A quick way to parse this message would be to simply assign 1 byte to the first two attributes and 2 bytes to the next three attributes:

```
class ICMP:
    def __init__(self, buff):
        header = struct.unpack('<BBHHH', buff)
        self.type = header[0]
        self.code = header[1]
        self.sum = header[2]
        self.id = header[3]
        self.seq = header[4]
```

Read the `struct` documentation (*https://docs.python.org/3/library/struct.html*) for full details about using this module.

You can use either the `ctypes` module or the `struct` module to read and parse binary data. No matter which approach you take, you'll instantiate the class like this:

```
    mypacket = IP(buff)
    print(f'{mypacket.src_address} -> {mypacket.dst_address}')
```

In this example, you instantiate the IP class with your packet data in the variable `buff`.

Writing the IP Decoder

Let's implement the IP decoding routine we just created into a file called *sniffer_ip_header_decode.py*, as shown here:

```
import ipaddress
import os
import socket
import struct
import sys

❶ class IP:
    def __init__(self, buff=None):
        header = struct.unpack('<BBHHHBBH4s4s', buff)
        self.ver = header[0] >> 4
        self.ihl = header[0] & 0xF

        self.tos = header[1]
        self.len = header[2]
        self.id = header[3]
        self.offset = header[4]
        self.ttl = header[5]
        self.protocol_num = header[6]
        self.sum = header[7]
        self.src = header[8]
        self.dst = header[9]
```