Binary Data

All Versions

Examples

Improvements requested:

Format a list of values into a byte object

```
from struct import pack

print(pack('I3c', 123, b'a', b'b', b'c'))  # b'{\x00\x00\x00abc'
```

Packing a structure

The module " **struct** " provides facility to pack python objects as contiguous chunk of bytes or dissemble a chunk of bytes to python structures.

The pack function takes a format string and one or more arguments, and returns a binary string. This looks very much like you are formatting a string except that the output is not a string but a chunk of bytes.

```
import struct
import sys
print "Native byteorder: ", sys.byteorder
# If no byteorder is specified, native byteorder is used
buffer = struct.pack("ihb", 3, 4, 5)
print "Byte chunk: ", repr(buffer)
print "Byte chunk unpacked: ", struct.unpack("ihb", buffer)
# Last element as unsigned short instead of unsigned char ( 2 Bytes)
buffer = struct.pack("ihh", 3, 4, 5)
print "Byte chunk: ", repr(buffer)
```

Output:

```
Native byteorder: little Byte chunk: '\x03\x00\x00\x00\x04\x00\x05' Byte chunk unpacked: (3, 4, 5) Byte
chunk: '\x03\x00\x00\x00\x04\x00\x05\x00'
```

You could use network byte order with data received from network or pack data to send it to network.

```
import struct
# If no byteorder is specified, native byteorder is used
buffer = struct.pack("hhh", 3, 4, 5)
print "Byte chunk native byte order: ", repr(buffer)
buffer = struct.pack("!hhh", 3, 4, 5)
print "Byte chunk network byte order: ", repr(buffer)
```

Output:

```
Byte chunk native byte order: '\x03\x00\x04\x00\x05\x00'

Byte chunk network byte order: '\x00\x03\x00\x04\x00\x05'
```

You can optimize by avoiding the overhead of allocating a new buffer by providing a buffer that was created earlier.

```
import struct
from ctypes import create_string_buffer
bufferVar = create_string_buffer(8)
bufferVar2 = create_string_buffer(8)
# We use a buffer that has already been created
# provide format, buffer, offset and data
struct.pack_into("hhh", bufferVar, 0, 3, 4, 5)
print "Byte chunk: ", repr(bufferVar.raw)
struct.pack_into("hhh", bufferVar2, 2, 3, 4, 5)
print "Byte chunk: ", repr(bufferVar2.raw)
```

Output:

```
Byte chunk: '\x03\x00\x04\x00\x05\x00\x00\x00'

Byte chunk: '\x00\x00\x03\x00\x04\x00\x05\x00'
```

Unpack a byte object according to a format string

```
from struct import unpack

print(unpack('I3c', b'{\x00\x00\x00abc'))  # (123, b'a', b'b', b'c')
```

Syntax

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

```
unpack(fmt, buffer)
```

Parameters

Remarks