

Demo of the python pack command.

For python 'pack' library see:
<https://docs.python.org/2/library/struct.html>

In Python:

```
>>> from struct import pack
```

```
>>> pack('B', 0)
'\x00'
```

```
>>> pack('B', 1)
'\x01'
```

```
>>> pack('B', 2)
'\x02'
```

```
>>> pack('B', 255)
'\xff'
```

```
>>> pack('B', 256)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
struct.error: ubyte format requires 0 <= number <= 255
```

```
>>> pack('B', -1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
struct.error: ubyte format requires 0 <= number <= 255
```

The function 'pack' converts a number to a binary string.
The '\x' means the string is being displayed using hexadecimal values.

The code 'B' means one byte which is 8 bits. There are $2^8 = 256$ values from 0 to 255.

```
pack('B', 6)
```

```
pack('B', 15)
```

```
pack('B', 16)
```

```
pack('B', 33)
```

```
pack('B', 65)
```

What does it give? Why? (See ascii table!)

```
-----  
for n in range(0, 32):  
    pack('B', n)
```

```
for n in range(0, 100):  
    pack('B', n)
```

```
-----  
  
In python, create file1:
```

```
f = open('file1', 'wb')  
str1 = pack('B', 6)  
str1  
f.write( str1 )  
f.close()
```

Show contents of file1 in binary bit-by-bit:

```
$ xxd -b file1
```

```
00000000: 00000110
```

.

one byte is 8 bits

Show contents of file1 in hex:

```
$ xxd file1
```

```
00000000: 06
```

.

one byte is 2 hex digits

```
-----  
  
In python, create file2:
```

```
f = open('file2', 'wb')  
f.write( pack('B', 6) )  
f.write( pack('B', 15) )  
f.close()
```

Show contents of file2 in binary:

```
$ xxd -b file2
```

```
00000000: 00000110 00001111
```

..

two bytes is 16 bits

Show contents of file2 in hex format:

```
$ xxd file2
```

```
00000000: 060f ..
```

two bytes is 4 hex digits

In python, create file3:

```
f = open('file3', 'wb')
for n in range(0,32):
    f.write( pack('B', n) )
```

```
f.close()
```

Show contents of file3 in binary:

```
xxd -b file3
00000000: 00000000 00000001 00000010 00000011 00000100 00000101 .....
00000006: 00000110 00000111 00001000 00001001 00001010 00001011 .....
0000000c: 00001100 00001101 00001110 00001111 00010000 00010001 .....
00000012: 00010010 00010011 00010100 00010101 00010110 00010111 .....
00000018: 00011000 00011001 00011010 00011011 00011100 00011101 .....
0000001e: 00011110 00011111 ..
```

each group of 8 bits is one byte

Show contents of file3 in hex format:

```
xxd file3
00000000: 0001 0203 0405 0607 0809 0a0b 0c0d 0e0f .....
00000010: 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f .....
```

each group of 4 hex digits is two bytes

```
f = open('file5', 'wb')
for n in range(0,128):
    f.write( pack('B', n) )
```

```
f.close()
```

For students to do:

experiment with other data formats.
Instead of 'B', try 'h' and 'i'.

Do you see how the numbers are stored in the binary file
for the other formats?

The available data formats are listed in
Section 7.3.2.2. Format Characters
at <https://docs.python.org/2/library/struct.html>