PYTHON兼容IPV6 的编码技巧

张龙

个人技术栈

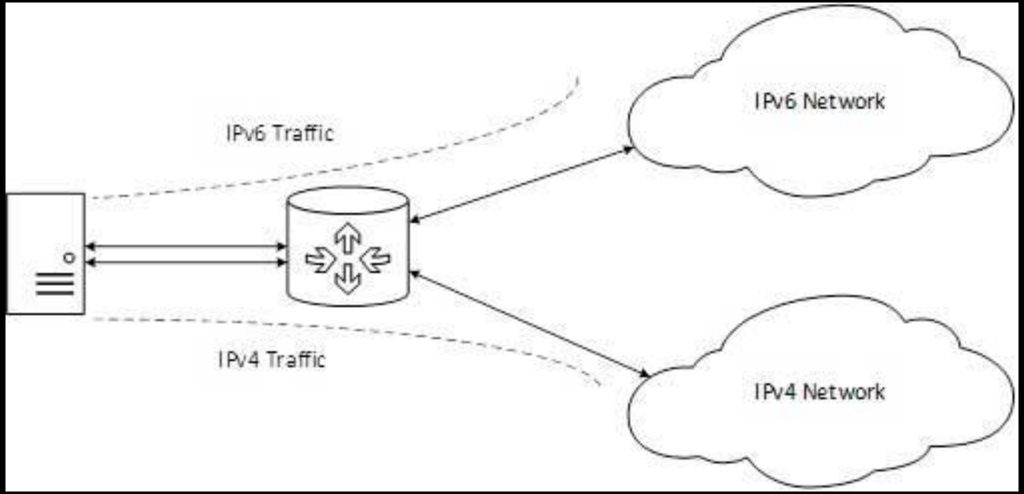
- 学校用C/C++
- 刚工作用java等
- 换工作用python等
- 学得快忘得快
- 先从事安全即服务，后退化到传统扫描盒子，又切换回云安全
- 熟悉应用层编程&调试，OS和虚拟化较弱，数通小白
- 编码有洁癖；觉得开发的能力需要用代码来说话

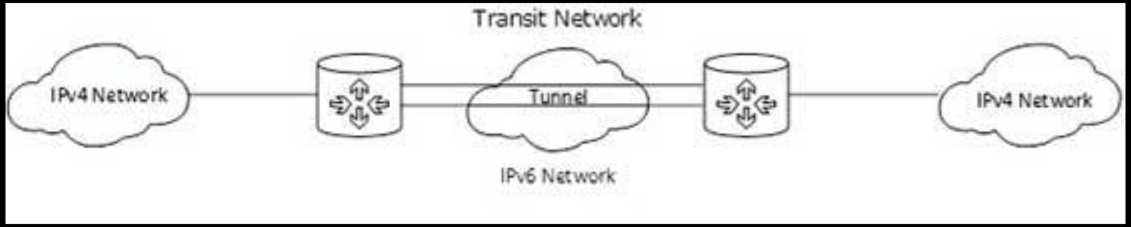
Agenda

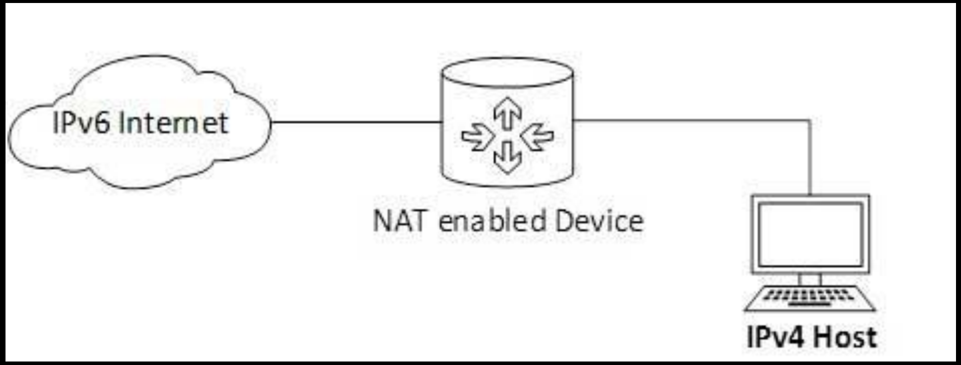
1. 背景介绍
2. 目标&原则
3. 对象表示
4. 方法使用

背景

- IPv6不向后兼容
- 从IPv4到IPv6需要平滑的过渡
  1. 双栈路由器
  2. **隧道**
  3. **NAT协议翻译**
- 过渡期后， 需要完全支持IPV6







目标&原则

目标

✓在业务层面不区分IPV4/6，能支持双栈

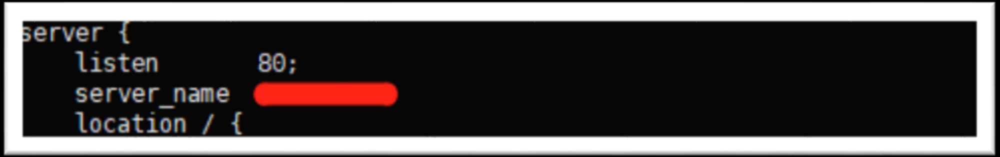
✓并不包含IPV6协议本身的内容

原则

- 接口设计兼容
- IP对象类兼容
- 避免使用不兼容的工具
  - ping vs ping6
- 非得使用请做兼容性封装

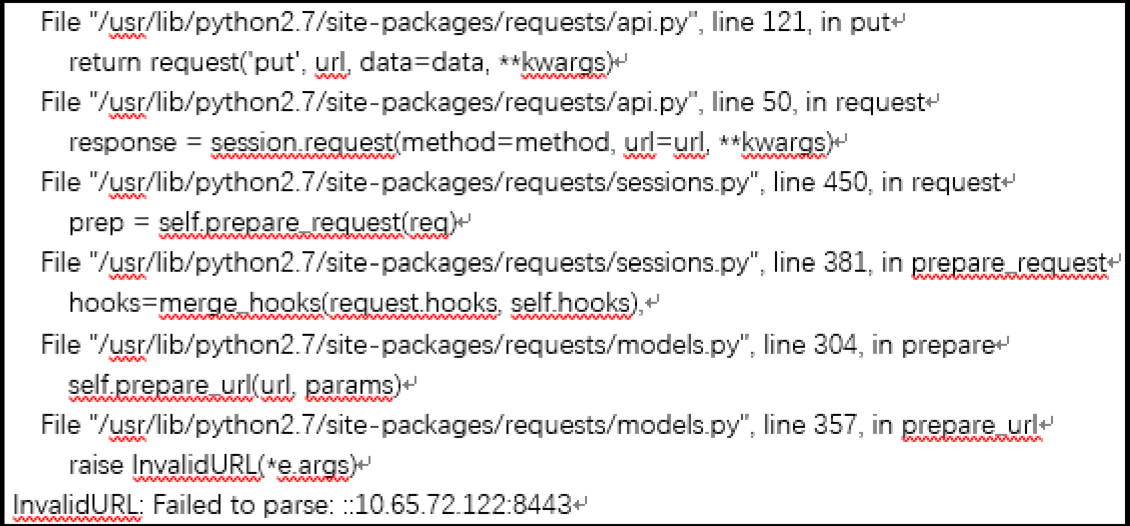
已知问题

- Nginx默认只接收IPv4的连接请求：

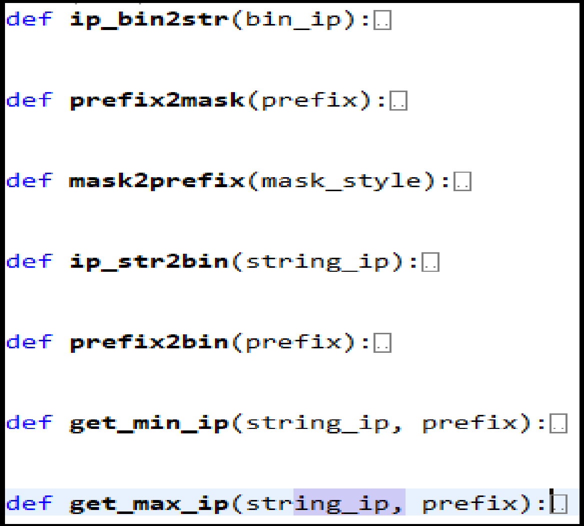


```
server {
    listen       80;
    server_name [redacted]
    location / {
```

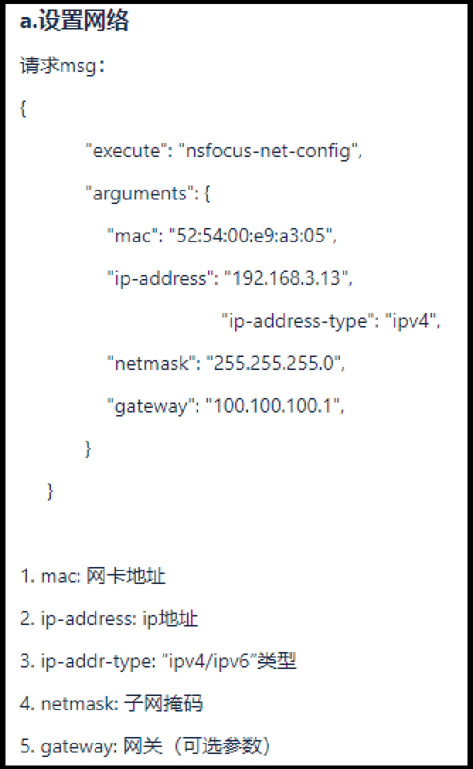
- IP的配置，存储和转换，网络请求等操作



```
  File "/usr/lib/python2.7/site-packages/requests/api.py", line 121, in put
    return request('put', url, data=data, **kwargs)
  File "/usr/lib/python2.7/site-packages/requests/api.py", line 50, in request
    response = session.request(method=method, url=url, **kwargs)
  File "/usr/lib/python2.7/site-packages/requests/sessions.py", line 450, in request
    prep = self.prepare_request(req)
  File "/usr/lib/python2.7/site-packages/requests/sessions.py", line 381, in prepare_request
    hooks=merge_hooks(request.hooks, self.hooks),
  File "/usr/lib/python2.7/site-packages/requests/models.py", line 304, in prepare
    self.prepare_url(url, params)
  File "/usr/lib/python2.7/site-packages/requests/models.py", line 357, in prepare_url
    raise InvalidURL(*e.args)
InvalidURL: Failed to parse: ::10.65.72.122:8443
```



```
def ip_bin2str(bin_ip):

def prefix2mask(prefix):

def mask2prefix(mask_style):

def ip_str2bin(string_ip):

def prefix2bin(prefix):

def get_min_ip(string_ip, prefix):

def get_max_ip(string_ip, prefix):
```

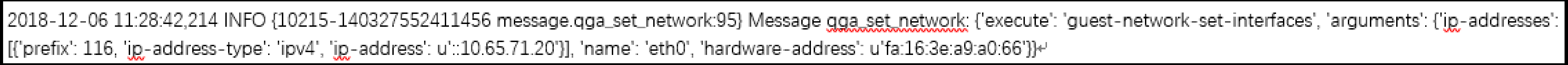


a.设置网络

请求msg：

```
{
        "execute": "nsfocus-net-config",
        "arguments": {
            "mac": "52:54:00:e9:a3:05",
            "ip-address": "192.168.3.13",
                            "ip-address-type": "ipv4",
            "netmask": "255.255.255.0",
            "gateway": "100.100.100.1",
        }
    }
```

1. mac: 网卡地址
2. ip-address: ip地址
3. ip-addr-type: "ipv4/ipv6"类型
4. netmask: 子网掩码
5. gateway: 网关（可选参数）



```
2018-12-06 11:28:42,214 INFO {10215-140327552411456 message.qga_set_network:95} Message qga_set_network: {'execute': 'guest-network-set-interfaces', 'arguments': {'ip-addresses': [{'prefix': 116, 'ip-address-type': 'ipv4', 'ip-address': u'::10.65.71.20'}], 'name': 'eth0', 'hardware-address': u'fa:16:3e:a9:a0:66'}}
```

对象表示

0010000000000001 0000000000000000 0011001000111000 1101111111100001 0000000001100011 0000000000000000 0000000000000000 1111111011111011

2001:0000:3238:DFE1:0063:0000:0000:FEFB

字符串

- 丢弃前导零
- 如果四个数字都是零，可以被省略
- 如果因为省略而出现了两个以上的冒号的话，可以压缩为一个
- 但这种零压缩在地址中只能出现一次
- 如果这个地址实际上是IPv4的地址，后32位可以用10进制数表示

常见格式：2001:0:3238:DFE1:63::FEFB （Global address，begin with 001）
IPv4映像地址：ffff:1.2.3.4（不建议使用）
IPv4一致地址：::1.2.3.4

敲黑板：

128bit，8组16进制，字符串最长**39**char
32bit，4组10进制，字符串最长**15**char

对象表示

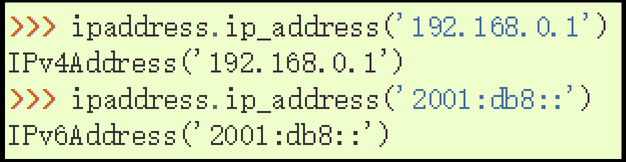
整型

- IPv4，小于2 ** 32的整数
- IPv6，小于2 ** 128的整数

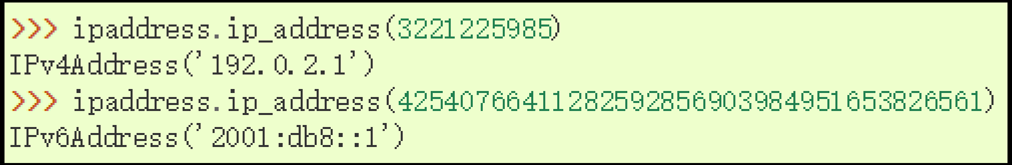
对象表示

Python的ipaddress模块提供了工厂功能，可以方便地创建：

- IP地址:ipaddress.ipaddress(**address**)

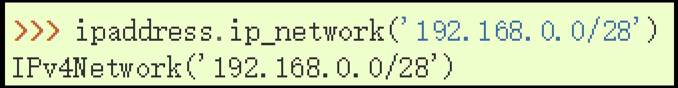


```
>>> ipaddress.ip_address('192.168.0.1')
IPv4Address('192.168.0.1')
>>> ipaddress.ip_address('2001:db8::')
IPv6Address('2001:db8::')
```



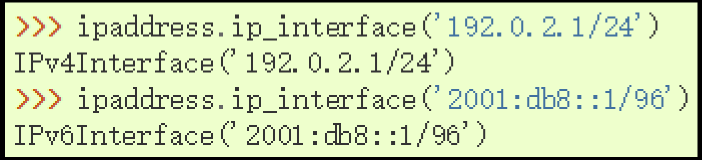
```
>>> ipaddress.ip_address(3221225985)
IPv4Address('192.0.2.1')
>>> ipaddress.ip_address(42540766411282592856903984951653826561)
IPv6Address('2001:db8::1')
```

- 网络: ipaddress.ip_network(address, strict=True)



```
>>> ipaddress.ip_network('192.168.0.0/28')
IPv4Network('192.168.0.0/28')
```

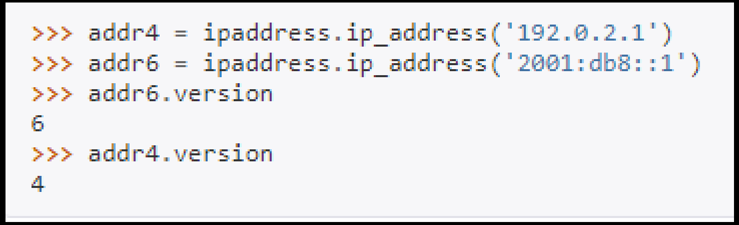
- 接口: ipaddress.ip_interface(address)



```
>>> ipaddress.ip_interface('192.0.2.1/24')
IPv4Interface('192.0.2.1/24')
>>> ipaddress.ip_interface('2001:db8::1/96')
IPv6Interface('2001:db8::1/96')
```

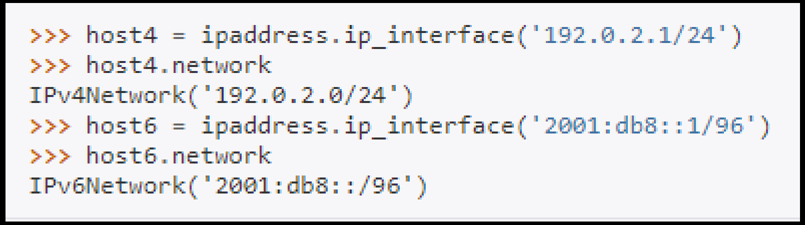
方法使用

1. 判断IP版本



```
>>> addr4 = ipaddress.ip_address('192.0.2.1')
>>> addr6 = ipaddress.ip_address('2001:db8::1')
>>> addr6.version
6
>>> addr4.version
4
```

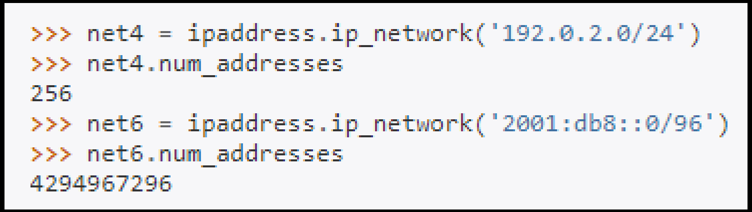
2. 从接口获取网络



```
>>> host4 = ipaddress.ip_interface('192.0.2.1/24')
>>> host4.network
IPv4Network('192.0.2.0/24')
>>> host6 = ipaddress.ip_interface('2001:db8::1/96')
>>> host6.network
IPv6Network('2001:db8::/96')
```

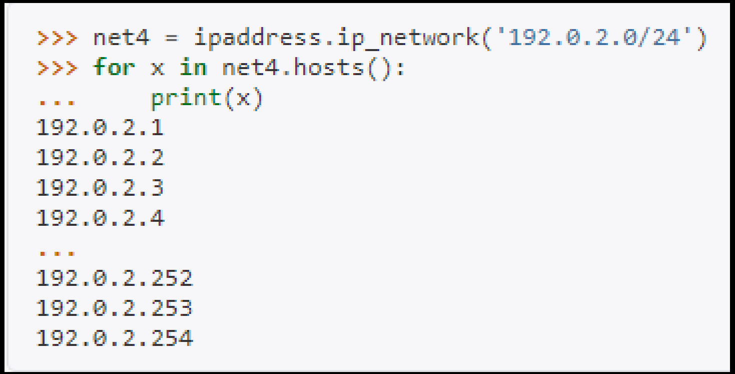
方法使用

3. 找出网段中有多少个独立的IP地址



```
>>> net4 = ipaddress.ip_network('192.0.2.0/24')
>>> net4.num_addresses
256
>>> net6 = ipaddress.ip_network('2001:db8::0/96')
>>> net6.num_addresses
4294967296
```

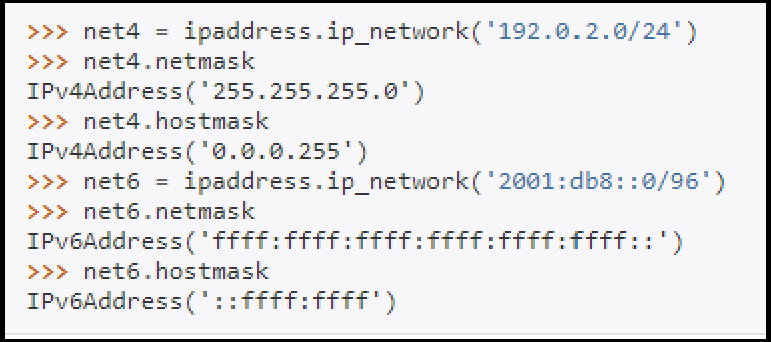
4. 获取网络上的“可用”IP地址



```
>>> net4 = ipaddress.ip_network('192.0.2.0/24')
>>> for x in net4.hosts():
...     print(x)
192.0.2.1
192.0.2.2
192.0.2.3
192.0.2.4
...
192.0.2.252
192.0.2.253
192.0.2.254
```

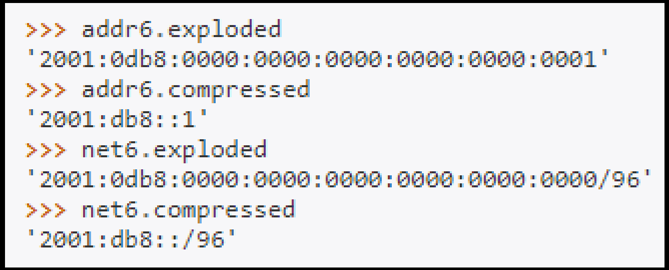
方法使用

5. 获取网络掩码和主机掩码



```
>>> net4 = ipaddress.ip_network('192.0.2.0/24')
>>> net4.netmask
IPv4Address('255.255.255.0')
>>> net4.hostmask
IPv4Address('0.0.0.255')
>>> net6 = ipaddress.ip_network('2001:db8::0/96')
>>> net6.netmask
IPv6Address('ffff:ffff:ffff:ffff:ffff:ffff::')
>>> net6.hostmask
IPv6Address('::ffff:ffff')
```

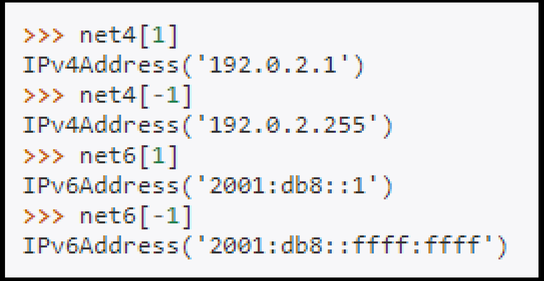
6. 展开或压缩IP地址



```
>>> addr6.exploded
'2001:0db8:0000:0000:0000:0000:0000:0001'
>>> addr6.compressed
'2001:db8::1'
>>> net6.exploded
'2001:0db8:0000:0000:0000:0000:0000:0000/96'
>>> net6.compressed
'2001:db8::/96'
```

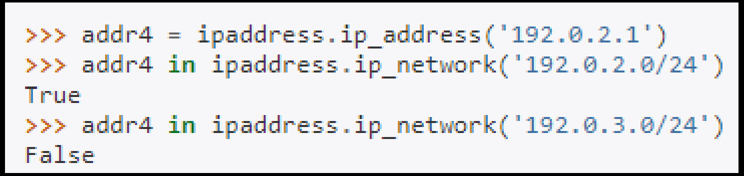
方法使用

7. 获取网络里的第N个IP地址



```
>>> net4[1]
IPv4Address('192.0.2.1')
>>> net4[-1]
IPv4Address('192.0.2.255')
>>> net6[1]
IPv6Address('2001:db8::1')
>>> net6[-1]
IPv6Address('2001:db8::ffff:ffff')
```

8. 判断IP是否在网络里面



```
>>> addr4 = ipaddress.ip_address('192.0.2.1')
>>> addr4 in ipaddress.ip_network('192.0.2.0/24')
True
>>> addr4 in ipaddress.ip_network('192.0.3.0/24')
False
```

方法使用

9. 同类型IP比较

```
>>> ipaddress.ip_address('192.0.2.1') < ipaddress.ip_address('192.0.2.2')
True
```

10. 类型转换

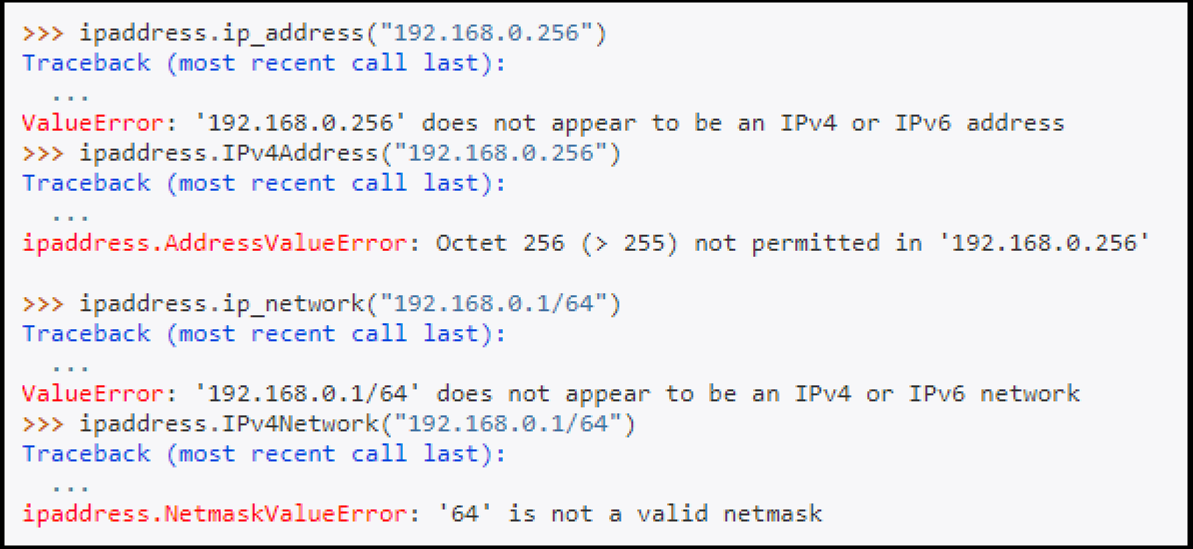
```
>>> addr4 = ipaddress.ip_address('192.0.2.1')
>>> str(addr4)
'192.0.2.1'
>>> int(addr4)
3221225985
```

方法使用

11. 异常捕获

valueError 子类：

- ipaddress.AddressValueError
- ipaddress.NetmaskValueError



```
>>> ipaddress.ip_address("192.168.0.256")
Traceback (most recent call last):
  ...
ValueError: '192.168.0.256' does not appear to be an IPv4 or IPv6 address
>>> ipaddress.IPv4Address("192.168.0.256")
Traceback (most recent call last):
  ...
ipaddress.AddressValueError: Octet 256 (> 255) not permitted in '192.168.0.256'

>>> ipaddress.ip_network("192.168.0.1/64")
Traceback (most recent call last):
  ...
ValueError: '192.168.0.1/64' does not appear to be an IPv4 or IPv6 network
>>> ipaddress.IPv4Network("192.168.0.1/64")
Traceback (most recent call last):
  ...
ipaddress.NetmaskValueError: '64' is not a valid netmask
```

方法使用

12. 拼接URL
    a. 使用域名（推荐）
    b. 封装类/函数

Q&A