struct package

This module performs conversions between Python values and C structs represented as Python bytes objects. This can be used in handling binary data from network connections.

In [1]:
```
import struct
```

struct.pack(fmt, v1, v2, ...)

Return a bytes object containing the values v1, v2, … packed according to the format string fmt. The arguments must match the values required by the format exactly.

- "!" means network endianess (big endian)
- "<" means little endian
- ">" means big endian
- "=" means native
- "h" measn short integer (2 bytes)

In [2]:
```
x = 256

print("Network endianess")
print(struct.pack('!h', x))

print("Little endian")
print(struct.pack('<h', x))

print("Big endian")
print(struct.pack('>h', x))

print("Native endianess")
print(struct.pack('=h', x))
```

```
Network endianess
b'\x01\x00'
Little endian
b'\x00\x01'
Big endian
b'\x01\x00'
Native endianess
b'\x00\x01'
```

struct.unpack(fmt, buffer)

Unpack from the buffer buffer (presumably packed by pack(fmt, ...)) according to the format string fmt. The result is a `tuple` even if it contains exactly one item. The buffer's size in bytes must match the size required by the format.

In [3]:
```
bx = struct.pack('!h', x)

print(struct.unpack('!h', bx))
print(struct.unpack('<h', bx))
```

```
(256,)
(1,)
```

In [4]:
```
print(struct.unpack('!h', bx)[0])
print(struct.unpack('<h', bx)[0])
```

```
256
1
```