Dört Built-in Data Yapısı

- Python, herhangi bir veriyi tutmak için kullanabileceğiniz dört yerleşik veri yapısıyla birlikte gelir :

- **List**
- **Tuple**
- **Dictionary**
- **Set**

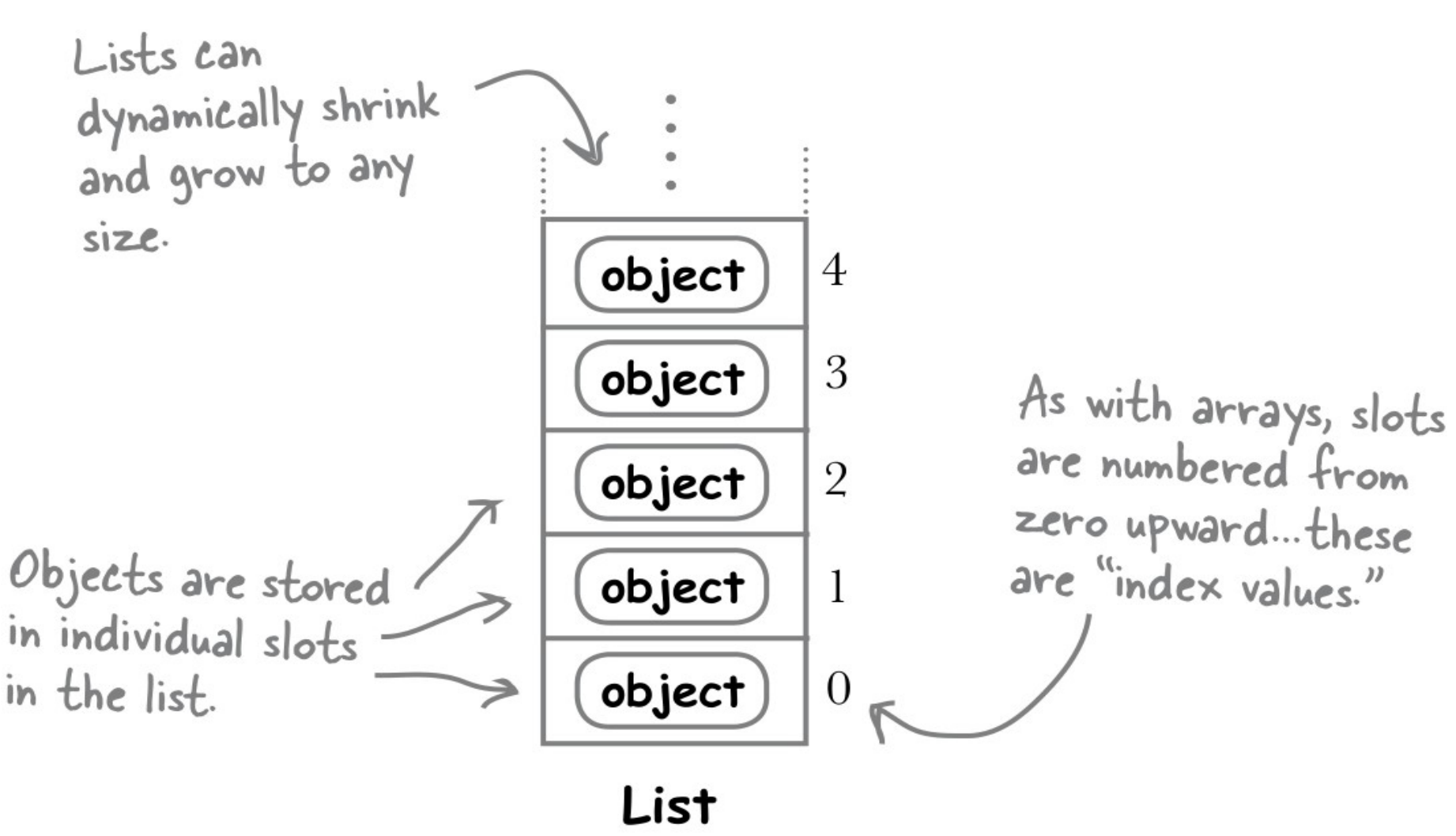
Dört Built-in Data Yapısı

- "**Built-in**" derken, list, tuples, dictionary ve set'lerin kodunuz için her zaman kullanılabilir olduğunu ve kullanımdan önce içe aktarılmaları gerekmediğini unutmayın: bu veri yapılarının her biri Python dilinin bir parçasıdır.

• List

Sıralı-Değiştirilebilir-Dinamik-Heterojen

- Python'daki bir list, diğer dillerdeki bir dizi kavramına çok benzer.
- Çünkü bir list'i ilgili nesnelerin indexlenmiş bir koleksiyonu olarak düşünebilirsiniz, list'teki her slot (objenin yerleştiği kutu-index ile işaretlenen) sıfırdan yukarıya doğru numaralandırılmıştır.


Lists can dynamically shrink and grow to any size.
object
object
object
object
object
4
3
2
1
0
As with arrays, slots are numbered from zero upward...these are "index values."
Objects are stored in individual slots in the list.
List

•List

- Listler “**dinamik**” tir :
- **Dinamik :** Diğer birçok programlama dilindeki dizilerin aksine, Python'da list’ler dinamiktir, çünkü istenilince büyüyebilir (ve küçülebilir).
- Herhangi bir nesneyi depolamak için kullanmadan önce bir list’in boyutunu önceden bildirmeye gerek yoktur.
- Dinamik olması; obje eklenince kendiliğinden boyutunu artırması,

  obje çıkarılınca da kendiliğinden boyutunu küçültmesi ile ilgilidir.

•List

- Listler “**heterojen**” dir :
- Listler heterojendir; çünkü depoladığınız nesnenin türünü önceden belirtmeniz gerekmez—isterseniz tek bir listede farklı türlerdeki nesneleri karıştırıp depolayabilirsiniz.

List

- Listler “**değiştirilebilir**” dir :
- Listeler **değiştirilebilirdir ;** çünkü bir listeyi istediğiniz zaman ekleyerek, kaldırarak veya değiştirerek değiştirebilirsiniz.

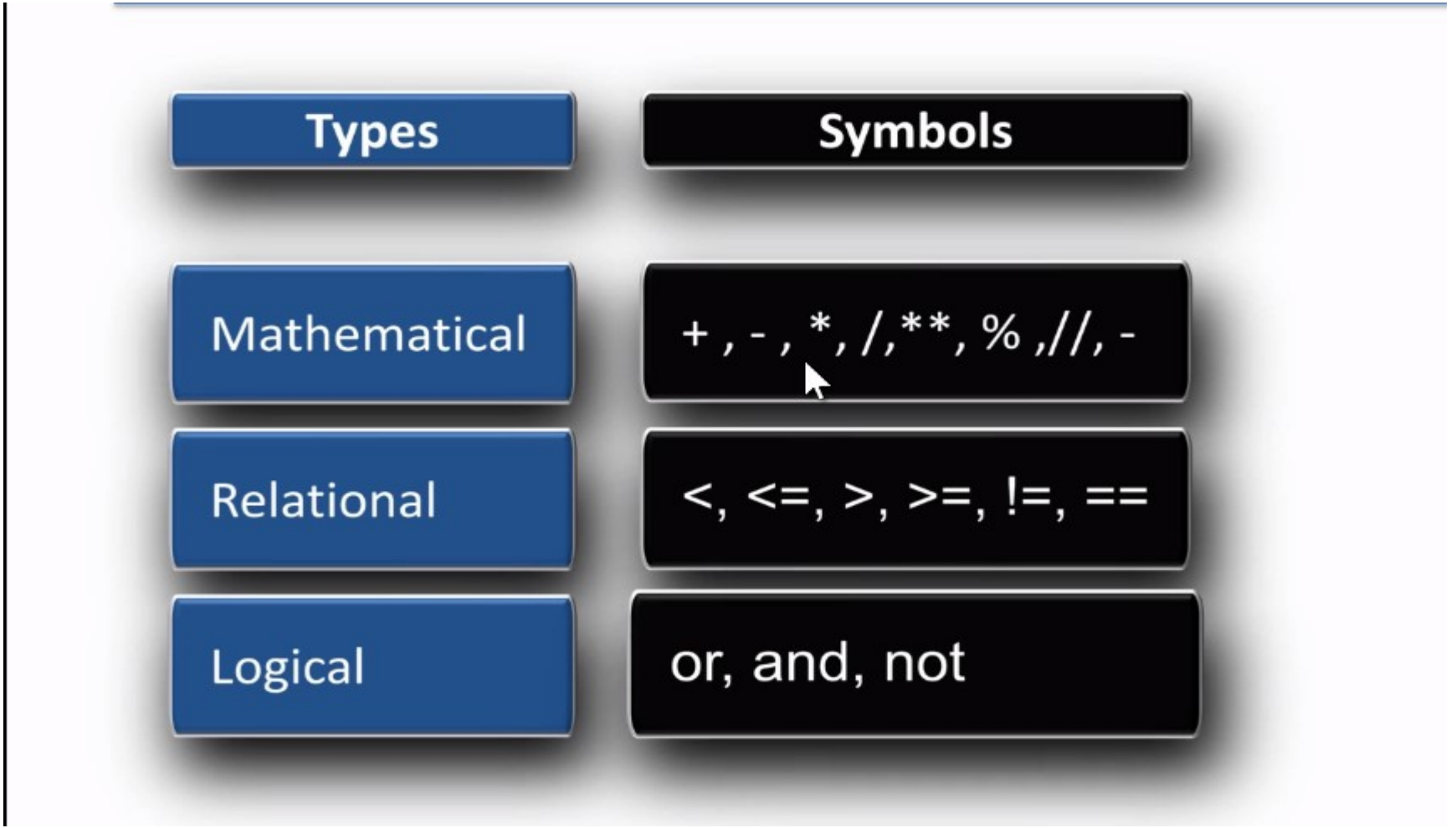
Operatörler ve Operand'lar

- **Operatörler**, Python interpreter'a bazı matematiksel veya mantıksal işlemler yapmasını söyleyen sembollerdir.
- Örneğin ;
- **3 * 2** işleminde ***** sembolü bir operatördür.
- 3 ve 2 ise operand'tır.
- **Operand**; üzerinde matematiksel ya da mantıksal işlem yapılan elemana denilir.

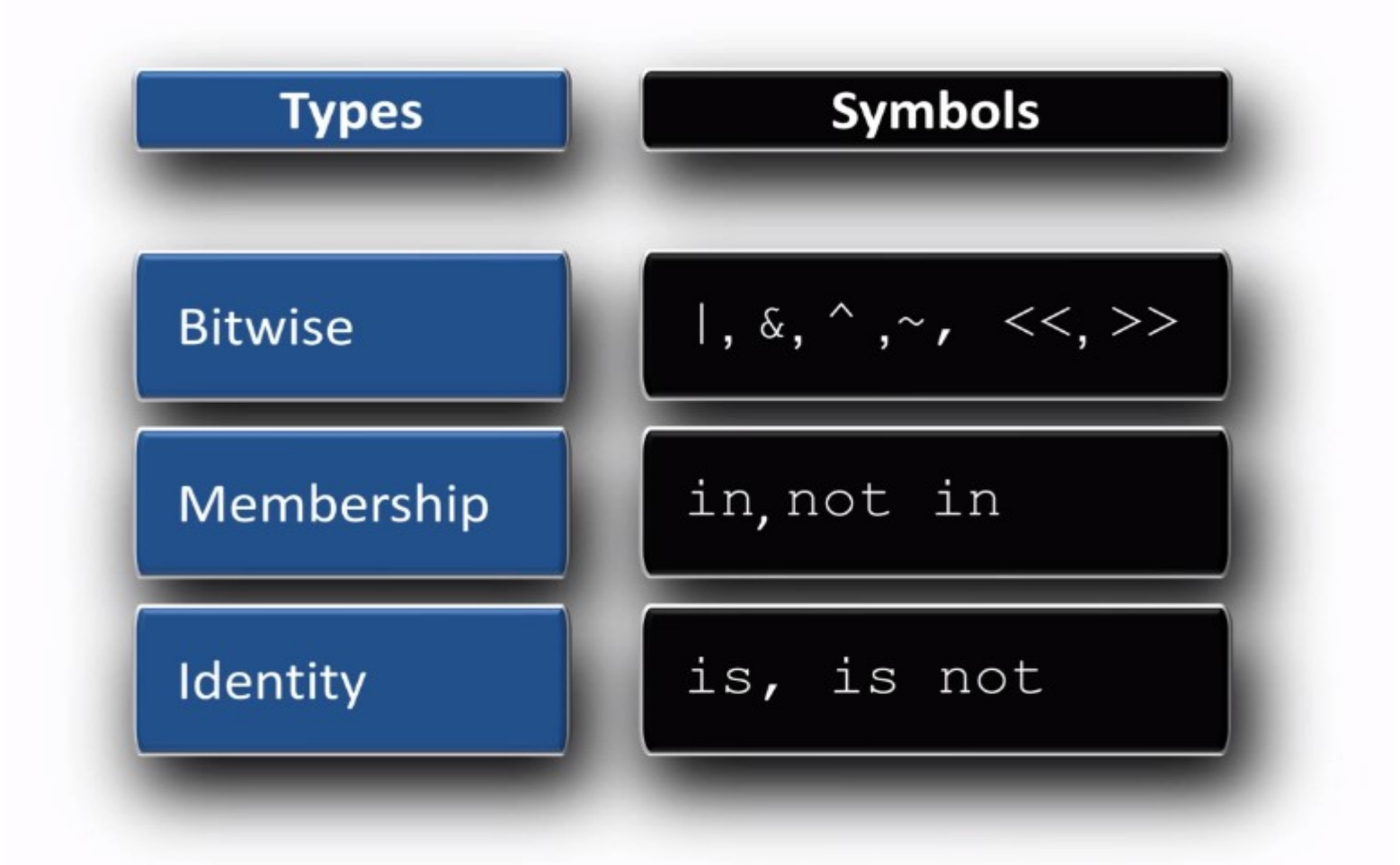
Operatörler ve Operand’lar

- Örneğin ;
- **-5**
- Burada **–** sembolü bir operatör, **5** ise operand’tır.
- 7 – 2
- 7 ve 2 operandlardır.
- - sembolü ise operatördür.

Operatör Türleri



Operatör Türleri



Aritmetik (Matematiksel) Operatorler

Arithmetic Operators in Python

Arithmetic Operators		
Operator	**Meaning**	**Usage**
+	Addition or unary plus	x + y +2
-	Subtraction or unary minus	x - y -2
*	Multiplication	x * y
/	Division (result is always a float)	x / y
%	Modulus	x % y (remainder of x/y)
//	Floor division - results into whole number (may be float)	x // y
**	Exponent	x**y (x to the power y)

İşlem Önceliği Sırası

Aşağıdaki tabloda öncelik sırası en yüksekten en düşüğe - yukarıdan aşağı doğru gitmektedir.

Precedence table from highest to lowest. (The highest precedence is on the first row).

Operator	Description
**	Exponentiation
~, +, -	Complement, unary plus and unary minus
*, /, %, //	Multiply, divide, modulo and whole (floor) division
+, -	Addition and subtraction
>>, <<	Right and left bitwise shift
&	Bitwise and
^ ,|	Bitwise exclusive or and bitwise or
<=, < ,> ,>=,	Relational operators
<> ,== ,!=	Equality operators
= ,%= ,/= ,//= ,-= ,+= ,*=, **=	Assignment operators
is , is not	Identity operators
in not in	Membership operators
not , or , and	Logical operators

İşlem Önceliği Sırası

- Aynı işlem önceliği seviyesine sahip operatörlerde işlem sırası soldan sağa doğru gider.

İşlem Önceliği Sırası

- print (3 * 2 ** 3)
- 24

- print (2 ** 3 / 4 + 6)
- 8.0

- print (40*4/20*5)
- 40.0

Relational / Comparison Operators
(Karşılaştırma Operatörleri)

Relational / Comparison Operators in Python

Comparison Operators		
Operator	**Meaning**	**Usage**
>	Greater than	x > y
<	Less than	x < y
==	Equal to	x == y
!=	Not equal to	x != y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Relational / Comparison Operators
(Karşılaştırma Operatörleri)

- Karşılaştırma operatörü kullandığınızda karşılaştırma yaptığınızı unutmamalısınız.
- Karşılaştırma yaptığınız için sonuçları mantıksal değerler olan **True** veye **False** döner. (True ve False'un bool tipinde değerler olduğunu hatırlayın.)
- Yani **<** opertörü ile **"Küçük müdür ?"** diye soru sormuş olursunuz. Cevap evet ise sonuç **True** döner, hayır ise **False** döner.

Relational / Comparison Operators
(Karşılaştırma Operatörleri)

- print(5 < 4)
- False

- print(8 == 9.0)
- False

- a = 3

  b = 5

  print(a != b)
- True

Mantıksal Operatorler

Logical Operators		
Operator	**Meaning**	**Usage**
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	complements the operand	not x

Mantıksal Operatorler ”and”

- >>> True and False

  False
- >>> True and True

  True
- >>> False and False

  False

Mantıksal Operatorler
”or”

- >>> True or False

  True

- or operatorunda taraflardan birinin True olması True sonucunu verir. Sadece her iki taraf da False olursa sonuç False olur.

- >>> False or False

  False

Mantıksal Operatorler
”not”

- >>> not True

  False

- >>> not False

  True

- >>> not (5 > 4)

Mantıksal Operatorler

- Mantıksal operatorler kullanarak (and ve or ile) birden fazla karşılaştırma ifadesini bağlayabilirsiniz.
- Böylece birden fazla koşulu bir araya getirebilirsiniz.
- >>> (5 > 4) and (3 == 4)

  False
- >>> (5 > 4) or (3 == 4)

Üyelik Operatörleri

Membership Operations		
Operator	**Meaning**	**Usage**
in	Check to see if value is in the sequence	5 in [2,5,3,7]
not in	Check to see if value is not in the sequence	5 not in [2,5,3,7]

Üyelik Operatörleri

- >>> 5 in [2, 3, 4, 7]

  False

- >>> 5 not in [2, 3, 4, 7]

  True

- Üyelik operatörleri bir elemanın bir dizi yapısı içerisinde olup olmadığını kontrol eder.
- >>> x = [6, ‘cat’, ‘dog’, 4.5]

Üyelik Operatörleri

- >>> my_str = ‘hello there’
- >>> ‘z’ **in** my_str

  False
- >>> ‘e’ **in** my_str

  True
- >>> ‘g’ **not in** my_str

  True

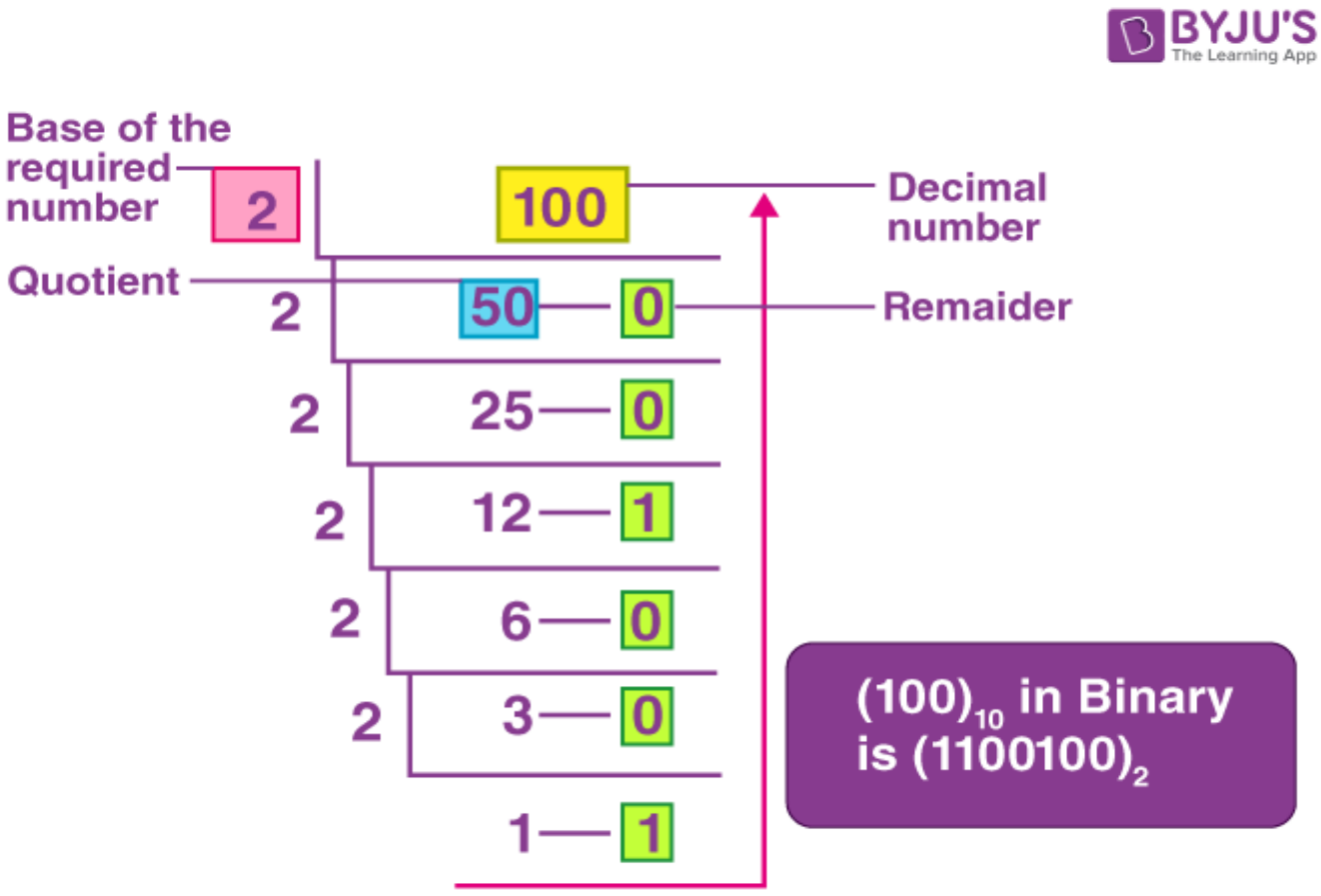
Bitwise Operatorler

- Python'da, bitsel operatörler, tamsayılar üzerinde bitsel hesaplamalar yapmak için kullanılır. Tamsayılar önce ikilik tabandaki karşılıklarına dönüştürülür ve ardından işlemler bit bit üzerinde gerçekleştirilir, bu nedenle bitsel operatörler adı verilir. Ardından sonuç ondalık biçimde döndürülür.

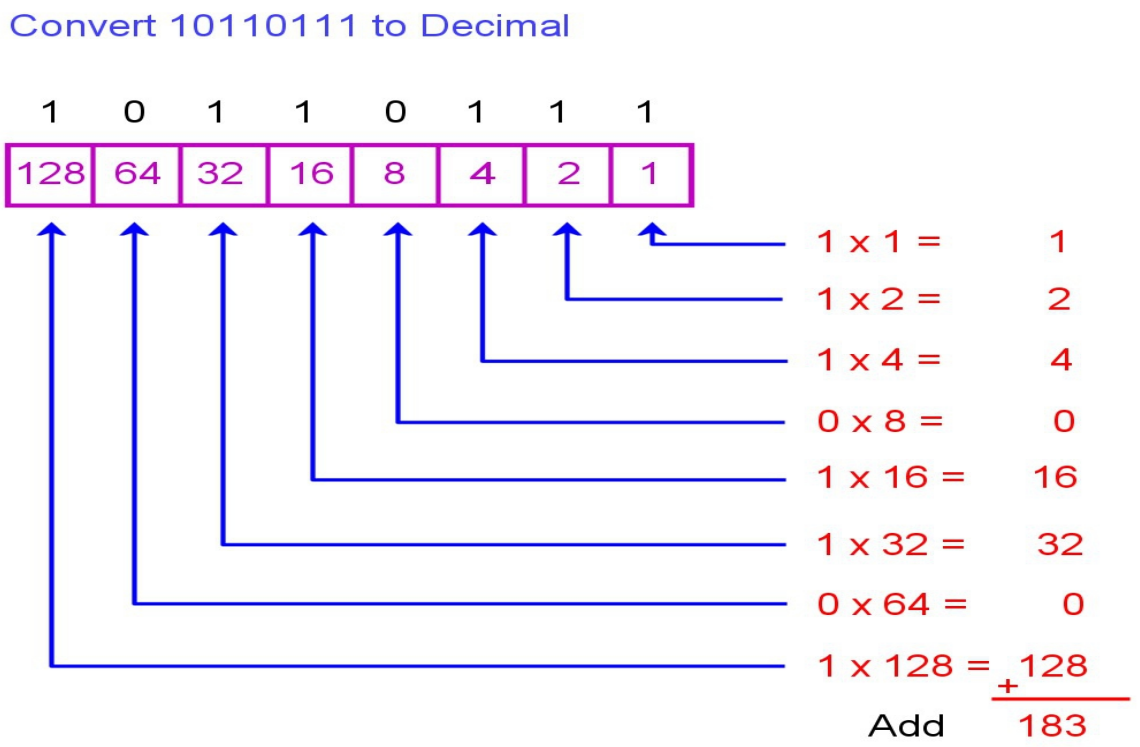
Bitwise Operatorler

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Bitwise Operatorler



Bitwise Operatorler



Bitwise Operatorler

- a = 10 = 1010 (Binary)

  b = 4 = 0100 (Binary)

  a & b = 1010

  &

  0100

  = 0000

Bitwise Operatorler

- a = 10 = 1010 (Binary)
- b = 4 = 0100 (Binary)

- a | b = 1010
  |
  0100
  = 1110

Bitwise Operatorler

- a = 10 = 1010 (Binary)
-
- ~a = ~1010
  = ~(1010)
  = (0101)
  = 5 (Decimal)

Bitwise Operatorler

- Bitwise XOR sembolü ^ şeklindedir.
- Bit değerlerinden her ikisi de aynı ise sıfıra dönüştürür, her ikisi de farklı ise 1 alır.
-

Bitwise Operatorler
left shift operator (<<)

Expression	Binary Value	Decimal Value
a	100111_2	39_{10}
a << 1	1001110_2	78_{10}
a << 2	10011100_2	156_{10}
a << 3	100111000_2	312_{10}

Bitwise Operatorler
right shift operator (>>)

Expression	Binary Value	Decimal Value
`a`	10011101_2	157_{10}
`a >> 1`	1001110_2	78_{10}
`a >> 2`	100111_2	39_{10}
`a >> 3`	10011_2	19_{10}

Conditional Statements
" if " Statement

- Koşul belirtmek için kullanılırlar.
- Belirli bir koşulun sağlanması halinde bir işlemi gerçekleştirmek istediğimizde kullanırız.
- Örneğin ;
- "Eğer" yağmur yağarsa, eve gideceğim.
- Buna benzer şekilde programlamada da belirli bir şartın-durumun sağlanması halinde bir işlemin gerçekleşmesini, ya da bir kod satırının çalışmasını istediğimiz durumlarda " if " statement kullanırız.

Conditional Statements
" if " Statement

- Bir şartın sağlanması durumunda "True" döndüğünü biliyoruz. Örneğin ;

  5 in [1, 3, 5] durumunun sonucu True döner.

- Ya da ;
- 8 == 3 sonucu False döner.

if else Statement

```
if expression:
    statement(s)
    . . .
    . . .
else :
    statement(s)
    . . .
print("not inside if")
```