回顾

- 自定义模块
- 模块的两种执行方式
- `__name__``__file__`,`__all__`
- 模块导入的多种方式
- 相对导入
- random:
    - random.random():
    - random.uniform(a,b):
    - random.randint(a,b):
    - random.shuffle(x):
    - random.sample(x,k):

今日内容

常用模块的介绍:

- time, datetime
- os, sys
- hashlib, json, pickle, collections

time: 和时间相关

封装了获取时间戳和字符串形式的时间的一些方法。

- time.time():获取时间戳
- time.gmtime([seconds]):获取格式化时间对象:是九个字段组成的
- time.localtime([seconds]):获取格式化时间对象:是九个字段组成的
- time.mktime(t):时间对象 -> 时间戳
- time.strftime(format[,t]):把时间对象格式化成字符串
- time.strptime(str,format):把时间字符串转换成时间对象

```
import time
# 获取时间戳
# 时间戳：从时间元年（1970 1 1 00:00:00）到现在经过的秒数。
# print(time.time())  # 1558314075.7787385   1558314397.275036
```

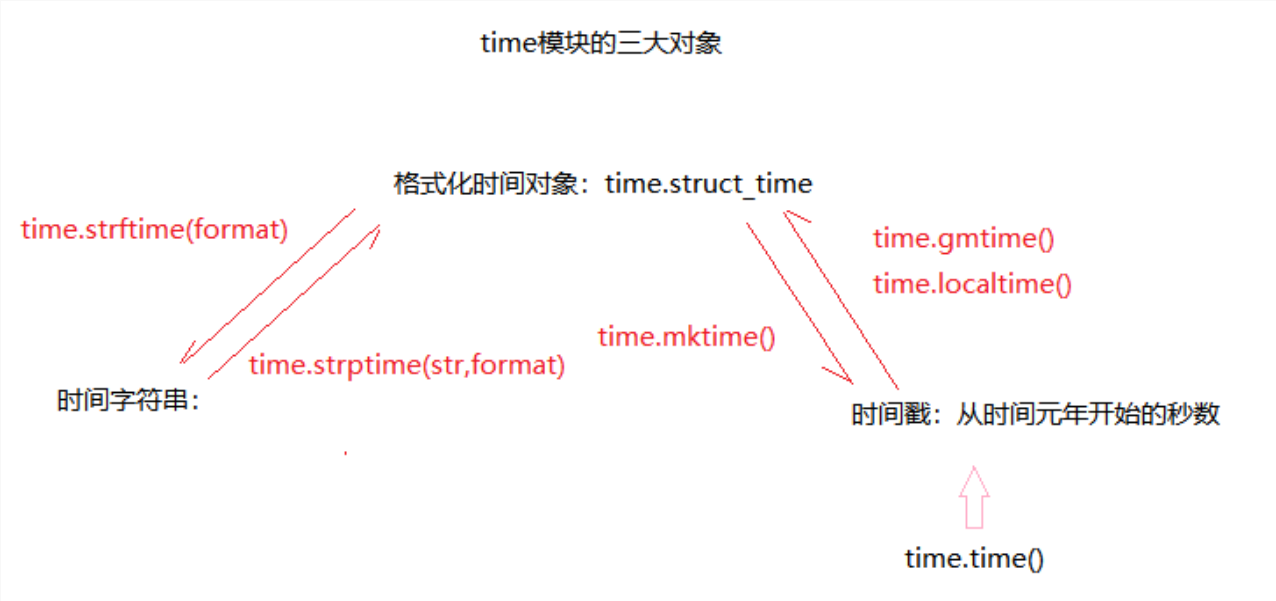
```
# 获取格式化时间对象:是九个字段组成的。
# 默认参数是当前系统时间的时间戳。
# print(time.gmtime())  # GMT:
# print(time.localtime())
# print(time.gmtime(1))        # 时间元年过一秒后，对应的时间对象

# 时间对象 -> 时间戳
# t1 = time.localtime()    # 时间对象
# t2 = time.mktime(t1)     # 获取对应的时间戳
# print(t2)
# print(time.time())


# 格式化时间对象和字符串之间的转换
# s = time.strftime("year:%Y %m %d %H:%M:%S")
# print(s)

# 把时间字符串转换成时间对象
# time_obj = time.strptime('2010 10 10','%Y %m %d')
# print(time_obj)
```

time模块三大对象之间的转换关系：



其他方法：

- time.sleep(x):休眠x秒.

datetime:日期时间相关

包含了和日期时间相关的类.主要有:

- date:需要年,月,日三个参数
- time:需要时,分,秒三个参数
- datetime:需要年,月,日,时,分,秒六个参数.
- timedelta:需要一个时间段.可以是天,秒,微秒.

获取以上类型的对象,主要作用是和时间段进行数学运算.

timedelta可以和以下三个类进行数学运算:

- datetime.time,datetime.datetime,datetime.timedelta

练习:

- 显示当前日期前三天是什么时间.
- 显示任意一年的二月份有多少天.

```
# 普通算法:根据年份计算是否是闰年.是:29天,否:28
# 用datetime模块.
# 首先创建出指定年份的3月1号.然后让它往前走一天.
year = int(input("输入年份:"))
# 创建指定年份的date对象
d = datetime.date(year,3,1)
# 创建一天 的时间段
td = datetime.timedelta(days=1)
res = d - td
print(res.day)
```

os模块

和操作系统相关的模块,主要是文件删除,目录删除,重命名等操作.

```
import os
# 和文件操作相关,重命名,删除
# os.remove('a.txt')
# os.rename('a.txt','b.txt')

# 删除目录,必须是空目录
# os.removedirs('aa')

# 使用shutil模块可以删除带内容的目录
# import shutil
# shutil.rmtree('aa')

# 和路径相关的操作,被封装到另一个子模块中:os.path
```

```
# res = os.path.dirname(r'd:/aaa/bbb/ccc/a.txt') # 不判断路径是否存在.
# print(res)
#
# # 获取文件名
# res = os.path.basename(r'd:/aaa/bbb/ccc/a.txt')
# print(res)

# 把路径中的路径名和文件名切分开,结果是元组.
# res = os.path.split(r'd:/aaa/bbb/ccc/a.txt')
# print(res)

# 拼接路径
# path = os.path.join('d:\\','aaa','bbb','ccc','a.txt')
# print(path)

# 如果是/开头的路径,默认是在当前盘符下.
# res = os.path.abspath(r'/a/b/c')
# 如果不是以/开头,默认当前路径
# res = os.path.abspath(r'a/b/c')
# print(res)


# 判断
# print(os.path.isabs('a.txt'))
# print(os.path.isdir('d:/aaaa.txt')) # 文件不存在.False
# print(os.path.exists('d:/a.txt'))
# print(os.path.isfile('d:/asssss.txt'))      # 文件不存在.False
```

sys模块

和解释器操作相关的模块.

主要两个方面:

- 解释器执行时获取参数:sys.argv[x]
- 解释器执行时寻找模块的路径:sys.path

例如:有a.py内容如下:

```
import sys
print('脚本名称:',sys.argv[0])
print('第一个参数:',sys.argv[1])
print('第二个参数:',sys.argv[2])
```

使用脚本方式运行:

```
python a.py hello world
```

json模块

JavaScript Object Notation:java脚本兑现标记语言.

已经成为一种简单的数据交换格式.

序列化:将其他数据格式转换成json字符串的过程.

反序列化:将json字符串转换其他数据类型的过程.

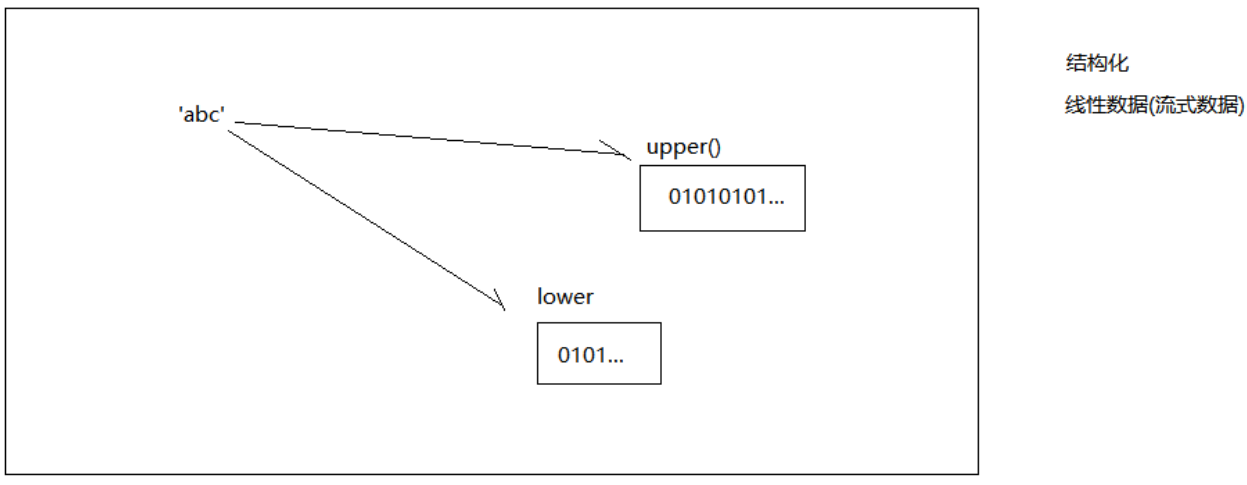
涉及到的方法:

- json.dumps(obj):将obj转换成json字符串返回到内存中.
- json.dump(obj,fp):将obj转换成json字符串并保存在fp指向的文件中.
- json.loads(s):将内存中的json字符串转换成对应的数据类型对象
- json.load(f):从文件中读取json字符串,并转换回原来的数据类型.

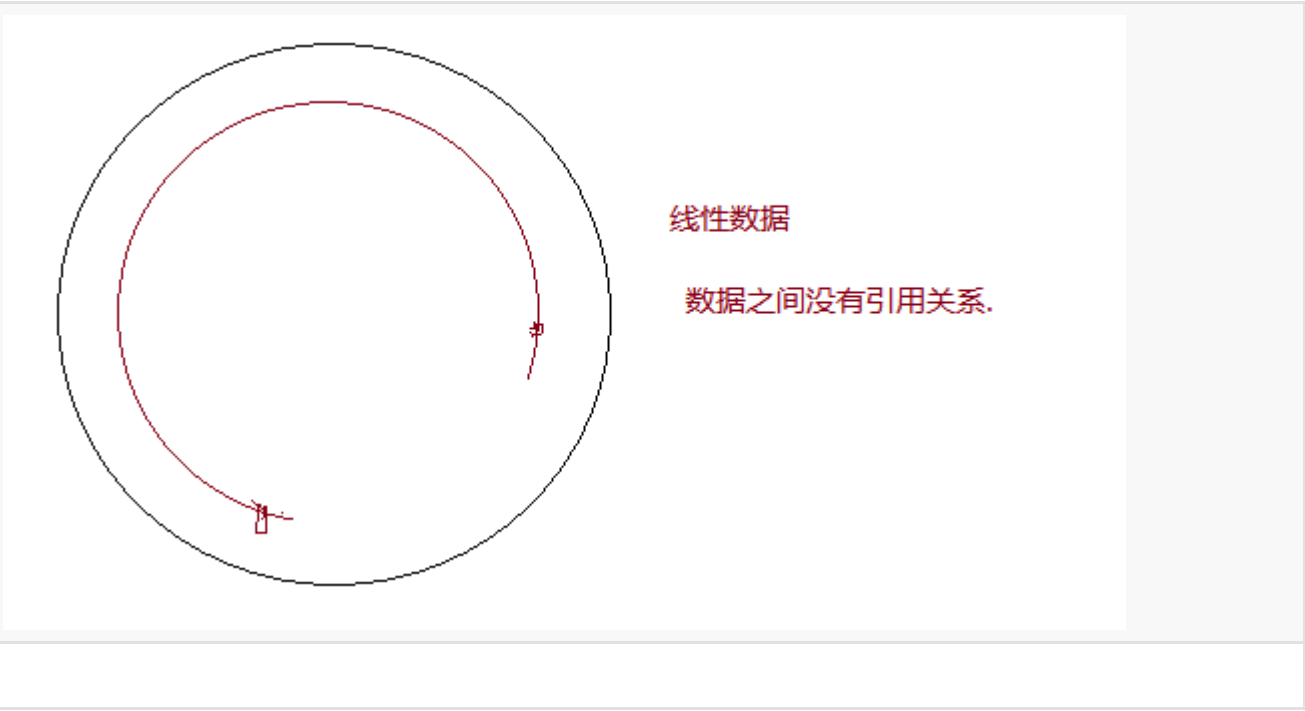
注意:

- json并不能序列化所有的数据类型:例如:set.
- 元组数据类型经过json序列化后,变成列表数据类型.
- json文件通常是一次性写入,一次性读取.但是可以利用文件本身的方式实现:一行存储一个序列化json字符串,在反序列化时,按行反序列化即可.

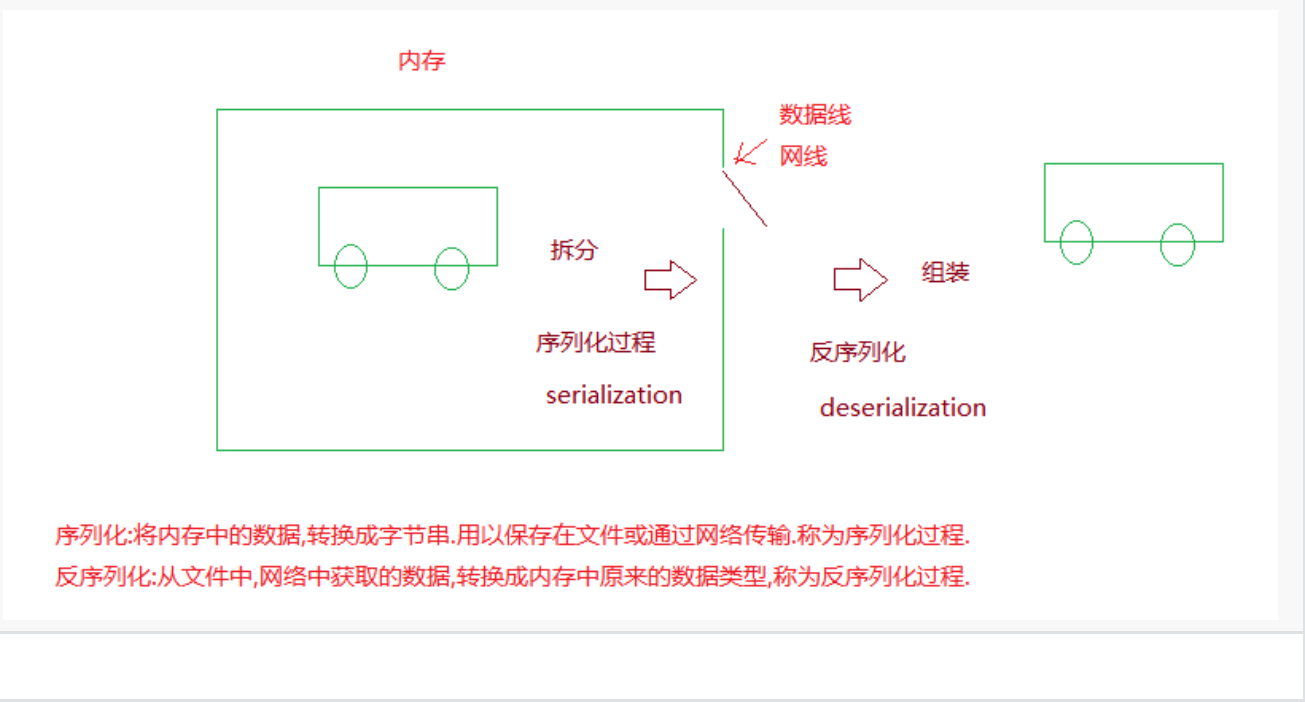
内存中的数据:结构化的数据



磁盘上的数据:线性数据.



序列化比喻：



pickle模块

python专用的序列化模块.

和json的方法一致.

区别在于:

```
json:
1.不是所有的数据类型都可以序列化.结果是字符串.
2.不能多次对同一个文件序列化.
3.json数据可以跨语言

pickle:
1.所有python类型都能序列化,结果是字节串.
2.可以多次对同一个文件序列化
3.不能跨语言.
```

hashlib模块

封装一些用于加密的类.

`md5(),...`

加密的目的:用于判断和验证,而并非解密.

特点:

- 把一个大的数据,切分成不同块,分别对不同的块进行加密,再汇总的结果,和直接对整体数据加密的结果是一致的.
- 单向加密,不可逆.
- 原始数据的一点小的变化,将导致结果的非常大的差异,'雪崩'效应.

```python
# 注册,登录程序:
def get_md5(username,passwd):
    m = hashlib.md5()
    m.update(username.encode('utf-8'))
    m.update(passwd.encode('utf-8'))
    return m.hexdigest()


def register(username,passwd):
    # 加密
    res = get_md5(username,passwd)
    # 写入文件
    with open('login',mode='at',encoding='utf-8') as f:
        f.write(res)
        f.write('\n')

def login(username,passwd):
    # 获取当前登录信息的加密结果
    res = get_md5(username, passwd)
    # 读文件,和其中的数据进行对比
    with open('login',mode='rt',encoding='utf-8') as f:
        for line in f:
```

```
            if res == line.strip():
                return True
        else:
            return False

while True:
    op = int(input("1.注册 2.登录 3.退出"))
    if op == 3 :
        break
    elif op == 1:
        username = input("输入用户名:")
        passwd = input("输入密码:")
        register(username,passwd)
    elif op == 2:
        username = input("输入用户名:")
        passwd = input("输入密码:")
        res = login(username,passwd)
        if res:
            print('登录成功')
        else:
            print('登录失败')
```

不同的加密对象,结果长度不同,长度越长,越耗时.常用的是md5.

总结

- 自定义模块
- random
- time
- datetime
- os
- sys
- json,pickle
- hashlib
- collections