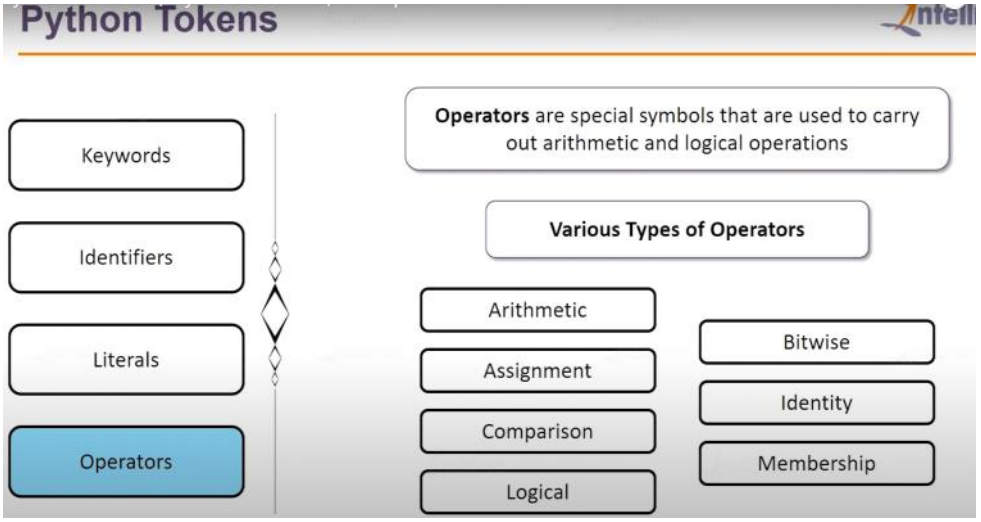
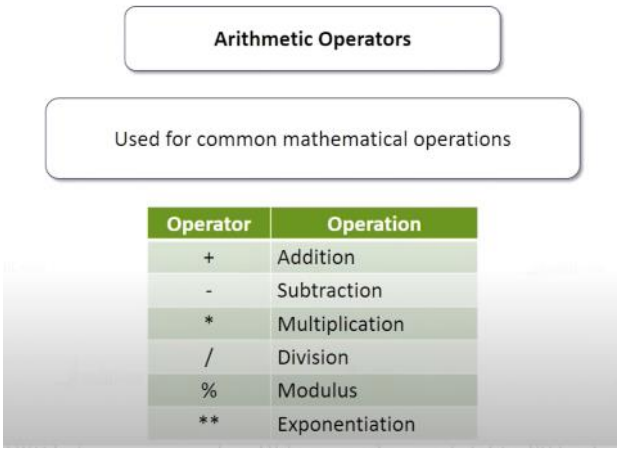
Python operator

Wednesday, August 17, 2022 2:09 PM

A



Arithmetic operators



Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation

Assignment operators

Assignment Operators

Used to assign values to variables

Operator	Operation
=	X=10
+=	X=X+2
-=	X=X-29
*=	X=X*12
/=	X=X/3
|=	X=X|6

Comparison operator

Comparison Operators

Compares values and returns either True or False

Operator	Operation
==	Equal
!=	Not equal
<	Less than
>	Greater than
>=	Greater than or equal to
<=	Less than or equal to

Logical Operator

Logical Operators

Used to combine conditional statements

Operator	Description
and	True if both statements are True
or	True if one of the statements is True
not	If True, then returns False

Bitwise Operator

Bitwise Operators

Used to compare binary numbers

Operator	Operation
&	AND
|	OR
^	XOR
~	NOT
<<	LEFT SHIFT
>>	RIGHT SHIFT

Note:
In python every number has it own I because it is object oriented programming language

```
>>> id(1)
140722158035792
>>> type(1)
<class 'int'>
>>> type(2)
<class 'int'>
>>> id(2)
140722158035824
```

Identity Operator

Identity Operators

Used to check if the objects are the same or not

Operator	Operation
Is	Returns True if both variables are the same object
Is not	Returns True if both variables are not the same object

Think you a array of [1 2 3]
Now you ask the user a input and verify it in the array is there or not fore this we work through loops
But in python we don't need that we use Membership operator

Membership Operators

Used to test if a sequence is present in an object

Operator	Operation
in	Returns True if the specified value is present in the object
not in	Returns True if the specified value is not present in the object

```
>>> nums = [1, 2, 3, 4]
>>> nums
[1, 2, 3, 4]
>>> 4 in nums
True
>>> 8 in nums
False
```

Now the user want to add new data in the array then we use append function

```
>>> nums.append(8)
>>> nums
[1, 2, 3, 4, 8]
>>> 8 in nums
True
```

to remove some data from the last index of the array we use pop

```
>>> nums.pop()
8
>>> nums
[1, 2, 3, 4]
```