Python 模块

目录

4 个功能: dump dumps load loads

Dumps:将字典转换成字符串

```
>>> dic={'age':23,'job':'student'}
>>> dic_str=json.dumps(dic)
>>> dic_str
'{"age": 23, "job": "student"}'
>>> type(dic_str)
<type 'str'>
>>> type(dic)
<type 'dict'>
```

Loads:将字符串转换成字典

```
>>> dic_obj=json.loads(dic_str)
>>> dic_obj
{u'age': 23, u'job': u'student'}
>>> type(dic_obj)
<type 'dict'>
```

Dump:字典存储到文件中

```
>>> import json
>>> dic={'age':23,'job':'student'}
>>> with open('abc.json','w') as f:
...     json.dump(dic,f)
...
>>>
```

写入abc.json文件中

Load:读取文件中的内容

```
>>> with open('abc.json','r') as f:
...     obj=json.load(f)
...     print(obj)
...
{u'age': 23, u'job': u'student'}
```

==========

pickle 模块

4 个功能: dumps loads dump load

dumps:存入数据

>>> dic={'age':23,'job':'student'}

>>> byte_data=pickle.dumps(dic)　　将字典转换成字符串

```
>>> byte_data
"(dp0\nS'age'\np1\nI23\nsS'job'\np2\nS'student'\np3\ns."
```

loads:读取数据

```
>>> obj=pickle.loads(byte_data)
>>> obj
{'age': 23, 'job': 'student'}
```

存储数据到文件中，使用 dump 和 load。由于 **pickle 写入的是二进制数据，** 所以打开方式需要以 wb 和 rb 的模式。

==========configparser 模块,只能在 python3 中使用

Configparser 是用来读取配置文件的包,配置文件格式如下:

```
1 [db]          ([]为 section)
2 db_host = 127.0.0.1   (key=value 为 option)
3 db_port = 69
4 db_user = root
5 db_pass = root
6 host_port = 69
7
8 [concurrent]
9 thread = 10
10 processor = 20
```

案例:

(1): config.sections()得到所有 sections 的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
['a.ini']
>>> config.sections()
['db', 'concurrent']
```

(2):获取指定 section 中的 option 的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
['a.ini']
>>> r=config.options("db")
>>> r
['db_host', 'db_port', 'db_user', 'db_pass', 'host_port']
>>>
```

获取指定section的option的值

(3):获取指定 option 指点的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
['a.ini']
>>> r=config.get("db","db_host")
>>> r
'127.0.0.1'
>>>
```

获取指定option指点的值

(4):获取指点 section 的所有值,会把 option 的键和值全部输出

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
['a.ini']
>>> r=config.items("db")
>>> r
[('db_host', '127.0.0.1'), ('db_port', '69'), ('db_user', 'root'), ('db_pass', 'root'), ('host_port', '69')]
>>>
```

获取指点section的所有值

(5): 修改某个 option 的值，如果不存在则会出创建

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
['a.ini']
>>> config.set("db","db_port","69")
>>> config.set("db","db_port","80")
>>> config.write(open("a.ini","w"))
>>>
```

(6): 检查 section 或 option 是否存在，返回 bool 值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.has_section("section")
False
>>> config.has_option("section","option")
False
>>>
```

(7):添加 section 节点

```
>>> import configparser
>>> import configparser
>>>
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.add_section("user")
>>> config.write(open("config.ini","w"))
>>>
```

添加section节点，然后把节点写入一个ini文件中

(8) : 删除 section 节点

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.add_section("user")
>>> config.write(open("config.ini","w"))
>>> config.remove_section("user")
True
>>> config.write(open("config.ini","w"))
```

删除使用remove_section

(9)：删除指定 section 组内的 option

```
>>> import configparser

>>> config=configparser.ConfigParser()

>>> config.read("a.ini")

>>> config.remove_option("db","host_port")

>>> config.write(open("a.ini","w"))
```

(10)：添加 option 的键和值：

XML

解析 XML

读取 xml 文件的内容：

```
>>> from xml.etree import ElementTree as ET

>>> tree=ET.parse("xo.xml")      ====》解析 XML

>>> root=tree.getroot()          ==➔得到 Root 节点

>>> print(root.tag,root.attrib)      ==》tag 是 string,attrib 是字典{}
```

Root 的子节点

Root 的子节点的 key

Root 的子节点的子节点的 key

```
>>> root[0][0].tag
'rank'
>>> root[0][1].tag
'year'
>>> root[0][2].tag
'gdppc'
>>> root[0][3].tag
'neighbor'
>>> root[0][4].tag
'neighbor'
>>> root[0][5].tag
```

Element.findall()只查找**直接的孩子**，返回所有符合要求的 Tag 的 Element，而 Element.find()只返回符合要求的第一个 Element。如果查看 Singapore 的 year 的值，可以

```
>>> for country in root.findall('country'):
...     if country.attrib['name']=='Singapore':
...             years=country.findall('year')
...             print(years[0].text)
...
2026
>>> for country in root.findall('country'):
...     if country.attrib['name']=='Singapore':
...             years=country.findall('year')
...             print(years[1].text)
...
```

```
>>> for country in root.findall('country'):
...     root.remove(country)
...
>>> tree=ET.ElementTree(root)
>>> tree.write('a1.xml',encoding="utf-8")
```

删除指定的属性值

```
>>> for country in root.findall("country"):
...     rank=country.find("rank").text          rank 的内容
...     name=country.get("name")                country 的 name 属性
...     print(name,rank)
...
Liechtenstein 2
Singapore 5
Panama 69
```

修改 XML

```
>>> for rank in root.iter("rank"):
...     new_rank=int(rank.text)+1
...     rank.text=str(new_rank)
...     rank.set("update","NO")
...
>>> ET.dump(root)
<data>
    <country name="Liechtenstein">
        <rank update="NO" updated="yes">3</rank>
        <year>2023</year>
        <gdppc>141100</gdppc>
        <neighbor direction="E" name="Austria" />
        <neighbor direction="W" name="Switzerland" />
    </country>
    <country name="Singapore">
        <rank update="NO" updated="yes">6</rank>
        <year>2026</year>
        <gdppc>59900</gdppc>
        <neighbor direction="N" name="Malaysia" />
    </country>
    <country name="Panama">
        <rank update="NO" updated="yes">70</rank>
        <year>2026</year>
        <gdppc>13600</gdppc>
        <neighbor direction="W" name="Costa Rica" />
        <neighbor direction="E" name="Colombia" />
    </country>
</data>
>>> tree.write("output.xml")
```

修改属性的值，dump将结果显示在屏幕上

```
>>> import xml.etree.ElementTree as ET
>>> tree=ET.parse("output.xml")
>>> root=tree.getroot()
>>> for rank in root.iter("rank"):
...     del rank.attrib["updated"]
...
>>> ET.dump(root)
<data>
    <country name="Liechtenstein">
        <rank update="NO">3</rank>
        <year>2023</year>
        <gdppc>141100</gdppc>
        <neighbor direction="E" name="Austria" />
        <neighbor direction="W" name="Switzerland" />
    </country>
    <country name="Singapore">
        <rank update="NO">6</rank>
        <year>2026</year>
        <gdppc>59900</gdppc>
        <neighbor direction="N" name="Malaysia" />
    </country>
    <country name="Panama">
        <rank update="NO">70</rank>
        <year>2026</year>
        <gdppc>13600</gdppc>
        <neighbor direction="W" name="Costa Rica" />
        <neighbor direction="E" name="Colombia" />
    </country>
</data>
>>>
```

小结：关于 *class* xml.etree.ElementTree.Element 属性相关

- attrib　　为包含元素属性的字典
- keys()　　返回元素属性名称列表
- items()　　返回(name,value)列表
- get(*key*, *default=None*)　　获取属性
- set(*key*, *value*)　　# 跟新/添加 属性
- del xxx.attrib[key]　　# 删除对应的属性
- ET.dump(root)　　很有用可以在屏幕上显示输出结果,(括号中输入不同的值，会输出不同的结果)

新建 xml