PYTHON IDENTIFIERS:

Identifiers are names for entities in a program such as class, variables and functions etc.

Rules for defining Identifiers:

Identifiers can be composed of uppercase, lowercase letters, underscore and digits bur should start only with an alphabet or an underscore.

- Identifiers can be a combination of lowercase letters (a to z) or uppercase letters (A to Z) or digits or an underscore.
- Identifiers cannot start with digit
- Keywords cannot be used as identifiers.
- Only (_) underscore special symbol can be used.

  Valid Identifiers: sum total _ab_ add_1

  Invalid Identifies: 1x x+y if

VARIABLES

A variable is nothing but a reserved memory location to store values. A variable in a program gives data to the computer.

Ex:

```
>>>b=20
>>>print(b)
```

PYTHON INDENTATION

Python uses indentation. Block of code starts with indentation and ends with the unintended line. Four whitespace character is used for indentation ans is preferred over tabs.

Ex:

```
x=1
if  x==1:
        print("x is 1")
```

Result:

```
x is 1
```

EXPRESSIONS

An Expression is a combination of values, variables and operators.

Ex:

```
>>>10+20
12
```

STATEMENTS

A Statement is an instruction that a python interpreter can execute.IN python enf of a statement is marked by a newline character.

c=a+b

Multiline statement can be used in single line using semicolon(;)

>>a=1;b=10;c=a +b

Ex:

>>>b=20

>>>print(b)

>>>print("\"Hello\"")

Difference between a Statement and an Expression

A statement is a complete line of code that performs some action, while an expression is any section of the code that evaluates to a value. Expressions can be combined —horizontally into larger expressions using operators, while statements can only be combined vertically by writing one after another, or with block constructs. Every expression can be used as a statement, but most statements cannot be used as expressions

TUPLE ASSIGNMENTS

Tuple Assignment means assigning a tuple value into another tuple.

Ex:

t=('Hello','hi')

>>>m,n=t

>>>print(m) → Hello

>>>print(n) → hi

>>>print(t) → Hello,hi

In order to interchange the values of the two tuples the following method is used.

>>>a=('1','4')

>>>b=('10','15')

>>>a,b=b,a

>>>print(a,b)

(('10','15'), ('1','4'))

COMMENTS

Comments are non-executable statements which explain what program does. There are two ways to represent a comment.

Single Line Comment

Begins with # hash symbol

Ex:

>>>print("Hello world") # prints the string

Multi Line Comment

Multi line comment begins with a double quote and a single quote and ends with the same

Ex:

>>>'''This is a multi line comment'''

OPERATORS:

Operators are the construct which can manipulate the value of operands.

Eg: 4+5=9

Where 4, 5, 9 are operand

+ is Addition Operator

= is Assignment Operator

Types of Operator:

1. Arithmetic Operator
2. Comparison Operator (or) Relational Operator
3. Assignment Operator
4. Logical Operator
5. Bitwise Operator
6. Membership Operator
7. Identity Operator

1. Arithmetic Operator

It provides some Arithmetic operators which perform some arithmetic operations

Consider the values of a=10, b=20 for the following table.

Operator	Meaning	Syntax	Description
+	Addition	a+b	It adds and gives the value 30
-	Subtraction	a-b	It subtracts and gives the value -10
*	Multiplication	a*b	It multiplies and gives the value 200
/	Division	a/b	It divides and gives the value 0.5
%	Modulo	a%b	It divides and return the remainder 0
**	Exponent	a**b	It performs the power and return 10^{20}
//	Floor	a//b	It divides and returns the least quotient

Example Program:

1.Write a Python Program with all arithmetic operators

>>>num1 = int(input('Enter First number: '))

```
>>>num2 = int(input('Enter Second number '))
>>>add = num1 + num2
>>>dif = num1 - num2
>>>mul = num1 * num2
>>>div = num1 / num2
>>>modulus = num1 % num2
>>>power = num1 ** num2
>>>floor_div = num1 // num2
>>>print('Sum of ',num1 ,'and' ,num2 ,'is :',add)
>>>print('Difference of ',num1 ,'and' ,num2 ,'is :',dif)
>>>print('Product of' ,num1 ,'and' ,num2 ,'is :',mul)
>>>print('Division of ',num1 ,'and' ,num2 ,'is :',div)
>>>print('Modulus of ',num1 ,'and' ,num2 ,'is :',modulus)
>>>print('Exponent of ',num1 ,'and' ,num2 ,'is :',power)
>>>print('Floor Division of ',num1 ,'and' ,num2 ,'is :',floor_div)
```

Output:

```
>>>
Enter First number: 10
Enter Second number 20
Sum of  10 and 20 is : 30
Difference of  10 and 20 is : -10
Product of 10 and 20 is : 200
Division of  10 and 20 is : 0.5
Modulus of  10 and 20 is : 10
Exponent of  10 and 20 is : 100000000000000000000
Floor Division of  10 and 20 is : 0
>>>
```

2. Comparison Operator (or) Relational Operator

These operators compare the values and it returns either True or False according to the condition. Consider the values of a=10, b=20 for the following table.

Operator	Syntax	Meaning	Description
==	a==b	Equal to	It returns false
!=	a!=b	Not Equal to	It returns true
>	a>b	Greater than	It returns false
<	a<b	Lesser than	It returns true
>=	a>=b	Greater than or Equal to	It returns false
<=	a<=b	Lesser than or Equal to	It returns true

3. Assignment Operator

Assignment operators are used to hold a value of an evaluated expression and used for assigning the value of right operand to the left operand.

Consider the values of a=10, b=20 for the following table.

Operator	Syntax	Meaning	Description
=	a=b	a=b	It assigns the value of b to a.
+=	a+=b	a=a+b	It adds the value of a and b and assign it to a.
- =	a-=b	a=a-b	It subtract the value of a and b and assign it to a.
*=	a*=b	a=a*b	It multiplies the value of a and b and assign it to a.
/=	a/=b	a=a/b	It divides the value of a and b and assign it to a.
%=	a%=b	a=a%b	It divides the value of a and b and assign the remainder to a.
**=	a**=b	a=a**b	It takes 'a' as base value and 'b' as its power and assign the answer to a.
//=	a//=b	a=a//b	It divides the value of a and b and takes the least quotient and assign it to a.

4. Logical Operator

Logical Operators are used to combine two or more condition and perform logical operations using Logical AND, Logical OR, Logical Not.

Consider the values of a=10, b=20 for the following table.

Operator	Example	Description
AND	if(a<b and a!=b)	Both Conditions are true
OR	if(a<b or a!=b)	Anyone of the condition should be true
NOT	not (a<b)	The condition returns true but not operator returns false

5. Bitwise Operator

Bitwise Operator works on bits and performs bit by bit operation.

Consider the values of a=60, b=13 for the following table.

Operator	Syntax	Example	Description
&	Binary AND	a&b= 12	It do the and operation between two operations
|	Binary OR	a|b= 61	It do the or operation between two operations
~	Binary Ones Complement	~a=61	It do the not operation between two operations
<<	Binary Left Shift	<<a	It do the left shift operation
>>	Binary Right Shift	>>a	It do the right shift operation

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

1. Write a Python Program with all Bitwise Operator

```
a = 10                                  # 10 = 0000 1010
b = 20                                  # 20 = 0001 0100
c = 0
c = a & b;                              # 0 = 0000 0000
print ("Line 1 - Value of c is ", c)
c = a | b;                              # 30 = 0001 1110
print ("Line 2 - Value of c is ", c)
c = ~a;                                 # -11 = 0000 1011
print ("Line 3 - Value of c is ", c)
c = a << 2;                             # 40 = 0011 1000
print ("Line 4 - Value of c is ", c)
c = a >> 2;                             # 2 = 0000 0010
print ("Line 5 - Value of c is ", c)
```

Output:

Line 1 - Value of c is 12

Line 2 - Value of c is 61

Line 3 - Value of c is -61

Line 4 - Value of c is 240

Line 5 - Value of c is 15

6. Membership Operator

Membership Operator test for membership in a sequence such as strings, lists or tuples.

Consider the values of a=10, b=[10,20,30,40,50] for the following table.

Operator	Syntax	Example	Description
in	value *in* String or List or Tuple	a in b returns True	If the value is '**in**' the list then it returns True, else False
not in	value *not in* String or List or Tuple	a not in b returns False	If the value is '**not in**' the list then it returns True, else False

Example:

```
x='python programming'
print('program' not in x)
```

print('program' in x)

print(' Program' in x)

Output:

False

True

False

7. Identity Operator

Identity Operators compare the memory locations of two objects.

Consider the values of a=10, b=20 for the following table.

Operator	Syntax	Example	Description
is	variable 1 *is* variable 2	a is b returns False	If the variable 1 value **is** pointed to the same object of variable 2 value then it returns True, else False
is not	variable 1 *is not* variable 2	a is not b returns False	If the variable 1 value **is not** pointed to the same object of variable 2 value then it returns True, else False

Example:

```
x1=7
y1=7
x2='welcome'
y2='Welcome'
print (x1 is y1)
print (x2 is y2)
print(x2 is not y2)
```

Output:

True

False

True

PRECEDENCE OF PYTHON OPERATORS

The combination of values, variables, operators and function calls is termed as an expression. Python interpreter can evaluate a valid expression. When an expression contains more than one operator, the order of evaluation depends on the **Precedence** of operations.

For example, Multiplication has higher precedence than Subtraction.

```
>>> 20 – 5*3
5
```

But we can change this order using Parentheses () as it has higher precedence.

```
>>> (20 - 5) *3
45
```

The operator precedence in Python are listed in the following table.

Table :Operator precedence rule in Python

S. No	Operators	Description
1.	()	Parentheses
2.	**	Exponent
3.	+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
4.	*, /, //, %	Multiplication, Division, Floor division, Modulus
5.	+, -	Addition, Subtraction
6.	<<, >>	Bitwise shift operators
7.	&	Bitwise AND
8.	^	Bitwise XOR
9.	|	Bitwise OR
10.	==, !=, >, >=, <, <=,	is, is not, in, not in Comparison, Identity, Membership operators
11.	not	Logical NOT
12.	and	Logical AND
13.	or	Logical OR

ASSOCIATIVITY OF PYTHON OPERATORS

If more than one operator exists in the same group. These operators have the same precedence. When two operators have the same precedence, associativity helps to determine which the order of operations. Associativity is the order in which an expression is evaluated that has multiple operator of the same precedence. Almost all the operators have left-to-right associativity. For example, multiplication and floor division have the same precedence. Hence, if both of them are present in an expression, left one evaluates first.

Example:

```
>>> 10 * 7 // 3
23
>>> 10 * (7//3)
20
>>> (10 * 7)//3
23
```

10 * 7 // 3 is equivalent to (10 * 7)//3.

Exponent operator ** has right-to-left associativity in Python.

```
>>> 5 ** 2 ** 3
390625
>>> (5** 2) **3
15625
>>> 5 **(2 **3)
390625
```

2 ** 3 ** 2 is equivalent to 2 ** (3 ** 2).

PRECEDANCE OF ARITHMETIC OPERATORS

Precedence	Operator	Description
1	**, ()	Exponent, Inside Parenthesis
2	/, *, %, //	Division, Multiplication, Modulo, Floor
3	+, -	Addition, Subtraction