3.5 字节到大整数的打包与解包¶

问题¶

你有一个字节字符串并想将它解压成一个整数。或者，你需要将一个大整数转换为一个字节字符串。

解决方案¶

假设你的程序需要处理一个拥有128位长的16个元素的字节字符串。比如：

```
data = b'\x00\x124V\x00x\x90\xab\x00\xcd\xef\x01\x00#\x004'
```

为了将bytes解析为整数，使用 `int.from_bytes()` 方法，并像下面这样指定字节顺序：

```
>>> len(data)
16
>>> int.from_bytes(data, 'little')
69120565665751139577663547927094891008
>>> int.from_bytes(data, 'big')
94522842520747284487117727783387188
>>>
```

为了将一个大整数转换为一个字节字符串，使用 `int.to_bytes()` 方法，并像下面这样指定字节数和字节顺序：

```
>>> x = 94522842520747284487117727783387188
>>> x.to_bytes(16, 'big')
b'\x00\x124V\x00x\x90\xab\x00\xcd\xef\x01\x00#\x004'
>>> x.to_bytes(16, 'little')
b'4\x00#\x00\x01\xef\xcd\x00\xab\x90x\x00V4\x12\x00'
>>>
```

讨论¶

大整数和字节字符串之间的转换操作并不常见。 然而，在一些应用领域有时候也会出现，比如密码学或者网络。 例如，IPv6网络地址使用一个128位的整数表示。 如果你要从一个数据记录中提取这样的值的时候，你就会面对这样的问题。

作为一种替代方案，你可能想使用6.11小节中所介绍的 `struct` 模块来解压字节。 这样也行得通，不过利用 `struct` 模块来解压对于整数的大小是有限制的。 因此，你可能想解压多个字节串并将结果合并为最终的结果，就像下面这样：

```
>>> data
b'\x00\x124V\x00x\x90\xab\x00\xcd\xef\x01\x00#\x004'
>>> import struct
>>> hi, lo = struct.unpack('>QQ', data)
>>> (hi << 64) + lo
94522842520747284487117727783387188
>>>
```

字节顺序规则(little或big)仅仅指定了构建整数时的字节的低位高位排列方式。 我们从下面精心构造的16进制数的表示中可以很容易的看出来：

```
>>> x = 0x01020304
```

```
>>> x.to_bytes(4, 'big')
b'\x01\x02\x03\x04'
>>> x.to_bytes(4, 'little')
b'\x04\x03\x02\x01'
>>>
```

如果你试着将一个整数打包为字节字符串，那么它就不合适了，你会得到一个错误。 如果需要的话，你可以使用 `int.bit_length()` 方法来决定需要多少字节位来存储这个值。

```
>>> x = 523 ** 23
>>> x
335381300113661875107536852714019056160355655333978849017944067
>>> x.to_bytes(16, 'little')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
OverflowError: int too big to convert
>>> x.bit_length()
208
>>> nbytes, rem = divmod(x.bit_length(), 8)
>>> if rem:
... nbytes += 1
...
>>>
>>> x.to_bytes(nbytes, 'little')
b'\x03X\xf1\x82iT\x96\xac\xc7c\x16\xf3\xb9\xcf...\xd0'
>>>
```