Python Tokens Types

Python breaks each logical line into a sequence of elementary lexical components known as **Tokens**. The normal token types are

1) Identifiers,

2) Keywords,

3) Operators,

4) Delimiters and

5) Literals.

Identifiers

An Identifier is a name used to identify a variable, function, class, module or object.

Example of valid identifiers

Sum, total_marks, regno, num1

Keywords

Keywords are special words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

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	

Operators

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called **operands**. Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as **operands**.

Python supports the following Arithmetic operators.

Operator - Operation	Examples	Result
Assume a=100 and b=10. Evaluate the following expressions		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30 (Integer Division)	3

Python supports following relational operators

Operator - Operation	Examples	Result
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

Python supports following Logical operators

Operator	Example	Result
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

Python supports following Assignment operators

Operator	Description	Example
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b="Computer"
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5
*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2
%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
**=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3

Conditional operator

Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.

The Syntax conditional operator is,

Variable Name = [on_true] if [Test expression] else [on_false]

Example :

min= 50 if 49<50 else 70 # min = 50

min= 50 if 49>50 else 70 # min = 70

Delimiters

Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings. Following are the delimiters.

(	)	[	]	{	}
,	:	.	‘	=	;
+=	-=	*=	/=	//=	%=
&=	|=	^=	>>=	<<=	**=

Literals

Literal is a raw data given in a variable or constant. In Python, there are various types of literals.

1) Numeric 2) String 3) Boolean

4)Escape Sequences

In Python strings, the backslash "\" is a special character, also called the **"escape"** character. It

Python supports the following escape sequence characters.

Escape sequence character	Description	Example	Output
\\	Backslash	>>> print("\\test")	\test
\’	Single-quote	>>> print("Doesn\'t")	Doesn't
\”	Double-quote	>>> print("\"Python\"")	"Python"
\n	New line	print("Python","\n","Lang..")	Python Lang..
\t	Tab	print("Python","\t","Lang..")	Python Lang..