网络知识

数据库连接

连接mysql需要下载mysql驱动

MySQLdb

仅支持python2

```python
import MySQLdb

# 创建连接
con = MySQLdb.connect(host, port, username, password, params)

con.commit()    # 提交事务
con.rollback()  # 回滚事务
con.close()     # 关闭连接

# 游标操作
cu = con.cursor()                                           # 创建游标
cu.execute(sql)                                             # 执行sql
cu.execute("insert into user values(%s,%s)",[(1,2),(2,3)])  # 执行sql：插入多条
cu.close()                                                  # 关闭游标
cu.fetchone()                                               # 取下一个查询结果
cu.fetchall()                                               # 获取所有记录列表
cu.rowcount                                                 # sql影响的行数
```

pymysql

支持python3

```python
import pymysql

# 建立连接
con = pymysql.connect(host, port, username, password,dbname,charset='utf8')
```

创建数据库

```python
def createDatabase(dbname):
  con = pymysql.connect(host, port, username, password,dbname,charset='utf8')
  cur = conn.cursor()

  cur.execute("create database if not exists " + dbname)

  cur.close()
  conn.close()
```

创建表

```python
def createTable(sql):
    con = pymysql.connect(host, port, username, password,dbname,charset='utf8')
```

```
    cur = conn.cursor()
    cur.execute(sql)
    rows = cur.fetchall()

    cur.close()
    conn.close()
    return rows
```

新增记录：事务提交

```python
def insert(sql):
    con = pymysql.connect(host, port, username, password,dbname,charset='utf8')

    cur = conn.cursor()
    cur.execute(sql)
    conn.commit()

    cur.close()
    conn.close()
```

删除/修改记录：事务提交、sql传参

```python
def delete(sql,*params):
    con = pymysql.connect(host, port, username, password,dbname,charset='utf8')

    cur=conn.cursor()
    cur.execute(sql,params)
    conn.commit()

    cur.close()
    conn.close()
```

查询记录：获取结果集

```python
def select(self, sql):
    con = pymysql.connect(host, port, username, password,dbname,charset='utf8')

    cur = conn.cursor()
    cur.execute(sql)
    rows = cur.fetchall()

    cur.close()
    conn.close()
    return rows
```

网络连接

网络请求

```python
import urllib
request.urlopen("http://www.baidu.com")
```

示例

```python
from urllib.request import urlopen
from urllib.parse import urlencode

# 请求地址、请求参数：转为url编码
requrl = "http://www.xxx.com/xxx"
params = urlencode({"username":"admin","password":123456})

'''GET请求'''
res = urlopen(requrl + '?' + params)

'''POST请求'''
res = Request(requrl, params.encode('ascii'))  # 将url编码转bytes

# 将响应内容转换为文本
```

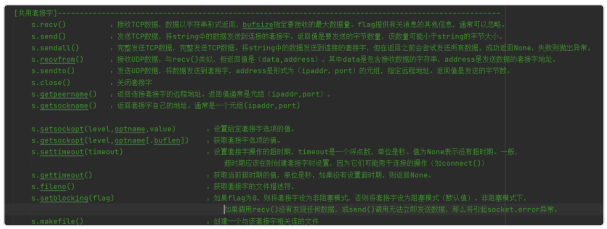
```python
resStr = res.read().decode()

```

webSocket

套接字的建立

```python
import socket

# 创建套接字
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

'''服务端套接字'''
sock.bind((localhost,8080))   # 绑定地址
sock.listen()                 # 开始tcp监听：可设置允许挂起的数量
sock.accept()                 # 接收连接，阻塞式

'''客户端套接字'''
sock.connect((localhost,8080))   # 连接，出错抛出异常
sock.connect_ex((localhost,8080))# 连接，出错返回错误码
```

套接字的使用



Internet 模块

电子邮件

创建邮件

```python
import email

# 参数2为内容类型，可为plain、html、base64
mail = MIMEText('内容','plain','utf-8') # 邮件内容
mail['From'] = Header('u1','utf-8')     # 发送者
mail['To'] = Header('u2','utf-8')       # 接收者
mail['Subject'] = Header('u3','utf-8')  # 主题

# 附件
mail1 = MIMEText('内容','base64','utf-8') # 邮件内容
mail1['Content-type'] = 'application/octet-stream'
mail1['Content-Disposition'] = 'attachment;filename="文件名"'
mail.attach(mail1)

# 图片
mail2 = MIMEMultipar('related')
mail.attach(mail2)

```

邮件发送服务

```python
import smtplib # qq发件

sm = smtplib.SMTP('localhost')
sm.sendmail('发件人地址','收件人地址列表',邮件)

# 使用第三方服务器

```

```
sm.connect('smtp.xx.com',8888)  # ssl端口
sm.login('用户名','口令')
```

网络爬虫

请求源码

```python
import request

head = {
  'User-Agent':'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36'
}
res = request.get(url,head)
res.encoding = 'gn2312'
str = res.text
```

解析源码：安装pip install bs4

```python
# lxml是解析器
dom = bs4.BeautifulSoup(html文本,"lxml")

# select查询：响应一个set集合
set = dom.select('#main > .photo')[0]['src']

# 如果select指定了一个具体的Tag，获取值的办法
val = set[0]['属性名']
```

文档解析

JSON 解析

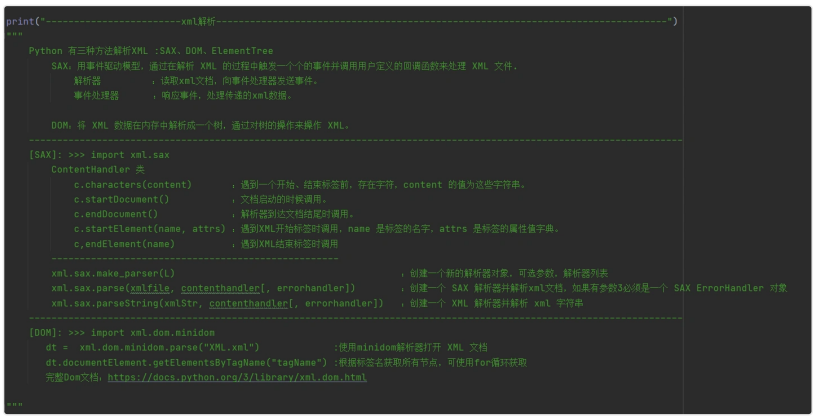
```
print("--------------------------json-----------------------------------------------------")
"""
    Python3 中可以使用 json 模块来对 JSON 数据进行编解码，它包含了两个函数：
        json.dumps(): 对数据进行编码。
        json.loads(): 对数据进行解码。
    在json的编解码过程中，python 的原始类型与json类型会相互转换，具体的转化对照如下：
    [编码对照表]: python >>> json----------------------------------------------------------
        dict                                >>>                 object
        list, tuple                         >>>                 array
        str                                 >>>                 string
        int, float, int- & float-derived Enums  >>>             number
        True                                >>>                 true
        False                               >>>                 false
        None                                >>>                 null
    [解码对照表]: json >>> python----------------------------------------------------------
        object                              >>>                 dict
        array                               >>>                 list
        string                              >>>                 str
        number (int)                        >>>                 int
        number (real)                       >>>                 float
        true                                >>>                 True
        false                               >>>                 False
        null                                >>>                 None
    [API]:https://docs.python.org/3/library/json.html


"""
```

示例

```python
import json

# Python 字典类型转换为 JSON 对象
data1 = {
    'no' : 1,
    'name' : 'Runoob',
    'url' : 'http://www.runoob.com'
}

json_str = json.dumps(data1)
print ("Python 原始数据：", repr(data1))
```

```python
print ("JSON 对象：", json_str)

# 将 JSON 对象转换为 Python 字典
data2 = json.loads(json_str)
print ("data2['name']: ", data2['name'])
print ("data2['url']: ", data2['url'])
```

XML 解析



```
print("-----------------------xml解析-----------------------------------------------------------------------------------------")
"""
    Python 有三种方法解析XML ：SAX、DOM、ElementTree
        SAX，用事件驱动模型，通过在解析 XML 的过程中触发一个个的事件并调用用户定义的回调函数来处理 XML 文件，
            解析器          ，读取xml文档，向事件处理器发送事件，
            事件处理器      ，响应事件，处理传递的xml数据，

        DOM，将 XML 数据在内存中解析成一个树，通过对树的操作来操作 XML，
    ----------------------------------------------------------------------------------------------------------------------
    [SAX]: >>> import xml.sax
        ContentHandler 类
            c.characters(content)       ，遇到一个开始、结束标签前，存在字符，content 的值为这些字符串，
            c.startDocument()           ，文档启动的时候调用，
            c.endDocument()             ，解析器到达文档结尾时调用，
            c.startElement(name, attrs) ，遇到XML开始标签时调用，name 是标签的名字，attrs 是标签的属性值字典，
            c,endElement(name)          ，遇到XML结束标签时调用
        ---------------------------------------------------
        xml.sax.make_parser(L)                                      ，创建一个新的解析器对象，可选参数，解析器列表
        xml.sax.parse(xmlfile, contenthandler[, errorhandler])      ，创建一个 SAX 解析器并解析xml文档，如果有参数3必须是一个 SAX ErrorHandler 对象
        xml.sax.parseString(xmlStr, contenthandler[, errorhandler])  ，创建一个 XML 解析器并解析 xml 字符串
    ----------------------------------------------------------------------------------------------------------------------
    [DOM]: >>> import xml.dom.minidom
       dt =  xml.dom.minidom.parse("XML.xml")             :使用minidom解析器打开 XML 文档
       dt.documentElement.getElementsByTagName("tagName") :根据标签名获取所有节点，可使用for循环获取
       完整Dom文档，https://docs.python.org/3/library/xml.dom.html
"""
```

示例

Python

```python
import xml.sax

class MovieHandler( xml.sax.ContentHandler ):
    def __init__(self):
        self.CurrentData = ""
        self.type = ""
        self.format = ""
        self.year = ""
        self.rating = ""
        self.stars = ""
        self.description = ""

    # 元素开始调用
    def startElement(self, tag, attributes):
        self.CurrentData = tag
        if tag == "movie":
            print ("*****Movie*****")
            title = attributes["title"]
            print ("Title:", title)

    # 元素结束调用
    def endElement(self, tag):
        if self.CurrentData == "type":
            print ("Type:", self.type)
        elif self.CurrentData == "format":
            print ("Format:", self.format)
        elif self.CurrentData == "year":
            print ("Year:", self.year)
        elif self.CurrentData == "rating":
            print ("Rating:", self.rating)
        elif self.CurrentData == "stars":
            print ("Stars:", self.stars)
        elif self.CurrentData == "description":
            print ("Description:", self.description)
        self.CurrentData = ""

    # 读取字符时调用
    def characters(self, content):
        if self.CurrentData == "type":
            self.type = content
        elif self.CurrentData == "format":
            self.format = content
        elif self.CurrentData == "year":
            self.year = content
        elif self.CurrentData == "rating":
            self.rating = content
        elif self.CurrentData == "stars":
            self.stars = content
        elif self.CurrentData == "description":
            self.description = content
```

```python

if ( __name__ == "__main__"):

    # 创建一个 XMLReader
    parser = xml.sax.make_parser()
    # 关闭命名空间
    parser.setFeature(xml.sax.handler.feature_namespaces, 0)

    # 重写 ContextHandler
    Handler = MovieHandler()
    parser.setContentHandler( Handler )

    parser.parse("movies.xml")
```

INI 解析

ini 文件格式

```ini
[user]
name=zhangsan
age=10

[account]
name=lisi
height=180
```

解析

```python
import configparser
import os

def get_ini()
  # 当前路径
  current_path = os.path.dirname(os.path.abspath(__file__))
  ini_file = os.path.join(current_path, '../conf/test.ini')
  con = configparser.ConfigParser()
  con.read(ini_file)
  # 获取配置
  username = con.get("user","name")
```

正则表达式

```python
import re  # 正则表达式模块

"""
    re.match("正则表达式","源文本","标志位")                    :匹配头部，返回位置（0,a），如果头部匹配不到，返回None。返回一个match对象。
    re.search("正则表达式","源文本","标志位")                   :返回匹配的第一个结果，否则返回None。
    re.sub("正则表达式","替换的字符串/函数","源文本","替换次数")   :替换源文本中的匹配项,替换次数默认为0。

    re.compile("正则表达式[,标志位]")                          :编译正则表达式，生成一个表达式对象(RegexObject)，供match、search使用。
        aaa = re.compile("正则")
        mmm = aaa.match("源文本","标志位")
        # group()、start()、end()、span()//分别返回子匹配字符串、起始位置、结束位置、起始与结束位置(参数为子表达式索引，默认为0)
    .
    re.findall("正则","源文本",匹配起点,匹配终点)  :找到全部匹配结果，返回一个列表。起点默认为0，终点默认为长度。
    re.finditer("正则","源文本",匹配起点,匹配终点) :找到全部匹配结果，返回一个迭代器。
    re.split("正则","源文本",分割次数,"标志位") :用正则结果作为分割符分割文本。分割次数默认为0(全部分割)，返回一个列表，()包围表达式时，列表中包含正则匹配结果。
"""
```

```python
"""
    标志位，
        re.I    使匹配不区分大小写
        re.L    做本地化识别（locale-aware）匹配
        re.M    多行匹配，影响 ^ 和 $
        re.S    使 . 匹配包括换行在内的所有字符
        re.U    根据Unicode字符集解析字符。这个标志影响 \w, \W, \b, \B
        re.X    该标志通过给予你更灵活的格式以便你将正则表达式写得更易于理解。

"""
"""
    当表达式包含多个子表达式时，(.*) are (.*?) .*
    re.match(...).group()   获取整体表达式匹配结果
    re.match(...).group(n)  获取第n个子表达式匹配结果（可多个参数，此时结果放在元组）
    re.match(...).groups()  获取所有子表达式匹配结果，结果放在元组
"""
```

示例: