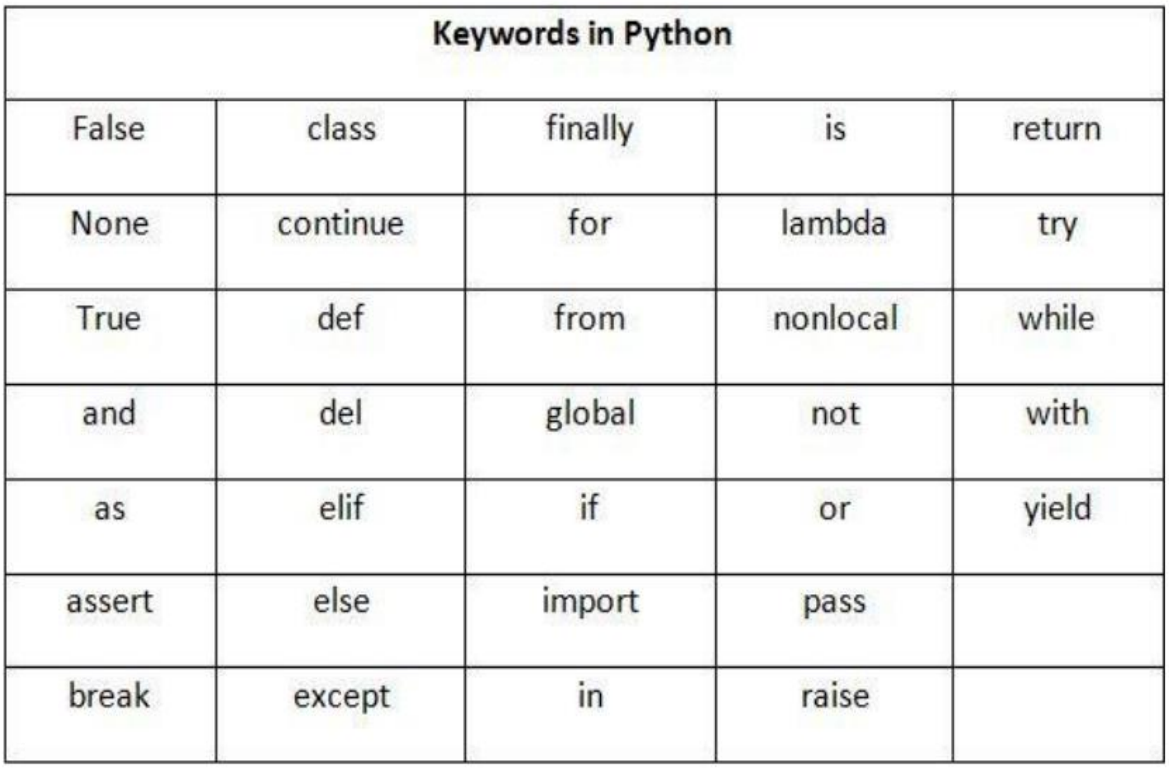


Keywords in Python

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Indentation:

Indentation is describe the block of code

Generally 4 whitespaces are used for indentation

Examble:

```
For i in range(10,20):
    print(i)
if( i ==10):
   break
```

comments:

single line comment : # this is demo

multiple line comment : “””………”””

DATA TYPES IN PYTHON

1. Numbers

2. String

3. List

4. Tuple

5. Set

6. Dictionary

1.Numbers:

Three types:

1.int

a=10

type(a)

2.float

a=35.6

type(a)

3.complex

a=15+5j

type(a)

String: String is a sequence of Unicode character. The user can use **single quotes or double quotes** .String are **immutable** object in python

s=’hii I am karan’

print(s)

type(s)

List :

The list is stored as an ordered sequence of items, this items are **mutable.** Items separated by commas are enclosed within **square bracket[]**

```
a = [10,20,30,40,50]

print(a)

type(a)
```

Tuple: The list is stored as an ordered sequence of items, this items are **immutable.** Items separated by commas are enclosed within **parentheses bracket()**

```
t=(10,20,30,'hii')

print(t)

type(t)
```

set: The set is an **unordered collection** of unique elements**. immutable** behaviour. Values separated by commas inside **curly braces{}**

```
a={10,20,30,"karan",10}

print(a)            #{10,20,30,"karan"}

print(type(a))
```

Dictionary:

The Dict is an **unordered collection** of **key value pairs** this written with two curly braces {}

```
d={"name":"karan","age":22,"address":"Chennai"}

print(type(d))

print(d)

print(d["name"])
```

variables: variables using for storing the data

name=”karan”

x=1

Rules of create variables:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Python Operators

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

Example

```
print(10 + 5)
```

list of operators:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

1. **Arithmetic operators:**

operators	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	multiplication	x * y
/	Division	x / y
%	modules	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

Assignment operators are used to assign values to variables

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

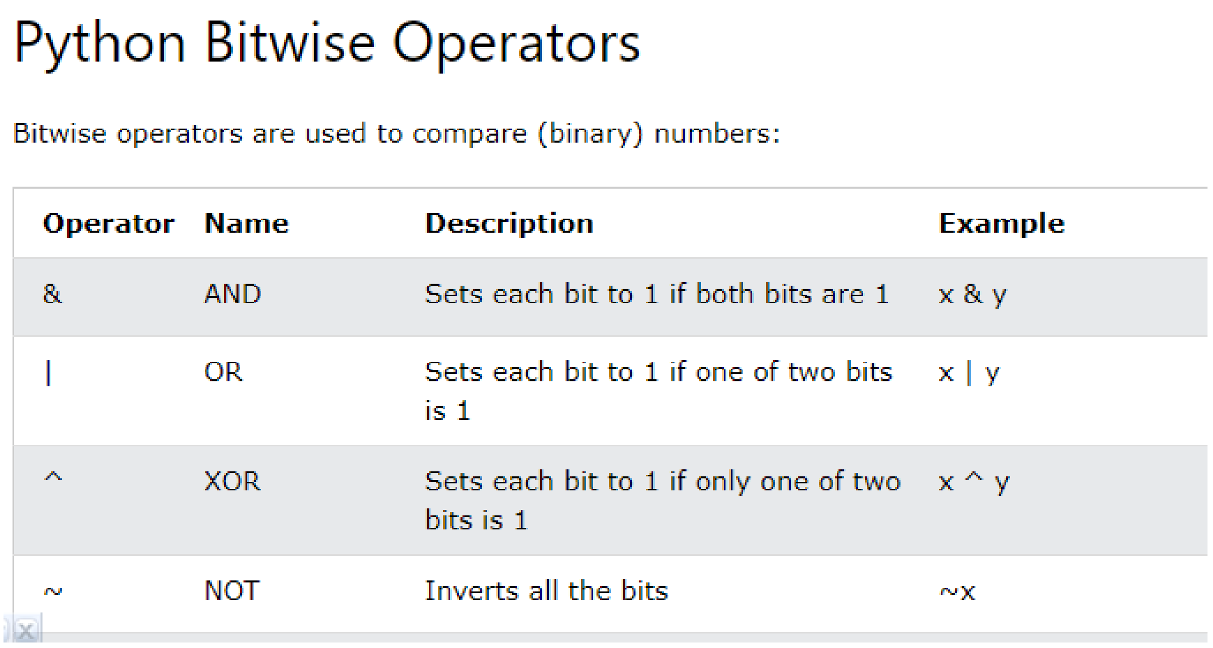
Identity operator:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y



Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x