11.4 通过CIDR地址生成对应的IP地址集¶

问题¶

你有一个CIDR网络地址比如“123.45.67.89/27”，你想将其转换成它所代表的所有IP （比如，“123.45.67.64”, “123.45.67.65”, ..., “123.45.67.95”））

解决方案¶

可以使用 `ipaddress` 模块很容易的实现这样的计算。例如：

```
>>> import ipaddress
>>> net = ipaddress.ip_network('123.45.67.64/27')
>>> net
IPv4Network('123.45.67.64/27')
>>> for a in net:
...     print(a)
...
123.45.67.64
123.45.67.65
123.45.67.66
123.45.67.67
123.45.67.68
...
123.45.67.95
>>>

>>> net6 = ipaddress.ip_network('12:3456:78:90ab:cd:ef01:23:30/125')
>>> net6
IPv6Network('12:3456:78:90ab:cd:ef01:23:30/125')
>>> for a in net6:
...     print(a)
...
12:3456:78:90ab:cd:ef01:23:30
12:3456:78:90ab:cd:ef01:23:31
12:3456:78:90ab:cd:ef01:23:32
12:3456:78:90ab:cd:ef01:23:33
12:3456:78:90ab:cd:ef01:23:34
12:3456:78:90ab:cd:ef01:23:35
12:3456:78:90ab:cd:ef01:23:36
12:3456:78:90ab:cd:ef01:23:37
>>>
```

`Network` 也允许像数组一样的索引取值，例如：

```
>>> net.num_addresses
32
>>> net[0]

IPv4Address('123.45.67.64')
>>> net[1]
IPv4Address('123.45.67.65')
>>> net[-1]
IPv4Address('123.45.67.95')
>>> net[-2]
```

```
IPv4Address('123.45.67.94')
>>>
```

另外，你还可以执行网络成员检查之类的操作：

```
>>> a = ipaddress.ip_address('123.45.67.69')
>>> a in net
True
>>> b = ipaddress.ip_address('123.45.67.123')
>>> b in net
False
>>>
```

一个IP地址和网络地址能通过一个IP接口来指定，例如：

```
>>> inet = ipaddress.ip_interface('123.45.67.73/27')
>>> inet.network
IPv4Network('123.45.67.64/27')
>>> inet.ip
IPv4Address('123.45.67.73')
>>>
```

讨论¶

`ipaddress` 模块有很多类可以表示IP地址、网络和接口。 当你需要操作网络地址（比如解析、打印、验证等）的时候会很有用。

要注意的是， `ipaddress` 模块跟其他一些和网络相关的模块比如 `socket` 库交集很少。 所以，你不能使用 `IPv4Address` 的实例来代替一个地址字符串，你首先得显式的使用 `str()` 转换它。例如：

```
>>> a = ipaddress.ip_address('127.0.0.1')
>>> from socket import socket, AF_INET, SOCK_STREAM
>>> s = socket(AF_INET, SOCK_STREAM)
>>> s.connect((a, 8080))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'IPv4Address' object to str implicitly
>>> s.connect((str(a), 8080))
>>>
```

更多相关内容，请参考 An Introduction to the ipaddress Module