------**

**11.10）configparser 模块 🡪配置文件**

**生成配置文件**



句柄

Config对象

config = {}

**生成结果：**



**---------------------------------------config文件增删改查（类似字典操作）**

**config.read(‘example.ini’)**

**print(config.sections())🡪打印config内容**

**print(config.options()) 🡪拿出键**

**print(config.items()) 🡪拿出键值对**

**print(config.get()) 🡪取出键下面对应键的值**

**11.11）hashlib 提供摘要算法；**

**MD5, SHA1, SHA224, SHA256, SHA384, SHA512**

**obj = hashlib.md5()**

**obj.update(“root”,encode(“utf8”)) 🡪将root转化成密文**

**print(obj.hexdigest()) 🡪打印密文内容**

**二、面向对象**

**1.三大编程范式**

**（1）面向过程编程**

**（2）函数式编程**

**（3）面向对象编程**

**1.1）类：**

**把一类事物的相同特征和动作整合到一起就是类，类是一个抽象概念；类是一种数据结构，就好比一个模型，该模型用来表示一类事物，用它来产生真实的物体；数据结构本质上是一个字典（类属性字典）。**

**当类之间有显著的不同，并且较小的的类是较大类所需要的组件时，用组合比较好，当类之间有很多相同的功能时，提取这些共同的功能做成基类，用继承比较好。**

**对象：就是基于类而创建的一个具体的事物，也是特征和动作整合到一起；数据与函数整合到一起的产物；**

**面向对象编程：用定义类+实例/对象的方法去实现面向对象的设计；**

**实例化：立即触发init函数，由类产生对象的过程叫做实例化，实例化的结果就是一个对象，或者说就是一个实例。**

**属性：数据属性（变量）🡪某些共有特征**

**函数属性 🡪共同存在的动作**

**实例：直接使用类的\_\_init\_\_()初始化，得到自己的实例属性字典，实例首先在自己的属性字典中查找，找不到就继续往上找（类中找），类似于作用域的效果；**

**静态属性：即数据属性，类提供装饰器@property将函数属性变成数据属性，实例对象在调用函数属性的时候不用加()了，直接 . 函数名就好了；property可以将函数逻辑给封装起来，调用时感受不到；跟实例绑定；**

**类方法：@classmethod装饰的方法称为类方法，该方法可供类使用，不需要实例，方法中自动参数（cls）,该参数自动接收类名；跟类绑定；**

**静态方法：@staticmethod，不再绑定实例或者类，不能调用类变量和实例变量，是类的工具包；**

**组合：类和类之间有某种关系，做类和类的关联；**

**面向对象编程特性：**

**继承：class B(A):🡪类B继承于类A；子类继承了父类的所有属性，子类先寻找自己的属性，找不到再到父类去找，子类定义的属性和父类重名了，也不会覆盖父类。继承有两种含义：①继承基类的方法，并且做出自己的改变或者扩展（代码重用）②申明某个子类兼容于某基类，定义一个接口类，子类继承接口类，并且实现接口中定义的方法；**

**③接口继承：导入模块abc,将基类定义成接口，如：**

**class A（metaclass=abc.ABCMeta）:**

**@abc.abstractmethod**

**def a1():**

**pass**

**限定后，子类继承了他，就必须实现接口类中的方法，否则就无法实例化，实现归一化设计；**

**④继承顺序：深度优先或广度优先。对于每定义的一个类，python会计算出一个方法解析顺序（MRO）列表，这个MRO列表就是一个简单的所有基类的线性顺序列表，为了实现继承python会从这个列表上从左到右开始查找基类，知道找到第一个匹配这个属性的类为止。新式类采用广度优先继承，基类继承object类，经典类基类也是经典类（不继承object）,采用深度优先继承。**

**在子类中调用父类的方法：super().父类方法—>实现在子类中继续使用父类方法。这样做子类中不用出现父类名，不用传self参数，可扩展性强。**

**多态：不同的对象可以调用相同的方法。对象通过他们共同的属性和动作来操作及访问，而不需考虑他们具体的类；反应在运行时状态（继承的一种体现机制）；**

**封装：将某些数据属性和函数属性隐藏起来，使用层面无法知道背后的逻辑。\_a🡪表示类内部定义的属性，不影响调用（但是这个标识告诉你该属性不应该在外部被调用）；\_\_A🡪在生成类的属性字典时，Python会自动重命名为\_类名\_\_A，此时外部也能访问，如果直接访问\_\_A是不行的，在内部使用完全没有问题,但是可以在类内部封装一个接口函数可供外部调用。**

**1.2）自省：程序能够自己检测自己，又叫反射，主要用于面向对象，python提供四个函数：**

**hasattr(object,name):判断object中有没有一个name字符串对应的方法或属性，有True；**

**getattr(object,name,default值):得到object中字符串name的值，如果找不到返回default值;**

**setattr(x,y,v):x表示对象，y对应字典键，v对应字典值（x对象的属性字典）；**

**delattr(x,y):删除对象x的y属性；**

**\_\_getattr\_\_:只有在对象调用的属性不存在时会触发；**

**\_\_delattr\_\_:删除时就触发；**

**\_\_setattr\_\_:设置对象属性时就会触发；**

**isinstance(obj,cls):检查obj是否是类cls的对象；**

**issubclass(sub,super)：检查sub类是否是super类的派生类；**

**\_\_getattribute\_\_():不管实例属性是否存在，都会运行 ；**

**\_\_getitem\_\_: 使用[]操作字典时触发**

**\_\_setitem\_\_: 使用[]操作字典时触发**

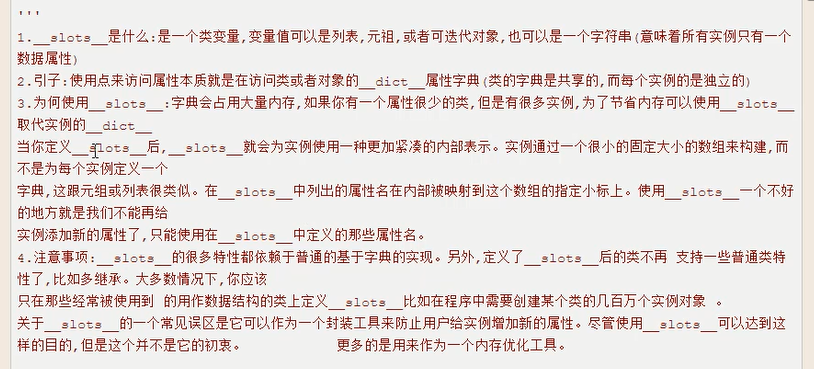
**\_\_delitem\_\_: 使用[]操作字典时触发**

**\_\_str\_\_：可以自定制对象的显示方式**

**format:format(d1)🡪\_\_format\_\_(d1):可以定制自己的格式化格式**

**\_\_repr\_\_:repr(f1)🡪f1.\_\_repr\_\_() 在解释器当中会触发，如果找不到str(),就找repr()作为str()的替代品；**

**\_\_slots\_\_:是一个类变量，变量值可以是列表，元组，或者可迭代对象，也可以是一个字符串，定义在类中的；字典会占用大量的内存，为节省内存可以使用\_\_slots\_\_代替\_\_dict\_\_(当一个累的属性不多，但产生的对象（实例）特别多时)，定义完\_\_slots\_\_后，类的\_\_dict\_\_方法失效，\_\_slots\_\_限定了属性的个数。**



**1.3）动态导入模块：\_\_import\_\_(‘m1.t’) 🡪此时导入的只是模块m1,无论几层导入的都是最顶层模块；import \* 不导入定义的私有属性；importlib模块可导入整个字符串（importlib.import\_module()）；**

**三、网络编程**

**的厂财务**

**四、WEB框架—用于写网站**

**大V委外**

**五、设计模式+算法**

**大部分**

**六、项目阶段**