Unit-2

Operators and Operands in Python:

Operators: Operators are special symbols that are used to manipulate the value of operands. They are applied on operand(s), which can be values or variables. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment.

Operands: Values and variables are known as operands.

Example:
result=a+b
(a, b operands and + is operator)

Python supports different types of operators which are as follows:
1) Arithmetic operators
2) Relational or Comparison operators
3) Logical Operators
4) Assignment Operators
5) Unary Operators
6) Bitwise Operators
7) Membership operators
8) Identity operators

1) Arithmetic operators

Some basic arithmetic operators are +, -, *, /, %, ** and //. You can apply these operators on numbers as well as on variables to perform corresponding operations.

Symbol/Operator	Description	Example 1	Example 2
+	Addition	>>>55+45 100	>>> “Good” + “Morning” GoodMorning
-	Subtraction	>>>55-45 10	>>>30-80 -50
*	Multiplication	>>>55*45 2475	>>> “Good”* 3 GoodGoodGood
/	Division	>>>17/5 3 >>>17/5.0 3.4 >>> 17.0/5 3.4	>>>28/3 9
%	Remainder/ Modulo	>>>17%5 2	>>> 23%2 1
**	Exponentiation	>>>2**3 8 >>>16**.5 4.0	>>>2**8 256
//	Integer Division	>>>7.0//2 3.0	>>>3/ / 2 1

2) Relational or Comparison operators

Comparison operators also known as relational operators are used to compare the values on its either sides and determines the relation between them. For example, assume a=100 and b = 200, we can use the comparison operators on them.

Symbol/Operator	Description	Example 1	Example 2
<	Less than	>>>7<10 True >>> 7<5 False	>>>”Hello"< "Goodbye" False >>>'Goodbye'< 'Hello' True

		>>> 7<10<15 True >>>7<10 and 10<15 True	
>	Greater than	>>>7>5 True >>>10<10 False	>>>"Hello"> "Goodbye" True >>>'Goodbye'> 'Hello' False

<=	less than equal to	>>> 2<=5
		True
		>>> 7<=4
		False
			>>>"Hello"<= "Goodbye"
			False
			>>>'Goodbye' <= 'Hello'
			True

>=	greater than equal to	>>>10>=10 True >>>10>=12 False	>>>"Hello">= "Goodbye" True >>>'Goodbye' >= 'Hello' False
! =	not equal to	>>>10!=11 True >>>10!=10 False	>>>"Hello"!= "HELLO" True >>> "Hello" != "Hello" False
==	equal to	>>>10==10 True >>>10==11 False	>>>"Hello" == "Hello" True >>>"Hello" == "Good Bye" False

3) Logical Operators

Python supports three logical operators logical and, logical or and logical not. Normally, the logical expressions are evaluated from left to right.

Symbol	Description
and (&)	If both the operands are true, then the condition becomes true. **Ex**: x and y
or (|)	If any one of the operand is true, then the condition becomes true. **Ex:** x or y
Not (!)	True if operand is false (complements the operand) **Ex:** not x

4) Assignment Operators

Assignment operators are used in Python to assign values to variables or operands. It is also known as shortcut operators that includes +=, -=, *=, /=, %=, //= and **= etc.

Symbol/Operator	Description	Example 1	Example 2
=	Assigned values from right side operands to left variable	x=12	c=a, assigns value of a to the variable c
+=	added and assign back the result to left operand	x+=2	x+=2 is same as x=x+2
-=	subtracted and assign back the result to left operand	x-=2	x-=2 is same as x=x-2
*=	multiplied and assign back the result to left operand	x*=2	x*=2 is same as x=x*2
/=	divided and assign back the result to left operand	x/=2	x/=2 is same as x=x/2
%=	taken modulus using two operands and assign the result to left operand	x%=2	x%=2 is same as x=x%2
**=	performed exponential (power) calculation on operators and assign value to the left operand	x**=2	x**=2 is same as x=x**2
//=	performed floor division on operators and assign value to the left operand	x / /= 2	x//=2 is same as x=x//2

5) Unary Operators

Unary operators act on single operands. Python supports Unary minus operator. An operand is preceded by a minus sign, the unary operator negates its value.

For example, if a number is positive, it becomes negative when preceded with a unary minus operator. Similarly, if the number is negative, it becomes positive after applying the unary minus operator.

Ex: b=10
       a = - (b)

The result of the expression is a = -10, because variable b has a positive value. After applying unary minus operator (-) on the operand b, the value becomes -10, which indicates it as a negative value.

6) Bitwise Operators

Bitwise operators perform operations at the bit level. These operators include bitwise AND, bitwise OR, bitwise XOR, and shift operators.

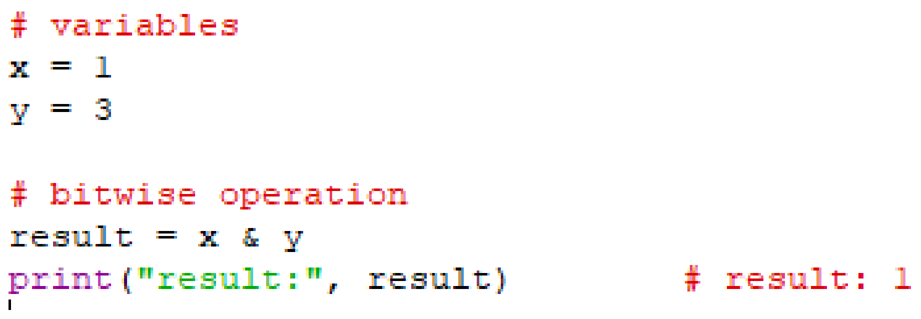
Bitwise AND (&)

Bitwise AND & will give 1 only if both the operands are 1 otherwise, 0.
The truth table for bitwise AND**.**

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

In the following example, we have two integer values 1 and 3 and we will perform bitwise AND operation and display the result.

Example:



```
# variables
x = 1
y = 3

# bitwise operation
result = x & y
print("result:", result)         # result: 1
```

Bitwise OR

Bitwise OR | will give 0 only if both the operands are 0 otherwise, 1.

The truth table for bitwise OR.

A	B	A|B
0	0	0
0	1	1

1	0	1
1	1	1

In the following example we have two integer values 1 and 2 and we will perform bitwise OR operation and display the result.

Example:

```
# variables
x = 1
y = 2

# bitwise operation
result = x | y

print("result:", result)        # result: 3
```

Bitwise XOR

Bitwise XOR ^ will give 1 for odd number of 1s otherwise, 0.

The truth table for bitwise XOR.

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

In the following example we have two integer values 2 and 7 and we will perform bitwise XOR operation and display the result. Example,

```
# variables
x = 2
y = 7

# bitwise operation
result = x ^ y

print("result:", result)        # result: 5
```

Shift Operators:

Python supports two bitwise shift operators. They are shift left (<<) and shift right (>>). These operations are used to shift bits to the left or to the right.

i) << Binary Left Shift: The left operands value is moved left by the number of bits specified by the right operand.

Example:

a=60

a<<2

output: 240

ii) >> Binary Right Shift: The left operands value is moved right by the number of bits specified by the right operand.

Example:

a=60

a >> 2

output: 15

7) Membership operators

Python supports two types of membership operators – in and not in. These operators used to test a value or variable is found in a sequence. (stings, list or tuple)

Symbol/Operator	Description	Example
In	True if value/variable is found in the sequence	>>>x=[1,2, 4,5,7,9] >>>5 in x True
not in	True if value/variable is not found in the sequence	>>>x=[1,2, 4,5,7,9] >>>10 not in x True

Example:

```
>>>a=”Hello World”
>>>print (‘H’ in a)
True
>>>print (‘m’ in a)
False
```

8) Identity operators

Python supports two types of identity operators. These operators compare the memory locations of two objects.

is and is not are the identity operators in Python. They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Symbol/Operator	Description	Example
is	True if the operands/values point(refer) to the same object	>>>x=5 >>>y=5 >>>x is y True
is not	True if the operands/values do not point(refer) to the same object	>>>a=[1,2, 4,5,7,9] >>>b=[1,2, 4,5,7,9] >>> x is not y True

Above given example, see that x and y are integers of same values, so they are equal as well as identical. But a and b are list. They are equal but not identical. Since list are mutable (can be changed), interpreter locates them separately in memory although they are equal.

Operators Precedence and Associativity

When an expression has more than one operator, then it is the relative priorities of the operators with respect to each other that determine the order in which the expression will be evaluated. The given table shows the lists of all operators from highest precedence to lowest.

Operator	Description
**	Exponentiation (raise to the power)
+ , -	unary plus and minus
* , /, %, //	Multiply, divide, modulo and floor division
+ , -	Addition and subtraction
>>,<<	Right and Left bitwise shift operators
&	Bitwise AND

<, <=, >, >=	Comparison operators
==, !=	Equality operators
% =, / =, // = , - =, + =, * =	Assignment operators
is, is not	Identity operators
in, not in	Membership operators
not, and, or	Logical operators

Operator precedence table is important as it affects how an expression is evaluated. For example

>>>10+3*5

160

This is because * has higher precedence than +, so it first multiplies 30 and 5 and then adds 10.

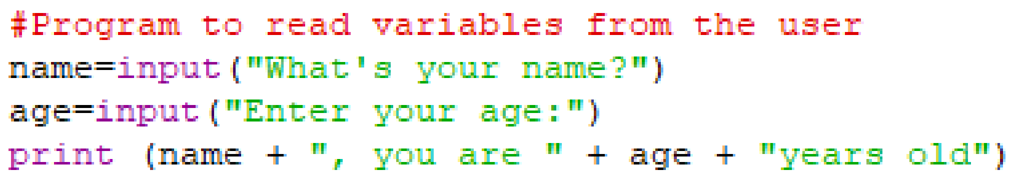
Input and Output:

Input Operation: The input() function prompts the user to provide(ask) some information on which the program can work and give the result. Always, the input function takes user's input as a string. So whether input a number or string, it is treated as a string only.

Example:

>>>x=input("Enter your name")

Enter your name: ABC



```
#Program to read variables from the user
name=input("What's your name?")
age=input("Enter your age:")
print (name + ", you are " + age + "years old")
```

Output:

What's your name? Ramesh

Enter your age: 20

Ramesh, you are 20 years old

Print Statement

Syntax: print expression/constant/variable

Print evaluates the expression before printing it on the monitor. Print statement outputs an entire (complete) line and then goes to next line for subsequent output (s). To print more than one item on a single line, comma (,) may be used.

Example

```
 >>> print “Hello”
 Hello
 >>> print 5.5
 5.5
 >>> print 4+6
 10
>>a=5
>>b=10
>>print (a,b)
```

Type Conversion or Type Casting

Convert a value from one data type to another data type known as type conversion or type casting. The conversion can be done explicitly (programmer specifies the conversions) or implicitly (Interpreter automatically converts the data type). The list of conversion built-in functions has given below.

Function	Description
int(x)	Converts x to an integer
long(x)	Converts x to a long integer
float(x)	Converts x to a floating point number
str(x)	Converts x to a string
tuple(x)	Converts x to a tuple
list(x)	Converts x to a list
set(x)	Converts x to a set

ord(x)	Converts a single character to its integer value
oct(x)	Converts an integer to an octal string
hex(x)	Converts an integer to a hexadecimal string
chr(x)	Converts an integer to a character
unichr(x)	Converts an integer to a Unicode character
dict(x)	Creates a dictionary if x forms a (key-value) pair

Errors

An error is a term used to describe any issue that arises unexpectedly that cause a computer to not function properly. There are three types of errors generally occur during running (execution) of a program. They are (i) Syntax error; (ii) Logical error; and (iii) Runtime error.

(i) **Syntax error:** Every programming language has its own rules and regulations (syntax). If we overcome the particular language rules and regulations, the syntax error will appear (i.e. an error of language resulting from code that does not conform to the syntax of the programming language). It can be recognized during compilation time.

Example

```
a = 0
while a < 10
        a = a + 1
        print a
```

In the above statement, the second line is not correct. Since the while statement does not end with ":". This will flash a syntax error.

(ii) **Logical error:** Programmer makes errors while writing program that is called "logical error". It is an error in a program's source code that results in incorrect or unexpected result. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. The logical error might only be noticed during runtime, because it is often hidden in the source code and are typically harder to find and debug.

```
a = 100
while a < 10:
        a = a + 1
        print a
```

In the above example, the while loop will not execute even a single time, because the initial value of "**a**" is 100.

(iii) **Runtime error:** A runtime error is an error that causes abnormal termination of program during running time. In general, the dividend is not a constant but might be a number typed by you at runtime. In this case, division by zero is illogical. Computers check for a "division by zero" error during program execution, so that you can get a "division by zero" error message at runtime, which will stop your program abnormally. This type of error is called runtime error.

Example

(a)

```
A=10
B=0
print (A/B)
```

Exercise

1. Write a program that asks two people for their names; stores the names in variables called name1 and name2; says hello to both of them.

2. Write a Python program which accepts the radius of a circle from the user and compute the area.

Sample Output :

```
r = 1.1
Area = 3.8013271108436504
```

3. Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them.

4. Write a Python program that will accept the base and height of a triangle and compute the area.

5. Write a Python program to solve (x + y) * (x + y).

Test Data : x = 4, y = 3

Expected Output : (4 + 3) ^ 2) = 49

6. Write a python program to sum of the first n positive integers.

7. Write a Python program to swap two variables.

8. Write a program to enter two integers and then perform all arithmetic operations on them.

9. Write a program to perform string concatenation.

10. Write a program to print the ASCII value of a character.

11. Write a program to swap two numbers without using temporary variable.

12. Write a program to calculate simple interest.

13. Write a program that prompts users to enter two integers x and y. The program then calculates and display x^y.

14. Write a program that calculates numbers of seconds in a day.

15. Write a program to calculate salary of an employee given his basic pay(to be entered by the user), HRA =10 percent of basic pay, TA=5 percent of basic pay. Define HRA and TA as constants and use them to calculate the salary of the employee.

16. Write a program to print the digit at one's place of Number.

17. Write a program to calculate average of two numbers. Print their deviation.

18. Write a program to covert a floating point number into the corresponding integer.

19. Write a program to convert a integer into the corresponding floating point number.

20. Write a program to calculate area of triangle using Heron's formula. (Hint: Heron's formula is : area = sqrt(s*(s-a)*(s-b)*(s-c))

Assignment-II

1. What is an operator? Explain all the operators available in Python.
2. What is the type conversion or Type casting? Explain type casting functions along with examples.
3. Define print statement and explain with example?
4. Define error?Explain types of erros with examples?