Bitwise

May 7, 2021

1 Bitwise operators

Python can use binary representation of number natively by using the prefix `0b`. For example `0b01010101`. Moreover, Python has the build-in function `bin` that given a number returns its binary representation.

Python has 6 bitwise operators.

Operator	Name
&	AND
|	OR
^	XOR
~	NOT
<<	Left-Shift
>>	Right-Shift

```
[2]: bin(144)
```

```
[2]: '0b10010000'
```

```
[8]: 0b01100110 & 0b01010101
```

```
[8]: 68
```

```
[9]: bin(0b01100110 & 0b01010101)
```

```
[9]: '0b1000100'
```

```
[10]: bin(0b01100110 | 0b01010101)
```

```
[10]: '0b1110111'
```

```
[11]: bin(0b01100110 ^ 0b01010101)
```

```
[11]: '0b110011'
```

```
[2]: bin(~0b11100110) # Note that the result is not b10011001, that's because the␣
     ↪operation has been done in Two's Complement
     # That means, making it negative and adding 1, in this case

[2]: '-0b11100111'

[13]: bin(0b01100110 << 1)

[13]: '0b11001100'

[14]: bin(0b01100110 << 2)

[14]: '0b110011000'

[15]: bin(0b01100110 << 5)

[15]: '0b110011000000'

[16]: bin(0b01100110 >> 1)

[16]: '0b110011'

[17]: bin(0b01100110 >> 2)

[17]: '0b11001'

[18]: bin(0b01100110 >> 5)

[18]: '0b11'

[3]: int(0xcafebabe).to_bytes(length=4, byteorder='little')

[3]: b'\xbe\xba\xfe\xca'

[8]: int(0xcafebabe).to_bytes(length=4, byteorder='big')

[8]: b'\xca\xfe\xba\xbe'

[9]: import sys
     sys.byteorder

[9]: 'little'

[10]: little_cafebabe = int(0xcafebabe).to_bytes(length=4, byteorder=sys.byteorder)

[11]: int.from_bytes(little_cafebabe, byteorder=sys.byteorder)
```

[11]: 3405691582

```
[12]: hex(_)
```

[12]: '0xcafebabe'

```
[14]: int(-241).to_bytes(length=2, byteorder='little')
```

```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
<ipython-input-14-9f950c3a1d5c> in <module>
----> 1 int(-241).to_bytes(length=2, byteorder='little')

OverflowError: can't convert negative int to unsigned
```

```
[15]: int(-241).to_bytes(length=2, byteorder='little', signed=True)
```

[15]: b'\x0f\xff'

```
[19]: bin(_[0])
```

[19]: '0b1010100'

2 Bytes type

Is an immutable sequence of bytes. By default the byte sequences are encoded in UTF-8. When accessing a single element from the bytes sequence, a single integer is returned (not a one byte sequence), however when an slice of the bytes sequence is retrieved a byte sequence is returned.

A new bytes object can be created in the following ways: * Single parameter constructor: * Empty constructor to obtain an empty byte sequence. * With a number, to allocate a byte sequence with that amount of `b'\x00'` in it. * A sequence of integers, the integer values must be between 0 and 255. * Two parameter constructor: * First parameter is a string `str` to be encoded and the second parameter is a string `str` to specify the encoding format. * Class method: * `fromhex` is a class method to convert a string `str` containing hex values into a bytes object.

```
[18]: b"This is OK because it's 7-bit ASCII"
```

[18]: b"This is OK because it's 7-bit ASCII"

```
[29]: b"This is not OK Ç"
```

```
  File "<ipython-input-29-18aeee4356b5>", line 1
    b"This is not OK Ç"
                       ^
```

```
SyntaxError: bytes can only contain ASCII literal characters.
```

```
[33]: bs = b"But this is \xc7"
```

```
[34]: bs.decode('latin1')
```

```
[34]: 'But this is Ç'
```

```
[35]: bs[5]
```

```
[35]: 104
```

```
[37]: bs[4:6]
```

```
[37]: b'th'
```

```
[38]: bytes()
```

```
[38]: b''
```

```
[39]: bytes(4)
```

```
[39]: b'\x00\x00\x00\x00'
```

```
[40]: bytes(range(65, 65+26))
```

```
[40]: b'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
[41]: bytes([63, 127, 228])
```

```
[41]: b'?\x7f\xe4'
```

```
[42]: bytes([63, 127, 228, 256])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-42-0c98bbd9b53d> in <module>
----> 1 bytes([63, 127, 228, 256])

ValueError: bytes must be in range(0, 256)
```

```
[43]: bytes('This is not OK Ç', 'latin1')
```

```
[43]: b'This is not OK \xc7'
```

```
[44]: ''.join(hex(c)[2:] for c in b'This is all fine')
```

```
[44]: '5468697320697320616c6c2066696e65'
```

```
[45]: bytes.fromhex('5468697320697320616c6c2066696e65')
```

```
[45]: b'This is all fine'
```