[**python\_读取yaml文件**](https://www.cnblogs.com/wwho/p/14446332.html)

<https://www.cnblogs.com/wwho/p/14446332.html>

## yaml简介

1. yaml : Yet Another Markup Language ：另一种标记语言。yaml 是专门用来写配置文件的语言，非常简洁和强大,更直观，更方便，有点类似于json格式。在自动化测试用的相当多,所以需要小伙伴们要熟练掌握，把测试数据存放在yaml文件，也可以进行参数化。
2. yaml基本语法规则：

* 大小写敏感
* 使用缩进表示层级关系
* 缩进时不允许使用Tab键，只允许使用空格。
* 缩进的空格数目不重要，只要相同层级的元素左侧对齐即可
* '#'表示注释，从这个字符一直到行尾，都会被解析器忽略，这个和python的注释一样

1. yaml支持的数据结构有三种：

* 对象：键值对的集合，又称为映射（mapping）/ 哈希（hashes） / 字典（dictionary）
* 数组：一组按次序排列的值，又称为序列（sequence） / 列表（list）
* 纯量（scalars）：单个的、不可再分的值。字符串、布尔值、整数、浮点数、Null、时间、日期

## yaml的使用

### 1.使用pip安装pyyaml模块

pip install pyyaml -i https://pypi.douban.com/simple

### 2.yaml的基本使用

在同一目录下新建 test\_data.yaml 和 read\_yaml.py文件。

yaml文件：  
test\_data.yaml

-

phone : 1326557

key : 01829bab25a57e4c11eff56db56d1f94

except : 200

py文件：  
read\_yaml.py

# -\*- coding:utf-8 -\*-

import yaml

f = open('test\_data.yaml', 'r', encoding='utf-8')

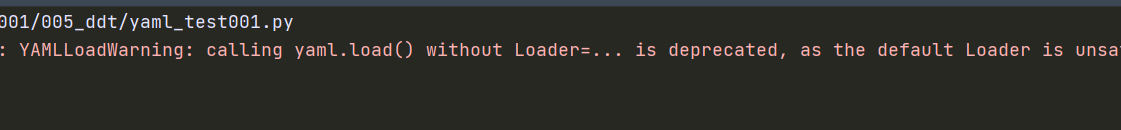
cfg = f.read()

d = yaml.load(cfg, Loader=yaml.FullLoader) # 用load转字典

# yaml5.1版本后弃用了yaml.load(file)这个用法，因为觉得很不安全，5.1版本之后就修改了需要指定Loader，通过默认加载器（FullLoader）禁止执行任意函数

# Loader=yaml.FullLoader 加上这行代码，告警就没了

print(d)

去掉告警的方法看以上备注：  


运行结果：

[{'phone': 1326557, 'key': '01829bab25a57e4c11eff56db56d1f94', 'except': 200}]

### 3.封装yaml

以上是yaml文件的读取的简单用法，将她进行封装好，后续使用直接调用即可。

import yaml

import os

class GetYaml():

def \_\_init\_\_(self, file\_path):

# 判断文件是否存在

if os.path.exists(file\_path):

self.file\_path = file\_path

else:

print('没有找到%s文件路径' % file\_path)

self.data = self.read\_yaml()

def read\_yaml(self):

with open(self.file\_path, 'r', encoding='utf-8')as f:

p = f.read()

return p

def get\_data(self, key=None):

result = yaml.load(self.data, Loader=yaml.FullLoader)

# 判断key是否存在

if key == None:

return result

else:

return result.get(key)

if \_\_name\_\_ == '\_\_main\_\_':

read\_yaml = GetYaml('test\_data.yaml')

r = read\_yaml.get\_data()

print(r)

### 4.python常用数据类型在yaml中的书写格式

做接口自动化或者UI自动化的时候，要进行数据分离，把测试数据写在yaml文件中，即在yaml文件中写测试用例。  
把常用的格式列出来：

#### 1.yaml是键值对的格式，python中字典也是键值对的形式。

yaml文件的格式：

user: admin

psw: 123456

python3.7中的格式：

{'user': 'admin', 'psw': 123456}

#### 2.字典嵌套字典

yaml文件的格式：

input:

user: admin

psw: 123456

python3.7中的格式：

{'input': {'user': 'admin', 'psw': 123456}}

#### 3.列表中嵌套字典（list）

yaml里面写一个数组，前面加一个'-'符号.  
yaml文件的格式：

- test1: 123456

- test2: 111111

- test3: 222222

python3.7中的格式：

[{'test1': 123456}, {'test2': 111111}, {'test3': 222222}]

#### 4.list嵌套dict

yaml文件格式：

- user: admin1

pwd: '123456'

- user: admin2

pwd: '123321'

- user: admin3

pwd: '123457'

python3.7中的格式

[{'user': 'admin1', 'pwd': '123456'},

{'user': 'admin2', 'pwd': '123321'},

{'user': 'admin3', 'pwd': '123457'}]

#### 5.dict嵌套list

yaml文件中的格式：

input1:

- admin1

- '123456'

input2:

- admin2

- '123321'

input3:

- admin3

- '123457'

python3.7中的格式：

{'input1': ['admin1', '123456'],

'input2': ['admin2', '123321'],

'input3': ['admin3', '123457']}

后续会讲yaml怎么和数据驱动结合起来用！敬请关注！