NSD Devops DAY04

1 案例1：通过本机发送邮件

1.1 问题

编写一个send_mail.py脚本，实现以下功能：

1. 创建bob和alice帐户
2. 编写发送邮件件程序，发件人为root，收件人是本机的bob和alice帐户

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建bob和alice帐户

```
[root@localhost day12]# ls /home/
Student  zabbix  zhangsan
[root@localhost day12]# useradd bob
[root@localhost day12]# useradd alice
[root@localhost day12]# ls /home/
alice  bob  Student  zabbix  zhangsan
```

步骤二：编写发送邮件件程序，发件人为root，收件人是本机的bob和alice帐户

```
[root@localhost day12]# vim send_mail.py

import smtplib
from email.mime.text import MIMEText
from email.header import Header
#邮件正文有三个参数：第一个为文本内容，第二个设置文本格式plain，第三个utf-8设置
message = MIMEText('Python邮件发送测试\n', 'plain', 'utf8')
标准邮件需要三个头部信息： From, To, 和 Subject
#发送者信息
```

Top

```
10.  message['From'] = Header('root@localhost', 'utf8')
11.  #接收者信息
12.  message['To'] = Header('bob@localhost', 'utf8')
13.  #主题信息
14.  message['Subject'] = Header('mail test', 'utf8')
15.
16.  sender = 'root@redhat.com'        #发送方
17.  receivers = ['bob@localhost', 'alice@126.com']    #收件方
18.  smtp_obj = smtplib.SMTP('localhost')    #用localhost发邮件
19.  # smtplib负责发送邮件
20.  smtp_obj.sendmail(sender, receivers, message.as_string())
```

SMTP是发送邮件的协议，Python内置对SMTP的支持，可以发送纯文本邮件、HTML邮件以及带附件的邮件。

Python对SMTP支持有smtplib和email两个模块，email负责构造邮件，smtplib负责发送邮件。

Python SMTP 对象使用 sendmail 方法发送邮件：

```
01.  smtp_obj.sendmail(sender, receivers, message.as_string())
```

参数说明：

sender: 邮件发送者地址。

receivers: 字符串列表，邮件发送地址。

message.as_string(): 发送消息 ，str模式

由于可以一次发给多个人，所以recives传入一个列表，邮件正文是一个str，as_string()把MIMEText对象变成str。

步骤三：测试脚本执行

```
01.  [root@localhost day12]# python3 send_mail.py
02.  [root@localhost day12]# mail -u bob
03.  Heirloom Mail version 12.5 7/5/10.  Type ? for help.
04.  "/var/mail/bob": 1 message 1 new
05.  >N  1 =?utf8?q?root=40loca  Mon Jul 30 09:36  18?663 ""
06.  &1
07.  From root@redhat.com Mon Jul 30 09:36:44 2018
08.  Return-Path: <root@redhat.com>
09.  X-Original-To: bob@localhost.tedu.cn
10.  Content-Type: text/plain; charset="utf8"
```

Top

```
From: root@localhost@room8pc16.tedu.cn
To: bob@localhost@room8pc16.tedu.cn
Subject: mail test
Date: Mon, 30 Jul 2018 09:36:44 +0800 (CST)
Status: R

Python邮件发送测试
&
```

2 案例2：通过互联网服务器发送邮件

2.1 问题

编写一个mail_client.py脚本，实现以下功能：

1. 通过自己互联网注册的邮箱，为其他同学互联网邮箱发邮件

2.2 方案

导入sys模块，用sys.argv方法获取get_web函数实参，让用户在命令行上提供http://www.tedu.cn和/tmp/tedu.html两个参数，调用get_web函数实现如下功能：

1)导入urllib模块，使用urllib模块的urlopen函数打开url（即网址），赋值给html

2)以写方式打开/tmp/tedu.html文件

3)以循环方式：

读html获取的数据，保存到data

将data写入/tmp/tedu.html

4)关闭html

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：环境准备

使用SMTP协议发送的邮件，需要先查看您的发件人邮箱是否有开启SMTP协议，如没有需要开启，测试使用的是126.com的邮箱作为发信人邮箱，开启SMTP协议如下

1. 先登录到126.com邮箱，如图-1所示：

Top

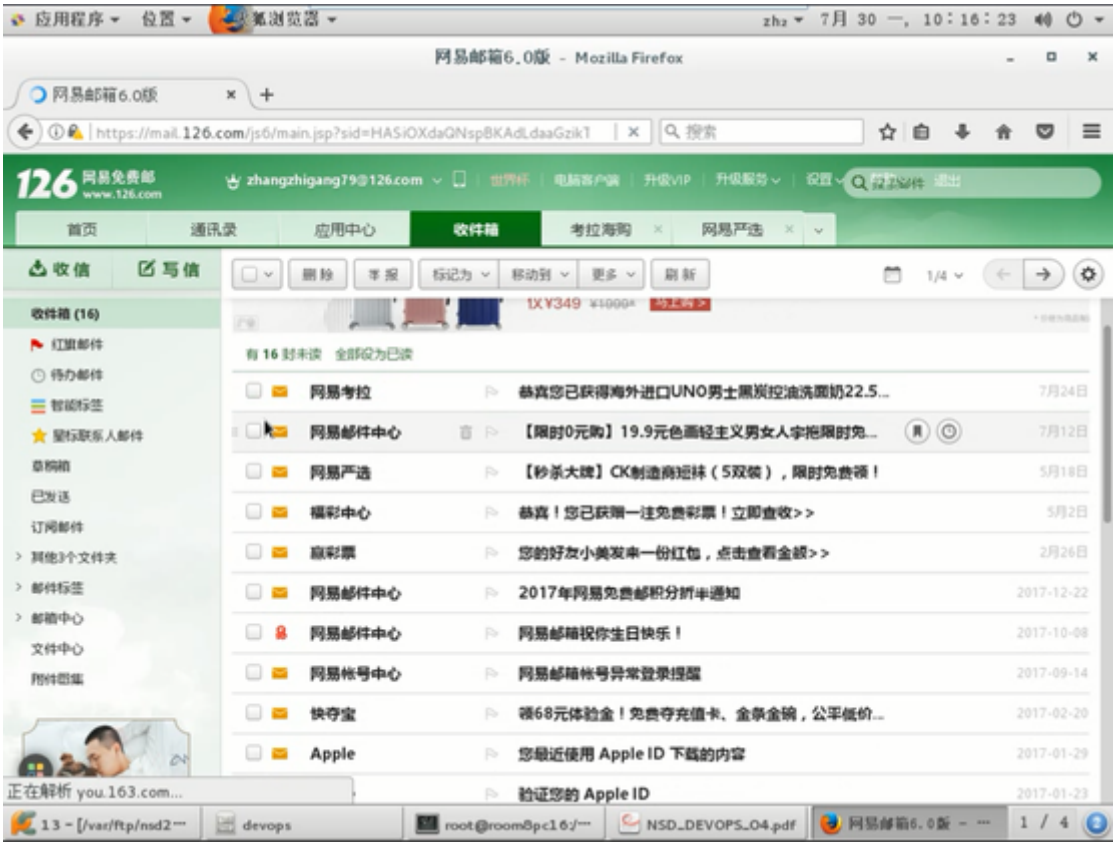

图-1

2. 看到邮箱上面的功能栏中有一个“设置”的选项，单击该选项，然后选择下拉菜单的“POP3/SMTP/IMAP”，如图-2所示：

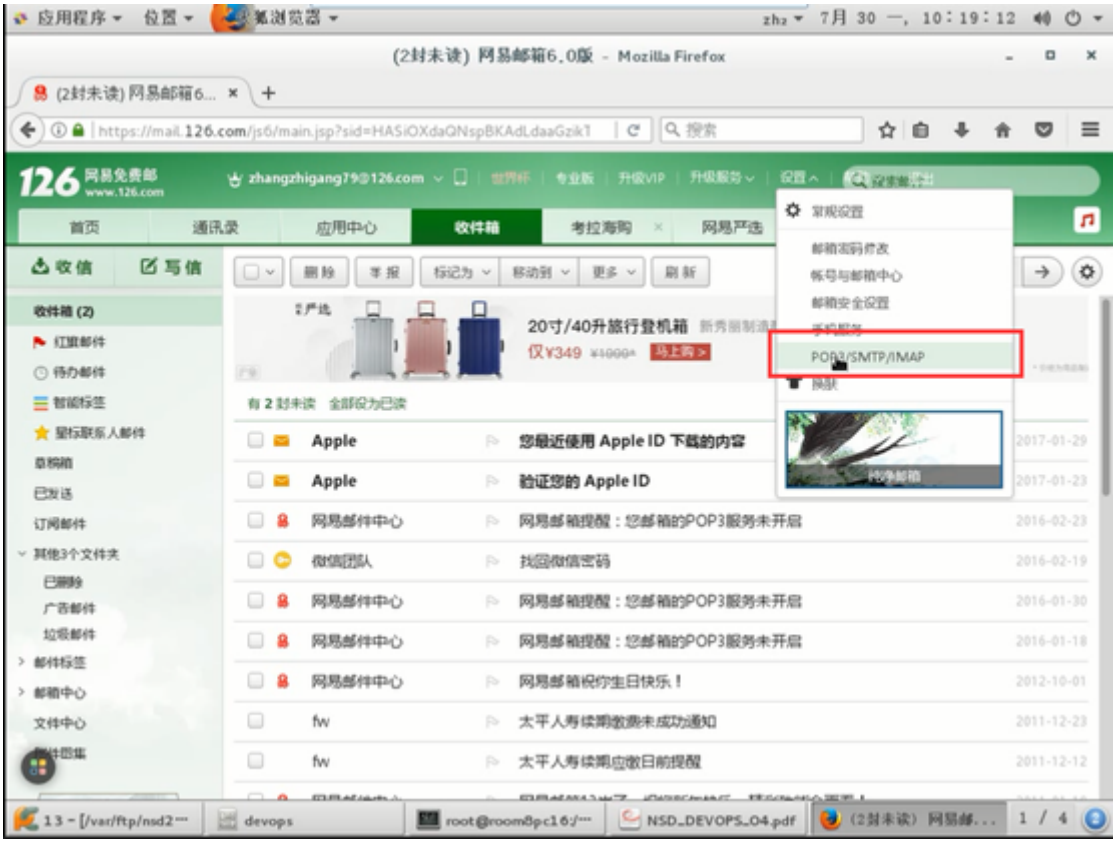

图-2

3. 如图-3所示，上面红框的两个必须勾选上，如没有勾选，要选择开启就可以勾选上了：

Top

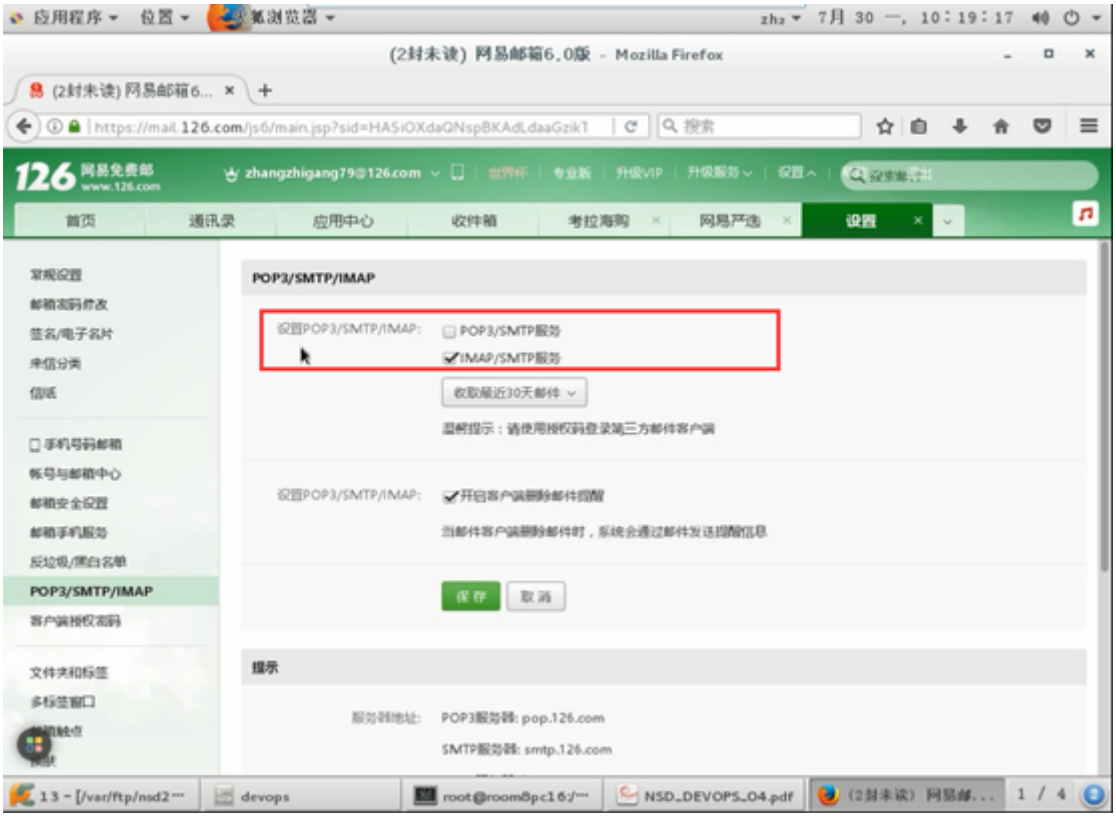

图-3

4.页面向下可以可以看到下图-4红框里是：SMTP服务器是:smtp.126.com：

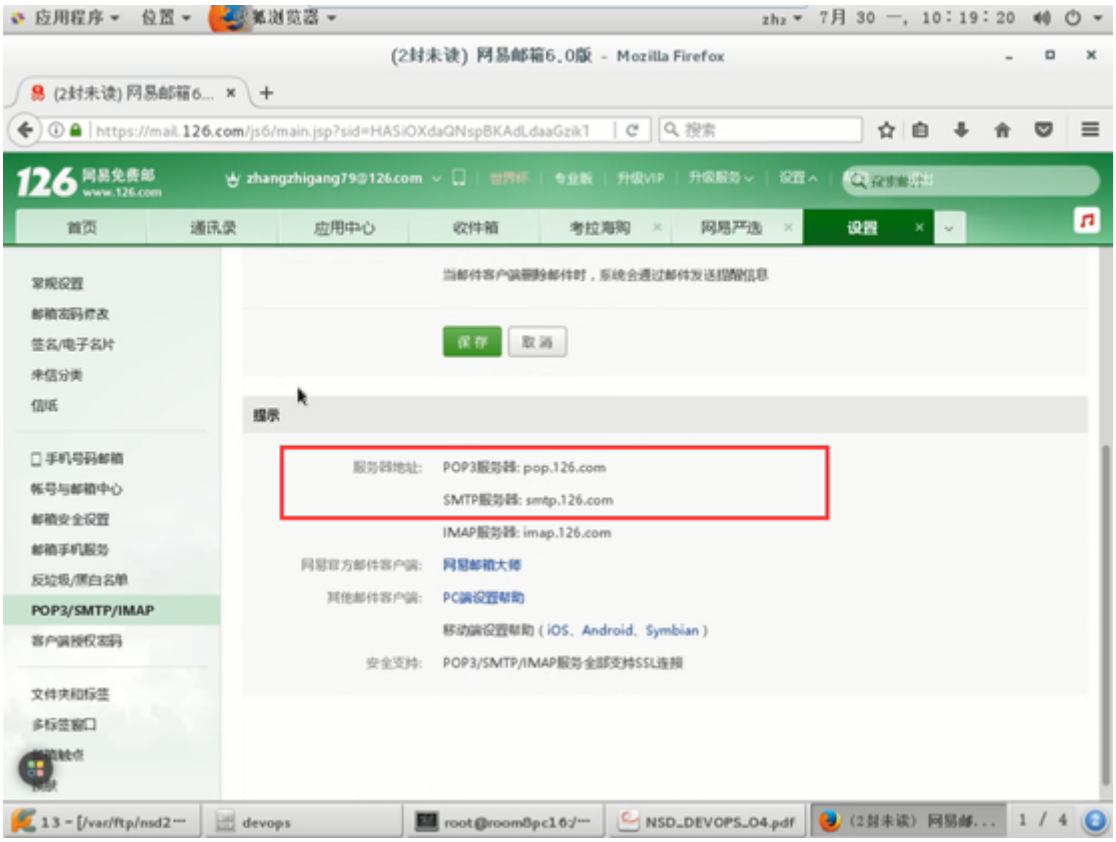

图-4

步骤二：编写脚本

Top

```
01. [root@localhost day12]# vim mail_client.py
```

```
#! /usr/bin/env python3

import smtplib
from getpass import getpass
from email.mime.text import MIMEText
from email.header import Header

mail_host = 'smtp.126.com'        #发件人邮箱账号
mail_user = 'zhangzhigang79@126.com'        #收件人邮箱账号
mail_pwd = getpass()        #获取密码
#邮件正文有三个参数：第一个为文本内容，第二个设置文本格式plain，第三个utf-8设置
message = MIMEText('Python邮件发送测试\n', 'plain', 'utf8')
#发送者信息
message['From'] = Header('zhangzhigang79@126.com', 'utf8')
#接收者信息
message['To'] = Header('zhangzhigang79@126.com', 'utf8')
#主题信息
message['Subject'] = Header('python 1802 mail test', 'utf8')

sender = 'zhangzhigang79@126.com'        #发送方
receivers = ['zhangzhigang79@126.com']        #接收方
smtp_obj = smtplib.SMTP()        #创建SMTP对象
smtp_obj.connect(mail_host)    #将SMTP对象与发送人邮件简历连接建立连接
smtp_obj.login(mail_user, mail_pwd)        #登录用户名密码
# SMTP 对象使用 sendmail 方法发送邮件
smtp_obj.sendmail(sender, receivers, message.as_string())
```

步骤三：测试脚本执行

```
[root@localhost day12]# python3 mail_client.py
Password：
```

如果发送成功，结果显示如图-5所示：

Top

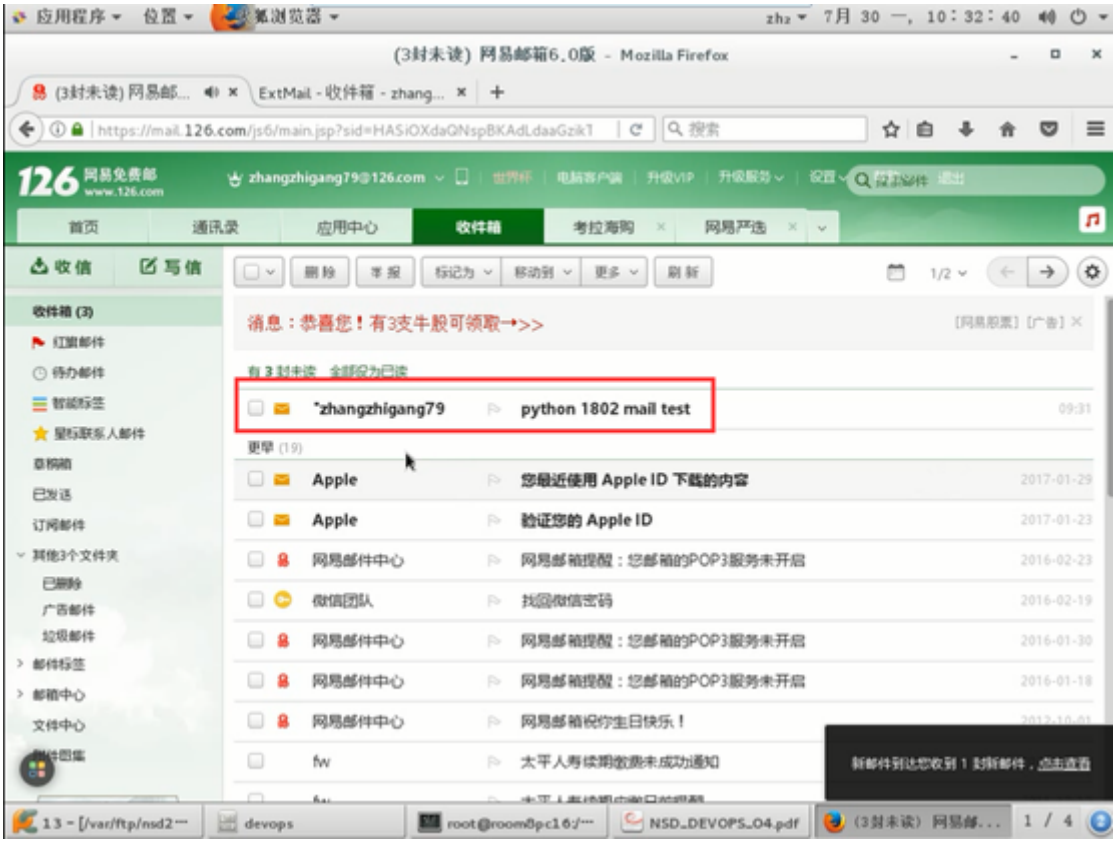

图-5

3 案例3：天气预报查询

3.1 问题

编写一个display_weather.py脚本，实现以下功能：

1. 运行程序时，屏幕将出现你所在城市各区县名字
2. 用户指定查询某区县，屏幕上将出现该区县当前的气温、湿度、风向、风速等

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：找到天气信息规律

1.首先我们想要实现的功能是天气预报，从哪获取天气这是一个问题，在这里可以使用http://www.weather.com.cn/data/sk/101051301.html这个应用程序编程接口，101051301是城市的ID，可以到http://www.weather.com.cn/查看，替换后浏览器打开，如图-6所示，图示为json格式：

图-6

注意：图中看不懂的文字是编码问题

2.从图-1中可以看出cityid就是城市ID，temp是温度，SD是湿度，我们编写代码可以直接获取到网站相应信息，编写脚本wather.py：

```
[root@localhost day12]# vim weather.py
#! /usr/bin/python     #这里是python的目录
from urllib.request import urlopen
import json

#打开网页，使用urllib模块的urlopen函数打开url，赋值给html
html = urlopen('http://www.weather.com.cn/data/sk/101010100.html')
#读html获取的数据，保存到data
data = html.read()
#从data中获取我们想要的信息，json.loads()是将json格式数据转换为字典
#（可以理解为json.loads()函数是将字符串转化为字典）
print(json.loads(data))
#关闭html
html.close()
```

3.执行脚本结果如下：

```
[root@localhost day12]# python3 weather.py
{'weatherinfo':{'city':'北京','cityid':'101010100','temp':'27.9','WD
```

从以上脚本执行结果中我们可以看到，从网站中获取到的数据是以字典形式显示，显示信息有城市、城市id，温度、风向等等，根据这种规律，编写下面代码

步骤二：编写代码实现如下功能

1.定义一个字典，该字典中键‘0’和‘1’对应的值为天气网址中城市对应的id，城市不同id则不同

2.运行程序时，屏幕将出现你所在城市名字

3.当用户指定查询某城市（即输入0或1时）

4.调用get_weather函数，函数的实际参数为city_codes字典对应值（即对应的城市id）

5. 打开天气网页，使用urllib模块的urlopen函数打开url，赋值给html

6.读html获取的数据，用json.loads()获取天气信息，获取到的信息为字典形式

7.从获取到的字典数据中提取气温、湿度、风向、风速等信息，保存在output变量中

8.将output变量作为get_weather函数的返回值，打印在屏幕上

```
[root@localhost day12]# vim display_weather.py
#! /usr/bin/python

from urllib.request import urlopen
```

Top

```
import json

def get_weather(city_code):        #定义一个输入城市id的函数
5.打开天气网页，使用urllib模块的urlopen函数打开url，赋值给html
    url = 'http://www.weather.com.cn/data/sk/%s.html' % city_code
    html = urlopen(url)
6.读html获取的数据，用json.loads()获取我们想要的信息
#json.loads()是将json格式数据转换为字典
#（可以理解为json.loads()函数是将字符串转化为字典）
    data = json.loads(html.read())
7.output为返回值，即最终屏幕显示的信息
    output = '风向：%s, 风力: %s， 温度：%s, 湿度：%s' %(
#data获取到的天气信息为字典，该字典中weatherinfo键对应的值还是一个字典，这个字
        data['weatherinfo']['WD'],
        data['weatherinfo']['WS'],
        data['weatherinfo']['temp'],
        data['weatherinfo']['SD']
    )
    return output


if __name__ == '__main__':
1.定义字典：键对应的值为天气网站网址接口中城市ID
    city_codes = { '0': '1010101100', '1': '1011121404'}
2.代码执行后，屏幕给出的提示信息
    prompt = """(0) 北京
(1) 台儿庄
请选择(0/1): """
3.根据提示信息，输入0或1
    choice = input(prompt)
4.调用get_weather函数，其实际参数为city_codes字典对应值
8.打印调用get_weather函数返回值
    print(get_weather(city_codes[choice]))
```

步骤三：测试脚本执行

Top

```
[root@localhost day12]# python3 display_weather.py
(0) 北京
```

```
(1) 台儿庄
请选择(0/1):0
 风向：南风， 风力： 小于3级， 温度：27.9， 湿度：28%
[root@localhost day12]# python3 display_weather.py
(0) 北京
(1) 台儿庄
请选择(0/1):1
 风向：东北风， 风力： 小于3级， 温度：22.3， 湿度：64%
```

4 案例4：使用requests获取天气

4.1 问题

编写一个display_weather2.py脚本，实现以下功能：

1. 运行程序时，屏幕将出现你所在城市各区县名字
2. 用户指定查询某区县，屏幕上将出现该区县当前的气温、湿度、风向、风速等

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：安装requests

```
[root@localhost ~]# pip3 install requests
```

步骤二：编写代码实现如下功能

1.定义一个字典，该字典中键 ‘0’ 和 ‘1’ 对应的值为天气网址中城市对应的id，城市不同id则不同

2.运行程序时，屏幕将出现你所在城市名字

3.当用户指定查询某城市（即输入0或1时）

4.调用get_weather函数，函数的实际参数为city_codes字典对应值（即对应的城市id）

5.通过requests发送一个GET请求到url网址，获取网页

6. 为获取的网页设置编码，并以json形式返回数据

7.从获取到的字典数据中提取气温、湿度、风向、风速等信息，保存在output变量中

8.将output变量作为get_weather函数的返回值，打印在屏幕上

```
[root@localhost day12]# vim display_weather2.py
#! /usr/bin/python
                                                                    Top
import requests        #引用requests模块

```

```
def get_weather(city_code):        #定义一个输入城市id的函数
    url = 'http://www.weather.com.cn/data/sk/%s.html' %city_code    #天气地址
5.通过requests发送一个GET请求到url网址，获取网页
    r = requests.get(url)
6. 为获取的网页设置编码，并以json形式返回数据
    r.encoding = 'utf8'
#Requests中内置的JSON解码器，以json形式返回,前提返回的内容确保是json格式的，不
    data = r.json()
    7.output为返回值，即最终屏幕显示的信息
    output = '风向：%s, 风力: %s， 温度：%s, 湿度：%s' %(
#data获取到的天气信息为字典，该字典中weatherinfo键对应的值还是一个字典，这个字
        data['weatherinfo']['WD'],
        data['weatherinfo']['WS'],
        data['weatherinfo']['temp'],
        data['weatherinfo']['SD']
    )
    return output

if __name__ == '__main__':
1.定义字典：键对应的值为天气网站网址接口中城市ID
    city_codes = {'0': '1010101100', '1': '1011121404'}
2.代码执行后，屏幕给出的提示信息
    prompt = """(0) 北京
(1) 台儿庄
请选择(0/1): """
3.根据提示信息，输入0或1
    choice = input(prompt)
4.调用get_weather函数，其实际参数为city_codes字典对应值
8.打印调用get_weather函数返回值
    print(get_weather(city_codes[choice]))
```

步骤三：测试脚本执行

```
[root@localhost day12]# python3 display_weather2.py
(0) 北京
(1) 台儿庄
请选择(0/1): 0
 风向：南风， 风力： 小于3级， 温度：27.9， 湿度：28%
[root@localhost day12]# python3 display_weather2.py
```

Top

```
07.     (0) 北京
08.     (1) 台儿庄
09.     请选择(0/1)：1
10.      风向：东北风， 风力： 小于3级， 温度：22.3， 湿度：64%
```

5 案例5：获取zabbix版本信息

5.1 问题

编写一个zabbix_apiversion.py脚本，实现以下功能：

1. 安装zabbix服务器
2. 获取zabbix api的url
3. 编写python程序，访问zabbix api，取得zabbix版本号

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：安装zabbix服务器

安装zabbix服务器具体步骤请参照NSD SECURITY DAY06 cookbook中案例2操作

步骤二：获取zabbix api的url

1)设置前端后，你就可以使用远程HTTP请求来调用API。为此，需要向api_jsonrpc.php位于前端目录中的文件发送HTTP POST请求。如果你的Zabbix前端安装在http://192.168.4.2/zabbix，那么用HTTP请求来调用apiinfo.version方法就如下面这样：

```
01.     POST 192.168.4.2/zabbix/api_jsonrpc.php HTTP/1.1
```

2)从zabbix官方文档中获取 Zabbix API 版本，如图-7所示：

官方文档地址如下：

https://www.zabbix.com/documentation/3.4/zh/manual/api/reference/apiinfo/version

Top

图-7

步骤三：编写脚本

在HTTP协议中，post提交的数据必须放在消息主体中，但是协议中并没有规定必须使用什么编码方式，从而导致了提交方式的不同。服务端根据请求头中的Content-Type字段来获知请求中的消息主体是用何种方式进行编码，再对消息主体进行解析。

请求的 Content-Type 头部必须设置为以下值之一：

application/json-rpc

application/json

application/jsonrequest

```
[root@localhost day12]# vim zabbix_apiversion.py
#! /usr/bin/env python3

import requests
import json    # python中的dict类型要转换为json格式的数据需要用到json库

#要访问的网址
url = 'http://192.168.4.2/zabbix/api_jsonrpc.php'
#请求头部信息
headers = {'Content-Type': 'application/json-rpc'}
# data是从官方文档处获得的
data = {
    "jsonrpc": "2.0",    #sonrpc协议的版本号，固定的
    "method": "apiinfo.version",        #在zabbix手册上查到的，查询zabbix版本
```

Top

```
15.         "params": [],        #没有额外参数
16.         "id": 1   #随便写个数字
17.     }
18.
19.     #使用requests发送请求，访问指定网站，向url发送data请求，r收到的返回响应为json格
20.     #将data转成json格式，zabbix要求提交的数据是json格式
21.     r = requests.post(url, headers=headers, data=json.dumps(data))
22.     #将json格式解码，zabbix返回的数据都是json格式
23.     print(r.json())
```

需要注意的是python中并没有json类型这一说法，通过json.dumps(data)转换的字典对象，最后得到的是一个字符串对象，也就是说，在python中json格式的数据实际上就是一个字符串

步骤四：测试脚本执行

```
01.     [root@localhost day12]# python3 zabbix_apiversion.py
02.     {"jsonrpc": "2.0", "result": "2.4.0", "id": 1}
```

6 案例6：获取令牌

6.1 问题

编写一个get_token.py脚本，实现以下功能：

1. 编写get_token函数
2. 该函数接受zabbix服务器url、用户名和密码作为参数
3. 函数返回值为用户令牌token

6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本get_token.py，获取身份令牌

1)从zabbix官方文档中使用 user.login 方法登录并获取身份验证令牌请求，如图-8所示：

官方文档地址如下：

https://www.zabbix.com/documentation/3.4/zh/manual/api/reference/user/login

Top

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

图-8

2)编写get_token.py文件

```
[root@localhost day12]# vim get_token.py
#! /usr/bin/env python3

import requests
import json    # python中的dict类型要转换为json格式的数据需要用到json库

#要访问的网址
url = 'http://192.168.4.2/zabbix/api_jsonrpc.php'
#请求头部信息
headers = {'Content-Type': 'application/json-rpc'}
# data是从官方文档处获得的
data = {
# API使用的JSON-RPC协议的版本; Zabbix API实现JSON-RPC版本2.0
    "jsonrpc": "2.0",
    "method": "user.login",    #调用的API方法
# params将被传递给API方法的参数
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1    #请求的任意标识符
}
```

Top

```
#使用requests发送请求，访问指定网站，向url发送data请求，r收到的返回响应为json格
#将data转成json格式
r = requests.post( url, headers=headers, data=json.dumps( data) )
#将json格式解码
print( r.json( ) ) )
```

步骤四：测试脚本执行，获取用户令牌token

```
[ root@localhost day 12] # python3 get_token.py
{ "jsonrpc": "2.0", "result": "0424bd59b807674191e7d77572075f 33", "id": 1}
```

7 案例7：创建主机

7.1 问题

编写一个remote_comm.py脚本，实现以下功能：

1. 主机192.168.4.10已安装zabbix_agent
2. 将该主机填加到zabbix监控的主机中
3. 主机属于Linux Servers组

7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本，检索组

1)从zabbix官方文档中使用hostgroup.get 方法获取主机组请求，如图-9所示：

官方文档地址如下：

https://www.zabbix.com/documentation/3.4/zh/manual/api/reference/hostgroup/get

Top

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Zabbix servers",
                "Linux servers"
            ]
        }
    },
    "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
    "id": 1
}
```

图-9

2)编写get_hostgroup.py文件，检索组

```
[root@localhost day12]# vim get_hostgroup.py
#! /usr/bin/env python3

import requests
import json    # python中的dict类型要转换为json格式的数据需要用到json库

#要访问的网址
url = 'http://192.168.4.2/zabbix/api_jsonrpc.php'
#请求头部信息
headers = {'Content-Type': 'application/json-rpc'}
# data是从官方文档处获得的
data = {
# API使用的JSON-RPC协议的版本；Zabbix API实现JSON-RPC版本2.0
   "jsonrpc": "2.0",
   "method": " hostgroup.get ",    #调用的API方法
# params将被传递给API方法的参数
    "params": {
       "output": "extend",
    },
    "auth": "0424bd59b807674191e7d77572075f33",    #之前获取到的令牌
   "id": 1   #请求的任意标识符
}
#使用requests发送请求，访问指定网站，向url发送data请求，r收到的返回响应为json格
```

Top

```
24.     #将data转成json格式
25.     r = requests.post( url, headers=headers, data=json.dumps( data) )
26.     ginfo = r.json( )
27.     print( ginfo[ 'result'] )     #打印主机组信息
28.     for item in ginfo[ 'result'] :
29.         print( item[ 'groupid'] , item[ 'name'] )     #打印主机组id和名称
```

3)测试脚本执行，获取了主机名称及id，如图-10所示：

```
plications', 'internal': '0', 'flags': '0'}, {'groupid': '1
3', 'name': 'Templates/Databases', 'internal': '0', 'flags'
: '0'}, {'groupid': '14', 'name': 'Templates/Virtualization
', 'internal': '0', 'flags': '0'}]
1 Templates
2 Linux servers
4 Zabbix servers
5 Discovered hosts
6 Virtual machines
7 Hypervisors
8 Templates/Modules
9 Templates/Network Devices
10 Templates/Operating Systems
11 Templates/Servers Hardware
12 Templates/Applications
13 Templates/Databases
14 Templates/Virtualization
```

图-10

步骤二：编写脚本，检索模板

1)从zabbix官方文档中使用template.get 方法获取模板请求，如图-11所示：

官方文档地址如下：

https://www.zabbix.com/documentation/3.4/zh/manual/api/reference/template/get

Request请求:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Template OS Linux",
                "Template OS Windows"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Top

图-11

2)编写get_template.py文件

```
[root@localhost day12]# vim get_template.py
#! /usr/bin/env python3

import requests
import json    # python中的dict类型要转换为json格式的数据需要用到json库

#要访问的网址
url = 'http://192.168.4.2/zabbix/api_jsonrpc.php'
#请求头部信息
headers = {'Content-Type': 'application/json-rpc'}
# data是从官方文档处获得的
data = {
# API使用的JSON-RPC协议的版本; Zabbix API实现JSON-RPC版本2.0
    "jsonrpc": "2.0",
    "method": "template.get",    #调用的API方法
# params将被传递给API方法的参数
     "params": {
       "output": "extend",
     },
     "auth": "0424bd59b807674191e7d77572075f33",    #之前获取到的令牌
    "id": 1   #请求的任意标识符
}
#使用requests发送请求，访问指定网站，向url发送data请求，r收到的返回响应为json格
#将data转成json格式
r = requests.post(url, headers=headers, data=json.dumps(data))
#将json格式解码
tinfo = r.json()
#print(tinfo)        #打印模板信息
for item in tinfo['result']:
    print(item['templateid'], item['host'])        #打印模板名及id
```

3)测试脚本执行，获取了模板名称及id，如图-12所示：

Top

```
10001 Template OS Linux
10047 Template App Zabbix Server
10048 Template App Zabbix Proxy
10050 Template App Zabbix Agent
10074 Template OS OpenBSD
10075 Template OS FreeBSD
10076 Template OS AIX
10077 Template OS HP-UX
10078 Template OS Solaris
10079 Template OS Mac OS X
10081 Template OS Windows
10093 Template App FTP Service
10094 Template App HTTP Service
```

图-12

步骤三：编写脚本，创建主机

1)从zabbix官方文档中使用host.create方法获取创建主机请求，如图-13所示：

官方文档地址如下：

https://www.zabbix.com/documentation/3.4/zh/manual/api/reference/host/create

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "50"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

图-13

2)编写get_host.py文件 Top

```
[root@localhost day12]# vim get_host.py
#! /usr/bin/env python3

import requests
import json    # python中的dict类型要转换为json格式的数据需要用到json库

#要访问的网址
url = 'http://192.168.4.2/zabbix/api_jsonrpc.php'
#请求头部信息
headers = {'Content-Type': 'application/json-rpc'}
# data是从官方文档处获得的
data = {
# API使用的JSON-RPC协议的版本; Zabbix API实现JSON-RPC版本2.0
   "jsonrpc": "2.0",
   "method": "user.login",    #调用的API方法
# params将被传递给API方法的参数
   "params": {
      "host": "mylinux",        #要创建的主机的名称
      "interfaces": [
         {
            "type": 1,  # 1 agent; 2 SNMP; 3 IPMI; 4 JMX
            "main": 1,  # 该接口是否在主机上用作默认接口。1默认
            "useip": 1,  # 是否应通过IP进行连接
            "ip": "192.168.4.3",
            "dns": "",
            "port": "10050"        #ip已装zabbix_agent，端口号为10050
         }
      ],
      "groups": [
         {
            "groupid": "2"        #之前检索到的主机组id
         }
      ],
      "templates": [
         {
            "templateid": "10001"        #之前检索到的模板id
         }
      ],
      "inventory_mode": 0,        #资产信息，0为停用
   },
```

Top

```
        "auth": "0424bd59b807674191e7d77572075f33",      #之前获取到的令牌
        "id": 1   #请求的任意标识符
    }
    #使用requests发送请求，访问指定网站，向url发送data请求，r收到的返回响应为json格
    #将data转成json格式
    r = requests.post( url, headers=headers, data=json.dumps( data) )
```

3)测试脚本执行，运行脚本成功后，主机创建成功，如图-14所示：

图-14

Top