எங்கள் வாழ்வும் எங்கள் வளமும்
மங்காத தமிழ் என்று சங்கே முழங்கு ... *புரட்சிக்கவி*

NOTICE

www.DataScienceInTamil.com

Day 16 - Batch 3 - Python Language

Chapter 011 - Binary, Decimal,Octal, Hexa,Unary,Binary,Priorities in Binding & Bit wise operator.docx

---

To watch the recorded Python and Data Science videos in YouTube:

Day 16- Batch 3 - Binary, Decimal,Octal, Hexa,Unary,Binary,Priorities in Binding & Bit wise operator

https://youtu.be/Ikd_rRtnIWs

Official Website:

https://DataScienceInTamil.com/

மேலும் முக்கிய கேள்விகள் பதில்களுக்கு :

https://www.DatascienceInTamil.com/#faq

To join DataScienceInTamil Telegram group:

இந்த குழுவில் உங்கள் நண்பர்களை இணைக்க விரும்பினால் அதற்கான லிங்க்

https://t.me/joinchat/lUZEsr-zidpjZjEx

To Join the class, please fill the form :

https://forms.gle/QFpLHwAoinFaX2cE6

Join Zoom Meeting (From Sep 26 2022 to Oct 26 2022)

https://us06web.zoom.us/j/88900302653?pwd=MVBFUlhqTTE1LzFFRUVpTzZ2S1Vsdz09

Meeting ID: 889 0030 2653

Passcode: 1234

Monday through Friday 8 PM to 10 PM IST (From Sep 26 2022 to Oct 26 2022)

We support open-source products to spread Technology to the mass.

- This is completely a FREE training course to provide introduction to Python language
- All materials / contents / images/ examples and logo used in this document are owned by the respective companies / websites. We use those contents for FREE teaching purposes only.
- We take utmost care to provide credits whenever we use materials from external source/s. If we missed to acknowledge any content that we had used here, please feel free to inform us at info@DataScienceInTamil.com.
- All the programming examples in this document are for FREE teaching purposes only.

Thanks to all the open-source community and to the below websites from where we take references / content /code example, definitions, etc., please use these websites for further reading:

- Book : Python Notes For Professionals
- https://www.w3schools.com
- https://www.geeksforgeeks.org
- https://docs.python.org
- https://www.askpython.com
- https://docs.python.org
- https://www.programiz.com
- https://www.programiz.com/
- https://www.openriskmanagement.com/
- https://pynative.com/python-sets/
- https://www.alphacodingskills.com/
- https://codedestine.com/
- https://appdividend.com/
- https://freecontent.manning.com/

1. Binary, Octal, Decimal and Hexadecimal Numeral Systems

Number System is a way to represent numbers in computer architecture. There are four different types of the number system, such as:

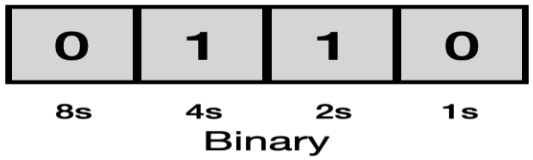
1. Binary number system (base 2)
2. Octal number system (base 8)
3. Decimal number system (base 10)

4. Hexadecimal number system (base 16)

Binary System:

A system in which information can be expressed by combinations of the digits 0 and 1.

The binary system is applied internally by almost all latest computers and computer-based devices because of its direct implementation in electronic circuits using logic gates. In Python, we use bin() to convert the decimal to binary.



Value of 6 represented in Binary

(15) = 8 + 4 + 2 + 1

128	64	32	16	8	4	2	1
0	0	0	0	1	1	1	1

```
a = 15
print(bin(a))
```

Output

```
0b1111
```

---

32 = 32

128	64	32	16	8	4	2	1

0	0	1	0	0	0	0	0

```
a = 32
print(bin(a))
```
Output : 0b100000

---

150 = 128 + 16 + 4 + 2

128	64	32	16	8	4	2	1
1	0	0	1	0	1	1	0

```
a = 150
print(bin(a))
```
Output : 0b10010110

---

301 = (128 + 64 + 32 +16 + 8 + 4 + 2 + 1) These values are from first byte. Remaining 45 will be stored in 2^{nd} byte.

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	1
0	0	1	0	1	1	0	1

1^{st} byte = 255 2^{nd} byte

```
a = 300
print(bin(a))
```
Output : 0b100101100

Binary to Decimal conversion (Positional Notation):

$(11001010)_2$
$1x2^7 + 1x2^6 + 0x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0$
$128 + 64 + 0 + 0 + 8 + 0 + 2 + 0$
$(\mathbf{202})_{10}$

Do it yourself(Homework)
$(1010.1011)_2$
$1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 1x2^{-4}$
$8 + 0 + 2 + 0 + 0.5 + 0 + 0.125 + 0.0625$
$(10.6875)_{10}$

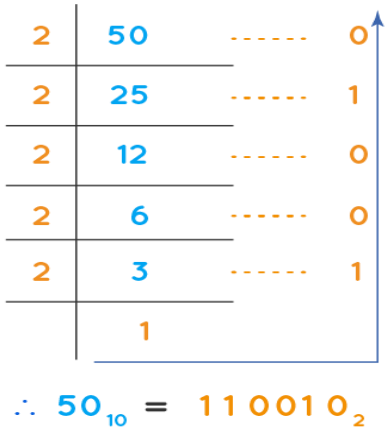
Binary to Decimal conversion (Doubling):

How to convert binary to decimal using Doubling process.

$(11101110)_2 = (238)_{10}$
1
1*2 +1 =3
3*2 +1=7
7 * 2 +0=14
14 *2 +1=29
29 * 2 +1=59
59 * 2 +1=119
119 * 2+0=238
$(238)_{10}$

Decimal to Binary conversion:

50 is the decimal number. The binary form of 50 is 00110010_2



14's binary Equivalent number is **$(1110)_2$**.
14 ÷ 2 = Q(7) R (0)
7 ÷ 2 = Q(3) R (1)
3 ÷ 2 = Q(1) R (1)
1 ÷ 2 = Q(0) R (1)

The value of 117 in binary is ( )$_2$

How to convert Floating **binary to decimal**

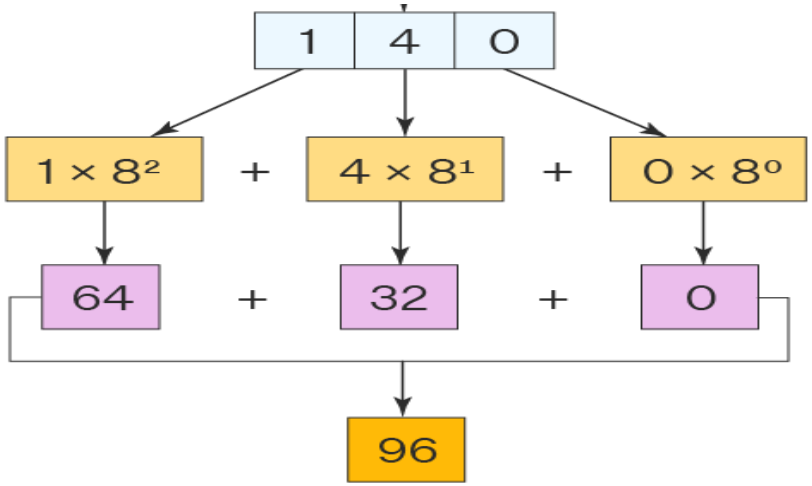
The Binary Representation of $(00.59375)_{10}$ is $(0.10011)_2$

0.59375 * 2 = 1.1875 (1 - 0.1875)
0. 1875 * 2 = 0.375 (0 – 0.375)
0.375 * 2 = 0.75 (0 – 0.75)
0.75 * 2 = 1.5 (1 – 0.5)
0.5 * 2 = 1.0 (1 – 0.0)

Octal System:

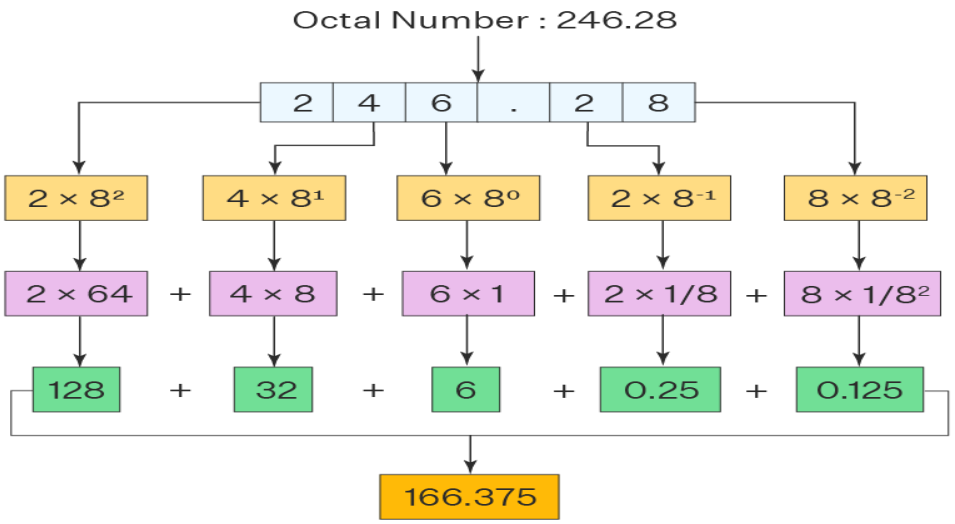
Octal numbers are represented with a base of 8.

Octal to Decimal conversion :



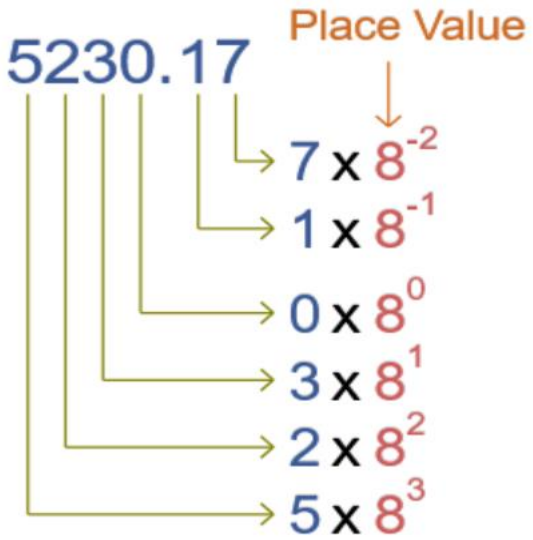
Decimal Number : 140

$(140)_8 = (96)_{10}$

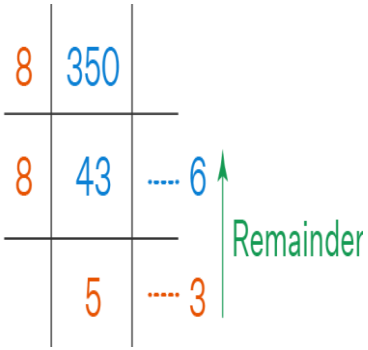


$(246.28)_8 = (166.375)_{10}$

$(317)_8 = 3 \times 8^2 + 1 \times 8^1 + 7 \times 8^0 = (207)_{10}$



Decimal to Octal conversion:



$(350)_{10} = (536)_8$

$(440)_{10} = (670)_8$

440 / 8 = Q(55) R(0)

55 / 8 = Q(6) R(7)

```
a = 440
print(oct(a))
```

Output : 0o670

Hexadecimal System:

The hexadecimal number system has a base value equal to 16. It is also pronounced sometimes as 'hex'. Hexadecimal numbers are represented by only 16 symbols. These symbols or values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Each digit represents a decimal value.
A = 10 , B = 11, C = 12, D = 13, E = 14 and F = 15 .

Hexadecimal to Decimal conversion :

$(7DE)_{16}$ = $(2014)_{10}$

$7DE = (7 * 16^2) + (13 * 16^1) + (14 * 16^0)$

$7DE = 1792 + 208 + 14 = 2014$

$(80E1)_{16}$ = $(32993)_{10}$

$8 * 16^3 + 0 * 16^2 + E * 16^1 + 1 * 16^0$

Decimal to Hexadecimal conversion :

$(242)_{10}$ = $(F2)_{16}$

16	242	
16	15	$2 \rightarrow 2$
	0	$15 \rightarrow F$

$(501)_{10}$ = $(1F5)_{16}$

501 / 16 = 31.3125

0.3125 * 16 = **5** (To calculate the remainder, multiply the decimal part by 16, gives least significant digit = 1st remainder = 5 here)

31 / 16 = 1.9375 (Quotient before the decimal point / 16)

0.9375 * 16 = **15** (Remainder part Multiply)

1 / 16 = 0.0625 (last quotient / 16)

0.0625 * 16 = **1** (Remainder part multiply)

The remainders written from below to top give you the hex values 1, 15 and 5.

81_{10} = ( )$_{16}$

16	81	
16	5	$1 \rightarrow 1$
	0	$5 \rightarrow 5$

2. Unary & Binary Operators in Python

Unary Operator :

For eg., 5 + 2 . Here, 5 becomes 1st operand and 2 becomes 2nd operand. '+' is the operator.
A unary operator is an operator which works on a single operand.

What are all the Unary operators supported by Python.

Python support unary minus operator (-).

If a number is positive, it becomes negative when the number is preceded by the unary operator.

i)
```
x=100
y=-(x)
print(y)
```

Output:
-100

ii)
```
a = 10
print(-a)

b = -11
print(-b)
```

Output

-10
11

iii)
```
a = 5
b = -6
print(a + b)     # Binary operator '+'
print(-(a + b)) #Unary operator '-' is operated on a single expression (a+b)
```

Output
-1
1

Binary Operator :

A binary operator operates on two operands. Arithmetic operators are examples of binary operators.

```
print(1 + 2)
print(3 - 4)
print(5 * 6)
print(7 / 8)
```

output
```
3
-1
30
0.875
```

3. Priorities and Binding

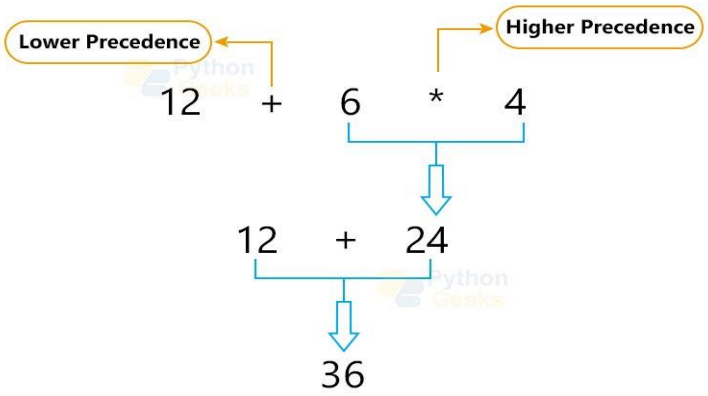
Priority implies that some operators act before others in a given expression.

Binding of operator indicates the order of computations by operators with equal priority within a expression. Python mostly has left side binding.
Subexpressions in parentheses are always calculated first. For almost all the operators the associativity is left-to-right, except for exponential(**), logical NOT and assignment operators(=).

Python follows PEMDAS rule.

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR



i) print (9 % 6 % 2)

-from left to right: first 9 % 6 gives 3, and then 3 % 2 gives 1;

-from right to left: first 6 % 2 gives 0, and then 9 % 0 causes a fatal error

ii) print ((7 * ((25 % 13) + 100) / (2 * 12)) // 2)
Output : 16.0

iii) print (5+8/2)
Output : 9.0

iv) print ((8+6)/2)
Output : 7.0

v) print ((6+8)*8-9/3+30)
Output : 139.0

vi) 3*(4//3)
3*4//3
Output : 3, 4

vii) 3-4+7
3-(4+7)
Output : 6-8

viii) print(not True or True)
Output:

ix) p = 1
q = 2
if(p > 0 and q > 0):
print('p and q are positive integer numbers.')

x) print(2**1**2)
print((2**1)**2)
Output : 2, 4

xi) a = 2>3
b = 3==3
b = 3==3 + 1
Output : False
True

Examples of non-associative operators are the assignment and comparison operators. For example a<b<c, actually check for a<b **and** b<c.
Similarly, chaining of assignments, a=b=c=1 is also valid.
7<5<9 Output : False
print(2 > (5<9)) Output : True
print(3 <= True) Output : False
---
a=b=c=6
print(a,b,c)
Output:
6 6 6
however, we cannot use the other assignment operators like +=, -=, etc. ---

xii) Change the order of the OR's, AND's to get in the expression
(3<=6) and (9==0) and (5>-1) or 4
output is True. Show different possibilities using coding
(3<=6) and (9==0) and (5>-1) or 4
(3<=6) and (9==0) or (5>-1) and 4
(3<=6) or (9==0) and (5>-1) and 4
Output:

4
4
True

4. Bitwise Operators

Bitwise operators are used to compare (binary) numbers.
In Python, the following are the mostly commonly used bitwise operators are used.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

A	B	A | B	A & B	A^ B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0

1 1 1 1 0 0

Bitwise Shift Operators

Shift operators are used to shifting the bits of a number left or right thereby multiplying or dividing the number by two respectively. They are used when we have to multiply or divide a number by two.

Bitwise Right Shift Operators

It shifts the bits of the number to the right and fills 0 on blank/voids right as a result. It provides a similar kind of effect as of dividing the number with some power of two.

x = 7
x >> 1
x = 3

Bitwise Left Shift Operator

It shifts the bits of the number to the left and fills 0 on blank/voids left as a result. It provides a similar kind of effect as of multiplying the number with some power of two.

x = 7
x << 1
= 14

a) a = 60
   b = 13
   Now in the binary format their values will be
   a = 0011 1100
   b = 0000 1101

   A & b = 0000 1100 (If both are 1, it becomes 1. All others are 0)
   A | b = 0011 1101 (if 1 is present in one operand, it becomes 1)

A ^ b = 0011 0001 (Both 1's and both 0's become 0. 1 & 0 0r 0& 1 becomes 1)
~a = 1100 0011 (Unary Operator – it makes 0 to 1, 1 to 0)

b)
```
a = 5
b = 6
# Print bitwise AND operation
print("a & b =", a & b)
# Print bitwise OR operation
print("a | b =", a | b)
# Print bitwise NOT operation
print("~a =", ~a)
# print bitwise XOR operation
print("a ^ b =", a ^ b)
```

Output

```
a & b = 4
a | b = 7
~a = -6
a ^ b = 3
```

c)
```
c = 10
# print bitwise right shift operator
print("c >> 1 =", c >> 1)

c = 5
 # print bitwise left shift operator
print("c << 1 =", c << 1)
```

Output

c >> 1 = 5
c << 1 = 10

d) a = 10 # 0000 1010
print(a >> 1) 0000 0101
Output
5

e) a = 5 # 0000 0101
print(a << 1) # 0000 1010 = 10
print(a << 2) # 0001 0100 = 20

f) a = 10 = 1010
~a = ~1010
= -(1010 + 1)
= -(1011)
= -11 (Decimal)

g) a = 20
b = 8
print("a & b =", a & b) # Print bitwise AND operation
print("a | b =", a | b) # Print bitwise OR operation
print("~a =", ~a) # Print bitwise NOT operation
print("a ^ b =", a ^ b) # print bitwise XOR operation

h) a = 10 = 1010
b = 4 = 0100
print(a ^ b) # 1010
Output

1110 = 14 (Decimal)