Convert base-2 binary number string to int

I'd simply like to convert a base-2 binary number string into an int, something like this:

```
>>> '11111111'.fromBinaryToInt()
255
```

Is there a way to do this in Python?

python

edited Aug 28 '16 at 9:48
wtanaka.com
181 1 8

asked Jan 19 '12 at 15:01
Naftuli Kay
24.6k 57 177 277

2 While it doesn't really matter, a binary string typically means a string containing actual binary data (a byte contains two hexadecimal digits, ie "\x00" is a null byte). – someone-or-other May 3 '14 at 18:25

5 Answers

You use the built-in `int` function, and pass it the base of the input number, i.e. `2` for a binary number:

```
>>> int('11111111', 2)
255
```

Here is documentation for python2, and for python3.

edited Feb 3 at 4:16
Aerovistae
12.6k 19 70 140

answered Jan 19 '12 at 15:02
unwind
277k 46 361 492

19 In case someone is looking for the opposite: `bin(255)` -> `'0b11111111'` . See this answer for additional details. – Akseli Palén Mar 13 '13 at 23:29

5 It should be noted that this only works for unsigned binary integers. For signed integers, the conversion options are a mess. – Fake Name Nov 1 '14 at 13:30

1 How to do this in python 3? – Saras Arya Feb 27 '15 at 5:45

2 @SarasArya It's very similar! :) I updated, see above. – unwind Feb 27 '15 at 8:01

1 And note that in an interactive REPL session (as suggested by the `>>>` prompt), you don't need to use `print` at all. The OP's hypothetical example didn't. So it really should be identical in Python 2 and 3. – John Y Jul 12 '16 at 22:36

Another way to do this is by using the `bitstring` module:

```
>>> from bitstring import BitArray
>>> b = BitArray(bin='11111111')
>>> b.uint
255
```

Note that the unsigned integer is different from the signed integer:

```
>>> b.int
-1
```

The `bitstring` module isn't a requirement, but it has lots of performant methods for turning input into and from bits into other forms, as well as manipulating them.

answered Jan 19 '12 at 15:06

Alex Reynolds
60.9k 42 189 274

Just type **0b11111111** in python interactive interface:

```
>>> 0b11111111
    255
```

answered Jan 27 '15 at 4:00

lengxuehx
451 6 16

Using int with base is the right way to go. I used to do this before I found int takes base also. It is basically a reduce applied on a list comprehension of the primitive way of converting binary to decimal (e.g. 110 = 2**0 * 0 + 2 ** 1 * 1 + 2 ** 2 * 1)

```
add = lambda x,y : x + y
reduce(add, [int(x) * 2 ** y for x, y in zip(list(binstr), range(len(binstr) - 1, -1,
-1))])
```

answered May 8 '13 at 13:04

Saurabh Hirani
513 3 13

2 Instead of defining `add = lambda x, y: x + y` , `int.__add__` can be provided to reduce. E.g. `reduce(int.__add__, ...)` – Jordan Jambazov Aug 28 '16 at 10:46

> If you wanna know what is happening behind the scene, then here you go.

```
class Binary():
def __init__(self, binNumber):
    self._binNumber = binNumber
    self._binNumber = self._binNumber[::-1]
    self._binNumber = list(self._binNumber)
    self._x = [1]
    self._count = 1
    self._change = 2
    self._amount = 0
    print(self._ToNumber(self._binNumber))
def _ToNumber(self, number):
    self._number = number
    for i in range (1, len (self._number)):
        self._total = self._count * self._change
        self._count = self._total
        self._x.append(self._count)
    self._deep = zip(self._number, self._x)
    for self._k, self._v in self._deep:
        if self._k == '1':
            self._amount += self._v
    return self._amount
mo = Binary('101111110')
```

answered Nov 16 '16 at 7:10

Mohammad Mahjoub

7 3