**Python2-1**

**字典基础操作**

**• 通过{ }操作符创建字典**

**• 通过dict()工厂方法创建字典**

**• 通过fromkeys()创建具有相同值的默认字典**

adict **= {'name':'bob'**,**'age':**25**}**

print**(**adict**)**

**{'name': 'bob', 'age': 25}**print**(**dict**(['ab'**,**'cd'**,**('name'**,**'niu')]))**

**{'a': 'b', 'c': 'd', 'name': 'niu'}**print**({}**.fromkeys**(['niu'**,**'wang'**,**'li']**,**'male'))**

**{'niu': 'male', 'wang': 'male', 'li': 'male'}**

**• 字典是映射类型,意味着它没有下标,访问字典中的值需要使用相应的键**

adict **= {'name':'bob'**,**'age':**25**}**print**(**20 **in** adict**)**print**('name' in** adict **)  
for** key **in** adict**:** print**('%s: %s' %(**key,adict**[**key**]))**print**('%(name)s is %(age)s years old' %** adict**) 会在字典里匹配name age**

**通过key更新新值 如果key不在字典里 则向字典里添加新值**

**• 通过del可以删除字典中的元素或整个字典**

**• 使用内部方法clear()可以清空字典**

**• 使用pop()方法可以“弹出”字典中的元素**

**• 使用字典键查找操作符[ ],查找键所对应的值**

**• 使用in和not in判断键是否存在于字典中**

**• len():返回字典中元素的数目**

**• hash():本身不是为字典设计的,但是可以判断某个对象是否可以作为字典的键**

**1. 支持新用户注册,新用户名和密码注册到字典中**

**2. 支持老用户登陆,用户名和密码正确提示登陆成功**

**3. 主程序通过循环询问进行何种操作,根据用户的选择,执行注册或是登陆操作**

**import** getpass  
adict **={}  
def zhuce():** name **=** input**('用户名：')  
 if** name **not in** adict**:** pasword**=**input**('密码：')** adict**[**name**]=**pasword  
 **else:** print**('用户已经存在')  
def denglu():** name**=**input**('用户名：')** password**=**getpass.getpass**('密码：')** # for password in adict[name]:  
 # print('登陆成功')  
 **if** adict.get**(**name**) !=** password**:** print**('失败')  
 else:** print**('成功')  
def show():** cmds**={'0':**zhuce,**'1':**denglu**}** promt**="""(0)注册  
 (1)登陆  
 (2)退出"""  
  
 while True:** choice **=** input**(**promt**)**.strip**()[**0**]  
 if** choice **not in '012':** print**('无效输入')  
 break  
 if** choice **== '2':** print**('Bye')  
 break** cmds**[**choice**]()  
if** \_\_name\_\_ **== '\_\_main\_\_':** show**()**

**1. Windows文本文件的行结束标志是\r\n**

**2. 类unix文本文件的行结束标志是\n**

**3. 编写程序,将unix文本文件格式转换为windows文本文件的格式**

# import sys  
# def unix2dos(fname):  
# dst\_name=fname + '.txt'  
# with open(fname) as src\_obj:  
# with open(dst\_name,'w') as dst\_obj:  
# for line in src\_obj:  
# line = line.rstrip() +'\r\n'  
# dst\_obj.write(line)  
# if \_\_name\_\_ == '\_\_main\_\_':  
# unix2dos(sys.argv[1])

**1. 在屏幕上打印20个#号**

**2. 符号@从20个#号穿过**

**3. 当@符号到达尾部,再从头开始**

**import** sys  
**import** time  
n**=**0  
print**('#'\***20,end**='')  
while True:** print**('\r%s@%s' %('#'\***n,**'#'\*(**19**-**n**))**,end**='') \r回车不换行** sys.stdout.flush**() #立即输出到屏幕 不放入缓冲区** n**+=**1  
 time.sleep**(**0.3**)  
 if** n **==** 20**:** n**=**0

**• 数学上,把set称做由不同的元素组成的集合,集合(set)的成员通常被称做集合元素**

**• 集合对象是一组无序排列的可哈希的值**

**• 集合有两种类型**

**– 可变集合set**

**– 不可变集合frozenset**

**• 集合支持用in和not in操作符检查成员**

**• 能够通过len()检查集合大小**

**• 能够使用for迭代集合成员**

**• 不能取切片,没有键**

**• |:联合,取并集**

**• &:交集**

**• -:差补**

aset **=** set**('hello')**print**(**aset**)**len**(**aset**)  
for** ch **in** aset**:** print**(**ch**)**bset**=('how')**# print((aset) |( bset))  
a **=** aset **|** bset #并集  
aset **&** bset #交集  
aset **-**bset #差补 aset有，bset没有的元素

**• set.add():添加成员**

**• set.update():批量添加成员**

**• set.remove():移除成员**

**• s.issubset(t):如果s是t的子集,则返回True,否则返回False**

**• s.issuperset(t):如果t是s的超集,则返回True,否则返回False**

**• s.union(t):返回一个新集合,该集合是s和t的并集**

**• s.intersection(t):返回一个新集合,该集合是s和t的交集**

**• s.difference(t):返回一个新集合,该集合是s的成员,但不是t的成员**

aset**=**set**('hello')**print**(**aset**)**aset.add**('new')**print**(**aset**)**aset.update**(['hello'**,**'world'])**print**(**aset**)**cset **=** set**('ho')**print**(**cset.issubset**(**aset**))** #cset是aset的子集吗  
print**(**aset.issuperset**(**cset**))** #aset是cset的超集吗  
aset.intersection**(**bset**)** #aset&bset 交集  
aset.union**(**bset**)** #aset|bset 并集  
aset.difference**(**bset**)** #aset-bset 差补

**集合差补应用**

**/etc/passwd /tmp/passwd**

**With open(‘/etc/passwd’) as f :**

**Aset = set(f)**

**With open(‘/tmp/passwd’) as c:  
 bset = set(f)**

**With open(‘/tmp/result.txt’,’w’) as e:  
 e.writelines(aset - bset)**

**时间表示方式**

**• 时间戳timestamp:表示的是从1970年1月1日00:00:00开始按秒计算的偏移量**

**• UTC(Coordinated Universal Time,世界协调时)亦即格林威治天文时间,世界标准时间。在中国为UTC+8。DST(Daylight Saving Time)即夏令时**

**• 元组(struct\_time):由9个元素组成**

**struct\_time元组**

****

**• time.localtime([secs]):将一个时间戳转换为当前时区的struct\_time。secs参数未提供,则以当前时间为准**

**• time.gmtime([secs]):和localtime()方法类似,gmtime()方法是将一个时间戳转换为UTC时区(0时区)的struct\_time**

**• time.time():返回当前时间的时间戳**

**• time.mktime(t):将一个struct\_time转化为时间戳**

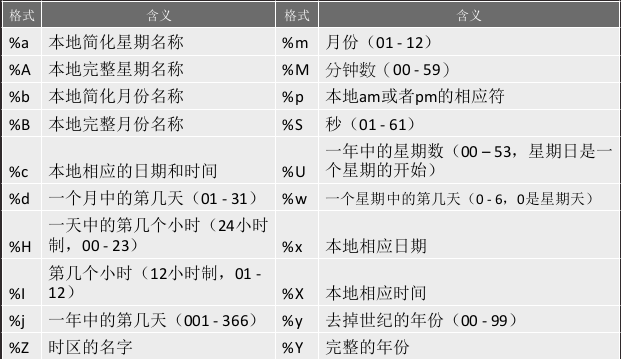
**• time.sleep(secs):线程推迟指定的时间运行。单位为秒**

**• time.asctime([t]):把一个表示时间的元组或者struct\_time表示为这种形式:‘Sun Jun 2023:21:05 1993’。如果没有参数,将会将time.localtime()作为参数传入**

**• time.ctime([secs]):把一个时间戳(按秒计算的浮点数)转化为time.asctime()的形式**

**• time.strftime(format[, t]):把一个代表时间的元组或者struct\_time(如由time.localtime()和time.gmtime()返回)转化为格式化的时间字符串。如果t未指定,将传入time.localtime()**

**• time.strptime(string[, format]):把一个格式化时间字符串转化为struct\_time。实际上它和strftime()是逆操作**

****

**datetime模块方法**

**• datetime.today():返回一个表示当前本地时间的datetime对象**

**• datetime.now([tz]):返回一个表示当前本地时间的datetime对象,如果提供了参数tz,则获取tz参数所指时区的本地时间**

**• datetime.strptime(date\_string, format):将格式字符串转换为datetime对象**

**• datetime.ctime(datetime对象):返回时间格式字符串**

**• datetime.strftime(format):返回指定格式字符串**

**使用timedelta可以很方便的在日期上做天days,小时hour,分钟,秒,毫秒,微妙的时间计算**

# from datetime import datetime,timedelta  
# dt = datetime.now()  
# print(dt)  
# days = timedelta(days=100,hours=3)  
# print(dt+days)

**什么是异常**

**• 当python检测到一个错误时,解释器就会指出当前流已经无法继续执行下去,这时候就出现了异常**

**• 异常是因为程序出现了错误而在正常控制流以外采取的行为**

**• 这个行为又分为两个阶段:**

**– 首先是引起异常发生的错误**

**– 然后是检测(和采取可能的措施)阶段**

**• 当程序运行时,因为遇到未解的错误而导致中止运行,便会出现traceback消息,打印异常**

****

**try-except语句**

**• 定义了进行异常监控的一段代码,并且提供了处理异常的机制**

**• 可以把多个except语句连接在一起,处理一个try块中可能发生的多种异常**

**• 异常也可以有参数,异常引发后它会被传递给异常处理器**

**• 当异常被引发后参数是作为附加帮助信息传递给异常处理器的**

**• 在try范围中没有异常被检测到时,执行else子句**

**• 在else范围中的任何代码运行前,try范围中的所有代码必须完全成功**

**• finally子句是无论异常是否发生,是否捕捉都会执行的一段代码**

**• 如果打开文件后,因为发生异常导致文件没有关闭,可能会发生数据损坏。使用finally可以保证文件总是能正常的关闭**

**try:** num **=** int**(**input**('numeber:'))** #把有可能发生异常的语句放到try中  
 result **=** 100 **/**num  
**except (**ValueError,ZeroDivisionError**):** #捕获异常  
 print**('无效输入，必须输入非零数字')  
except (**KeyboardInterrupt,EOFError**):** print**('\nBye')  
else:** print**(**result**)** #异常不发生的语句放到else中  
**finally:** print**('DONE')** #不管是否发生异常都要执行的语句 放到finally中  
#try-except 和 try-finally 这两个组合用的比较多

**• 对文件系统的访问大多通过python的os模块实现**

**• 该模块是python访问操作系统功能的主要接口**

**• 有些方法,如copy等,并没有提供,可以使用shutil模块作为补充**

****

**• 把数据写入文件时,常规的文件方法只能把字符串对象写入。其他数据需先转换成字符串再写入文件 。**

**• python提供了一个标准的模块,称为pickle。使用它可以在一个文件中储存任何python对象,之后又可以把它完整无缺地取出来**

**• 分别调用dump()和load()可以存储、写入**