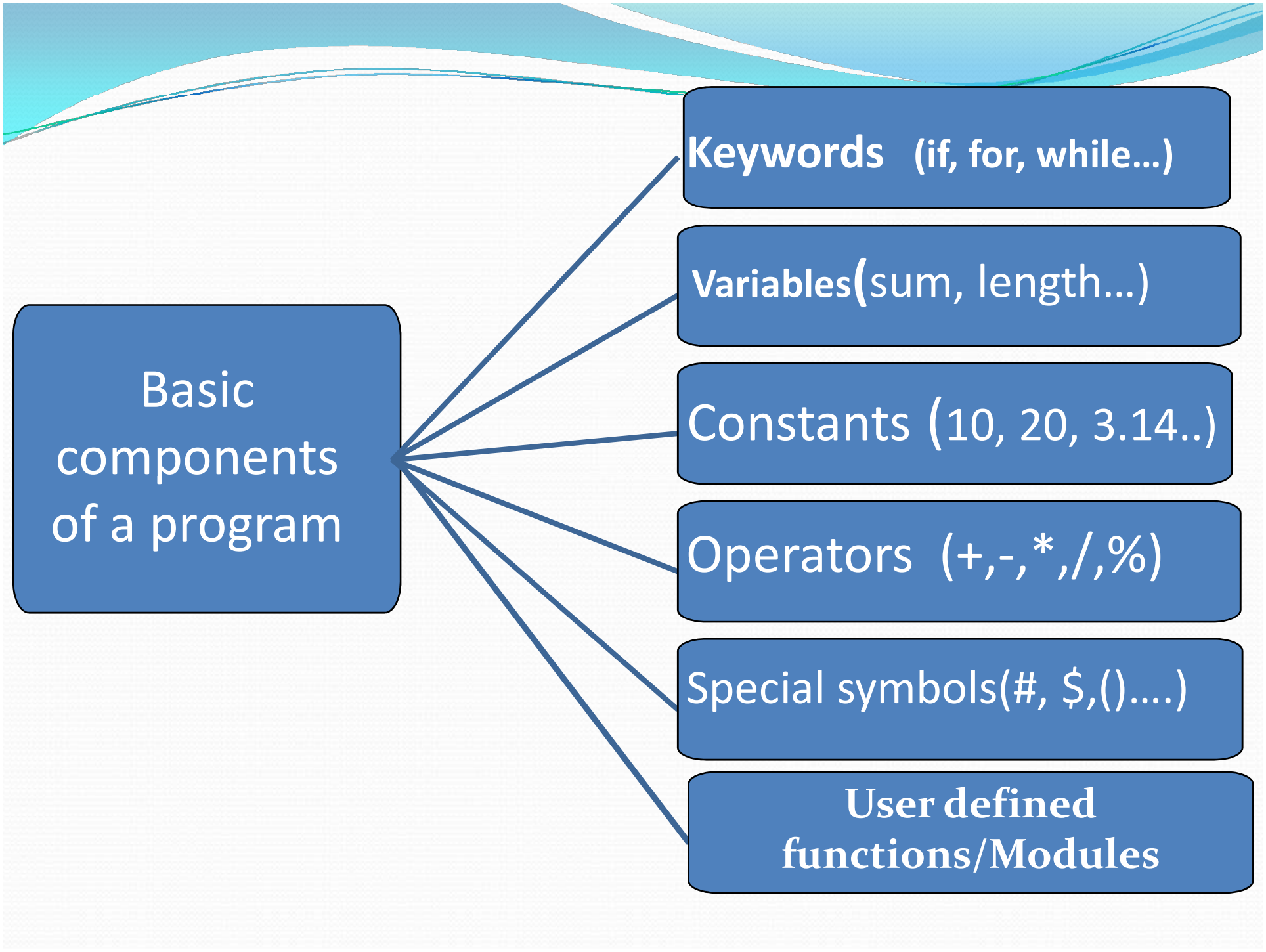
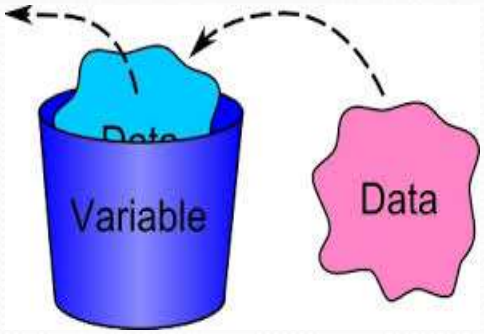

Basic components of a program
Keywords (if, for, while…)
Variables(sum, length…)
Constants (10, 20, 3.14..)
Operators (+,-,*,/,%)
Special symbols(#, $,()….)
User defined functions/Modules

Variable:- A quantity which changes during the program execution and it is used to store data is called variable.

Rules for declaring variables:



- It should not be a keyword.
- It should not begin with digit.
- It should not contain any blank space.
- It should not contain any special char such as $, #.....

Conventions

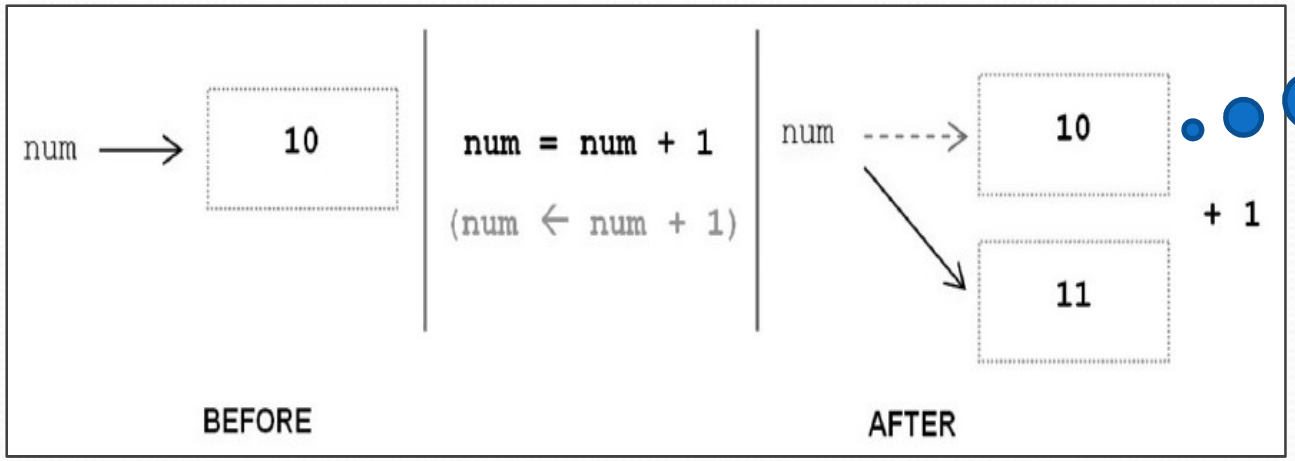
- Create a name that makes sense
- Use camelCase notation to declare a variable

  Example: myName myAge myAddress

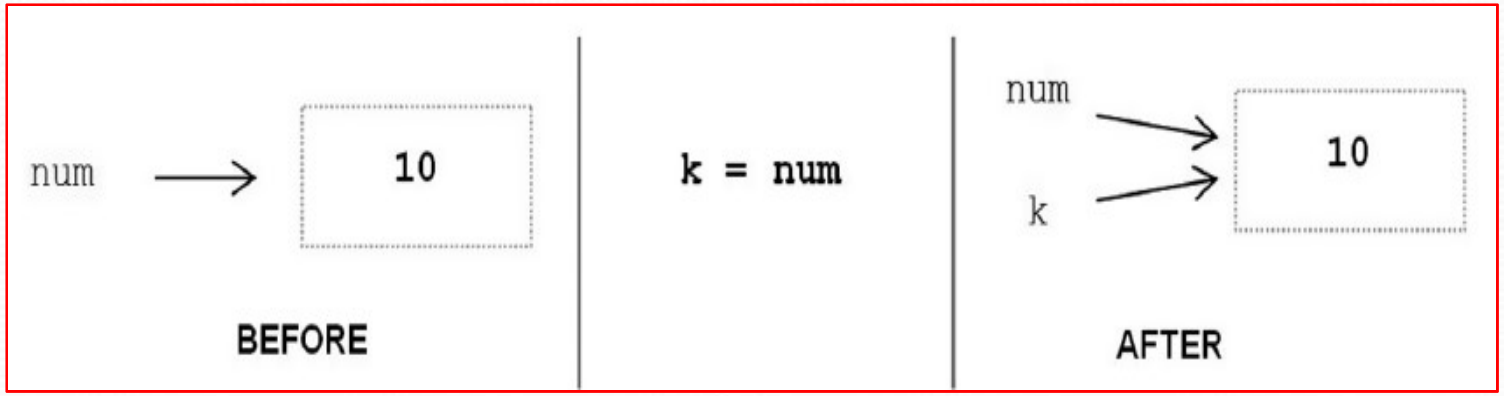
Variables

Num=10

Num → 10



immutable



Operators & Expressions

&,|,^,<<,>>,~

+ * - / %

< > == !=..

disc=b*b-4*a*c

a+=k

Big=a

Operators in Python

- Arithmetic operators(+,-,*,/,%,//)
- Relational operators(<,<=,>,>=,!=.==)
- Logical operators(and,or,not)
- Assignment operators(=,+=,-=.....)
- Bitwise operators(&,|,^,<<,>>,~)
- Special operators
  - Identity operators
  - Membership operators

Arithmetic operators(+,-,*,/,%,//)

operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Float Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	integer division	x // y

// is also called as truncation division

Arithmetic operators : used to perform arithmetic operations such as addition, subtraction, multiplication, division etc.

Example:

a=10, b=20;

c=a+b;

2. num=123

rem = num%10;

What is the value of rem?

rem=3;

3. num=123

num=num/10

value of num is ???

num =12.3

Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	Float division
%	modulus

Python Comparison Operators/ **Relational operators**

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

Return value of a relational operator is true or False

2<3 → true
3>4 → false

Logical operators :

Used to combine two or more relational expressions.

```
a=10;b=20;c=30
if ( a>b and a>c):
    print(a, "is largest")
elif ( b>a  and  b>c):
    print(b, "is largest")
else:
    print(c ,"is largest")
```

Operator	Meaning
and	Logical AND
or	Logical OR
not	Logical NOT

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
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Assignment operator s:

Multiple assignments:

```
a=b=c=10
print(a,b,c)
```

Binary numbers and bitwise operators

BITWISE OPERATORS	
&	AND
|	OR
^	Exclusive OR(XOR)
~	One's Complement (NOT)
>>	Shift right
<<	Shift left

Bitwise operator :

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
XOR	Bitwise Exclusive OR
<<	Left-shift
>>	Right-shift
~	one's compliment ;

~ => one's compliment ; change 0 to 1 and 1 to 0

1). Bitwise AND (&)

1 0 1

1 0 1

---

1 0 1

2). Bitwise OR (|)

1 0 0

1 0 1

---

1 0 1

3) Bitwise XOR(^)

If both bits are different then 1 else 0.

1 0 1

0 1 0

---

1 1 1

Programs on bitwise operators

1)Program to demonstrate working of bitwise operators

2)Program to check given number is even or odd using bitwise operator(without using % operator)

3) Given a set of numbers where all elements occur even number of times except one number, find the odd occurring number"

Program to demonstrate working of bitwise operators

```
// Bitwise AND
a=12
b = 25
print(a&b)
Output is 8
```

```
//Bitwise  OR

a=12

b = 25

print(a|b)
```

```
//Bitwise XOR

a=12

b = 25

print(a^b) Output = 21
```

```
a = 5 # 0101
b = 6 # 0110
print ("a & b ", a & b) # 0100  => 4
print ("a | b ", a | b) # 0111 => 7
print ("a ^ b ", a ^ b) # 0011 => 3
print ("a << 4 ", a << 4) # 0101 0000 => 80
print ("75 >> 3 ", 75 >> 3) # 0100 1011 >> 3 => 0100 1 => 9
print ( "~ a ", ~a )  #  -6  , This is the same as -x - 1.
```

//program to check even or odd number using bitwise operator

```
num=int(input("Enter a number: "))

if((num&1)==1):

    print("odd number", num);

else:

    printf("even number", num);
```

Given a set of numbers where all elements occur even number of times except one number, find the odd occurring number

a= [12, 12, 14, 90, 14, 14, 14];

Ans=90

a=[12, 12, 14, 90,90, 14, 14, 14,11]

```
res=0
for i in a:
    res=res^I
print(res)
```

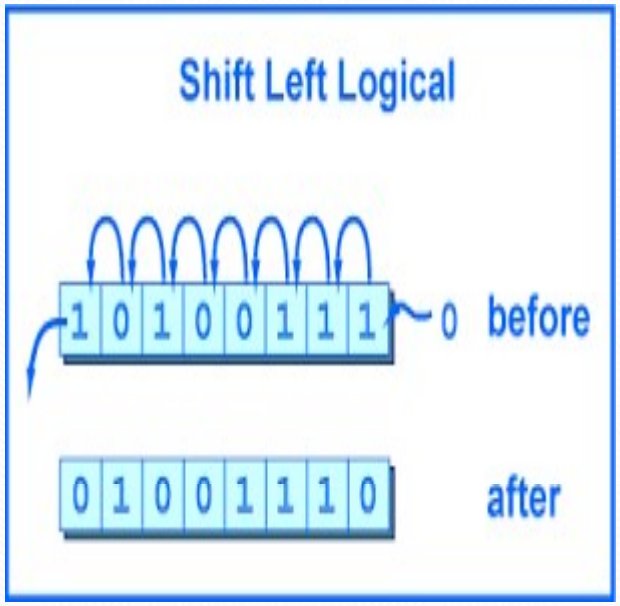
4) Left-shift(increase the value)

a=2

a=a<<1

0	0	0	0	0	0	1	0

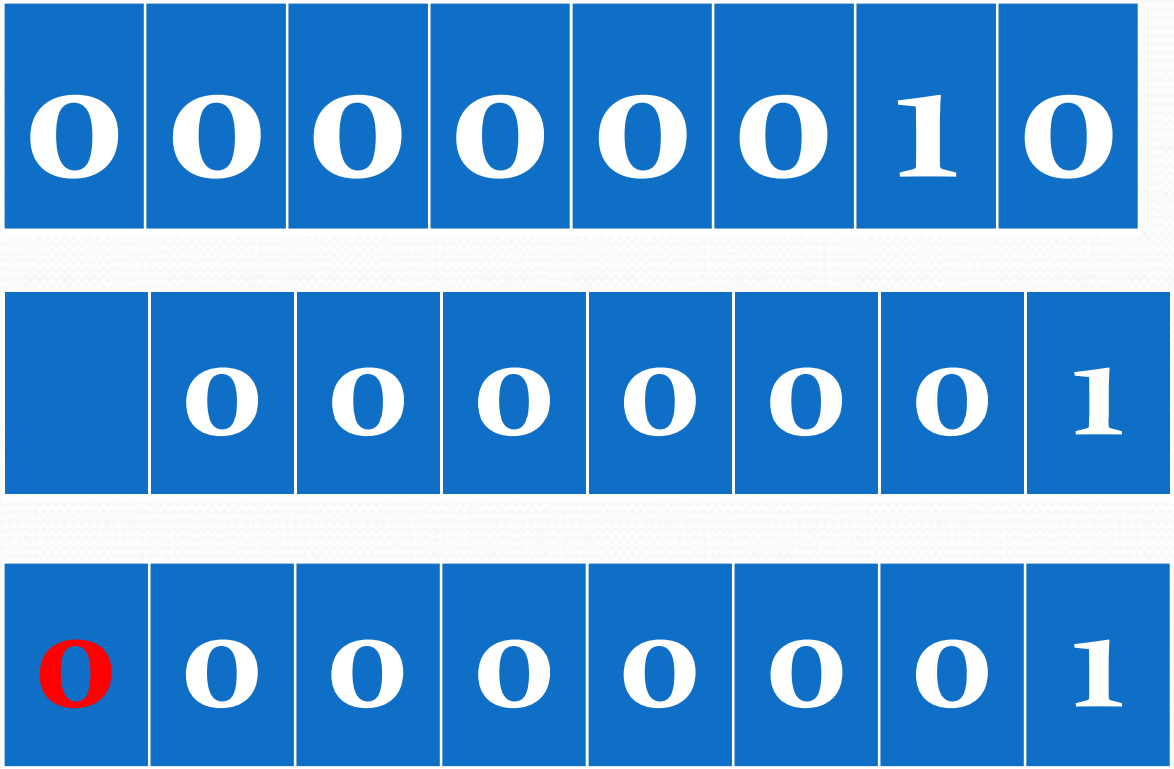
0	0	0	0	0	1	0	0



0

5) Right-shift (decreases the value)

a=2 b=a>>1



Swapping two numbers

```
a=4
b=7
print(a,b)
temp=a
a=b
b=temp
print(a,b)
```

```
a=3
b=5
print(a,b)
a = a + b
b = a - b
a = a - b
print(a,b)
```

```
a=7
b=8
print(a,b)
a,b=b,a
print(a,b)
```

```
a = 5
b = 6
print(a,b)
a = a ^ b  # 0101 ^ 0110 => 0011 => 3
b = a ^ b  # 0011 ^ 0110 => 0101 => 5
a = a ^ b  # 0011 ^ 0101 => 0110 => 6
print(a,b)
```

Special operators

- Identity operators

  identity operators compare the memory locations of two objects

  a=10

  b=10

  Id(a) is same as id(b), so they have same identity

- Membership operators

  They are used to determine if a particular value occurs within a list of values.

  (10,20,30,40,50,60,70)

  Is number 10 part of this list

Identity operators

identity operators compare the memory locations of two objects

operator	Description	Example
is	Returns true if both variables are of same identity, false otherwise. That is True if both the operands are identical (refer to the same object, same mem location)	x is y
is not	True if the operands are not identical	x is not y

Example

```
a = 20
b = 20

if ( a is b ):
   print ("Line 1 - a and b have same identity")
else:
   print ("Line 1 - a and b do not have same identity")

if ( id(a) == id(b) ):
   print ("Line 2 - a and b have same identity")
else:
   print ("Line 2 - a and b do not have same identity")

b = 30
if ( a is b ):
   print ("Line 3 - a and b have same identity")
else:
   print ("Line 3 - a and b do not have same identity")

if ( a is not b ):
   print ("Line 4 - a and b do not have same identity")
else:
   print ("Line 4 - a and b have same identity")
```

```
a=10
b=10
print(id(a),id(b))

c=a

print(id(a),id(b),id(c))

c=b

print(id(a),id(b),id(c))

c=100

print(id(a),id(b),id(c))
```

Example:

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is z)

# returns True because z is the same object as x

print(x is y)

# returns False because x is not the same object as y, even if they have thew same content

print(x == y)

# to demonstrate the difference betweeen "is" and "==": this comparison returns True
    because x is equal to y
```

Membership operators

Membership operators

in

not in

They are used to check whether a value is found in a sequence (string, list, tuple, set and dictionary).

operator	Meaning	Example
in	**True if value/variable is found in the sequence**	**5 in x**
not in	**True if value/variable is not found in the sequence**	**5 not in x**

To search an item in the list (in operator)

```
studlist=[
            "acharya","anusha","ayushi","akshaya",
            "anant","pavan","kumar","kishore“,
             "charan","bhumika","anirudh","sujay",
           ]

key="pavan"
if(key in studlist):
   print("yes name found")
else:
   print("name not found")
```

Membership operators

Membership Operators	Examples	Result
in	10 in (10, 20, 30)	True
	red in ('red','green','blue')	True
not in	10 not in (10, 20, 30)	False

operators and polymorphism:

Some operators behave differently based on the type of the operands. They exhibit different forms. These are said to be polymorphic.

operator + on numbers is addition operator.

operator + on strings, tuples, lists is concatenation operator - juxtapose the two items.

operator * on numbers is multiplication operator.

Operator * on strings, tuples, lists with an integer is replication operator - repeat the elements # of times.

Also, observe that the operator remains commutative even if the operands are switched.

Polymorphic operators

```
print(10 + 20)
print("one" + "two")
print([10, 20] + [30, 40])

# * is replication operator
print(2 * 3)
print("2" * 3)
print("python" * 3)
print((10, 20) * 3)
print(3 * "2")
```

Polymorphic operators

```
# polymorphic operator
# +
print(10 + 20) # 30
print("one" + "two") # onetwo # concatenation
print([10, 20] + [30, 40]) # [10, 20, 30, 40] # concatenation

# * is replication operator
print(2 * 3) # 6
print("2" * 3) # 222 # replicate
print("python" * 3) # pythonpythonpython
print((10, 20) * 3)  #(10, 20, 10, 20, 10, 20)
print(3 * "2") # 33 # commutative.
```

Python **Expression:**

An expression has a value.

Something that can be evaluated is called expression

Examples for expressions.

- Binary Expressions
- 3 + 4
- Compound Expressions: expression within parentheses

(3 + 4)*(4_2)

sqrt(s*(s-a)*(s-b)*(s-c))

-Unary Expressions

-(2 * 5)

Conversion of expressions :

Mathematical expression	Python - expression
$\frac{a}{b}$	a/b
$S = \frac{a+b+c}{2}$	S= (a+b+c)/2
$S = a + b + \frac{c}{2}$	S=a+b+c/2
$ax^2+bx+c=0$	A*x*x+b*x+c=0
area= $\sqrt{s(s-a)(s-b)(s-c)}$	area=sqrt(s*(s-a)*(s-b)*(s-c))

Arithmetic Expressions

Mathematical expression	Python expression
axb-c	a*b-c
(m+n)(x+y)	(m+n)*(x+y)
$\left[\frac{ab}{c}\right]$	a*b/c
$3x^2+2x+1$	3*x*x+2*x+1
$\frac{a}{b}$	a/b
S= $\frac{a+b+c}{2}$	S=(a+b+c)/2

Functions

Built in

sqrt(),sin(),max(),int(),....

User defined

```
def FindSimpleInterest(p,t,r):
    return (p*t*r)/100
def AreaofRectangle(l,b):
    return (l*b)
```

Mathematical functions in math module

function	Description
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
exp(x)	Returns e**x
log2(x)	Returns the base-2 logarithm of x
log10(x)	Returns the base-10 logarithm of x
pow(x, y)	Returns x raised to the power y
sqrt(x)	Returns the square root of x
cos(x)	Returns the cosine of x (x in radians)
sin(x)	Returns the sine of x(x in radians)
tan(x)	Returns the tangent of x (x in radians)
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians
ceil(x)	Returns smallest integer not less than x.
floor(x)	Returns the largest integer less than or equal to x

Math functions

```
import math
print(math.sin(math.radians(0)))
print(math.cos(math.radians(0)))
math.sin(math.radians(90))
res=math.cos(math.radians(90))
print(round(res,1))
```

Math functions

```
import math   # This will import math module
print("math.ceil(100.12) : ", math.ceil(100.12))
print ("math.ceil(100.72) : ", math.ceil(100.72))
print ("math.ceil(math.pi) : ", math.ceil(math.pi))
print ("math.ceil(-45.17) : ", math.ceil(-45.17))
```

```
import math   # This will import math module
print ("math.floor(100.12) : ", math.floor(100.12))
print ("math.floor(100.72) : ", math.floor(100.72))
print ("math.floor(math.pi) : ", math.floor(math.pi))
print("math.floor(-45.17) : ", math.floor(-45.17))
```

Evaluating expressions

```
sqrt(s*(s-a)*(s-b)*(s-c))
```

```
a+b*c
```

```
X = 2+9*((3*12)-8)/10
```

```
x1=(-b+ math.sqrt (b*b-4*a*c))/(2*a)
```

Precedence

Priority of operators is called Precedence (Order of evaluation)

Example:

Multiplication has more precedence than addition.

Example: Let us take a=2 b=3 c= 4

a+b*c

What is the answer ? 20 or 14 ?

2+3*4 = 20 (is it this way 5*4=20)

OR

2+3*4=14 (this way 2+12=14)

Correct answer is 14 , because * has high priority than +

So first 4*3 =12 and then 2+12 =14

This order of evaluation of operators is called precedence

High Precedence

Low Precedence

operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Example

Evaluate :

@ a=1, b=-5, c=6

x1=(-b+ math.sqrt (b*b-4*a*c))/(2*a)

=(-(-5)+sqrt((-5*-5-4*1*6))/(2*1)

=(5 + sqrt(-5*-5-4*1*6))/(2*1)

=(5 + sqrt(25 -4*1*6))/(2*1)

=(5 + sqrt(25 -4*6))/(2*1)

=(5 + sqrt(25 -24))/(2*1)

=(5 + sqrt(1))/(2*1)

=(5 + 1.0)/(2*1)

=(6.0)/(2*1)

=6.0/2 = 3.0

Associativity

```
res=100/10*10;

printf(res);
```

```
res=100*10/10%3;

print(res);
```

```
res=100/10/10;

print(res);
```

```
res=100+10/10;

print(res);
```

Associativity

- Ans=100/10*10
- What is the answer ?
- 100 or 1
- It indicates the order in which two operators of same precedence are evaluated
- Left to right or Right to left
- (Which part of the expression is executed first)

Associativity

Operators	Description	Associativity
`**`	Exponentiation	**Right to left**
`+x`, `-x`, `~x`	Positive, negative, bitwise not	left to right
`*`, `/`, `//`, `%`	Multiplication, float division, integer division, remainder	left to right
`+`, `-`	Addition, subtraction	left to right
`<<`, `>>`	Bitwise left, right shifts	left to right
`&` `|`	Bitwise and ,Bitwise or	left to right
`in`, `not in`, `is`, `is not`, `<`, `>`, `>=`, `!=`, `==`	Comparision, membership and identity tests	left to right
`and`,`not`,`or`	Boolean AND,NOT OR	left to right

For Example:

a=1

b=2

c=3

d=4

res=a>b+c&d

How to evaluate this exp ?

Rules for evaluation of expression

1. First parenthesized sub expression from left to right are evaluated.
2. If parentheses are nested, the evaluation begins with the innermost sub expression
3. The precedence rule is applied in determining the order of application of operators in evaluating sub expressions
4. The associatively rule is applied when 2 or more operators of the same precedence level appear in a sub expression.
5. Arithmetic expressions are evaluated from left to right using the rules of precedence
6. When parentheses are used, the expressions within parentheses assume highest priority

Evaluate exp

res=3+5+(1+1)**2**2*3<<1>>1+3-1

Print(res)

Evaluate exps

```
res=3-5*(1+5)**4**5%4&3<6%4*(6+4)%4
Print(res)

x = 2+9*((3*12)-8)/10
Print(x)
```

Arity or rank

Refers to the number of operands required for the operator could be 1 or 2 or 3

Precedence:

Priority of operators is called Precedence (Order of evaluation)

Example:

Multiplication has more precedence than addition.

Associativity:

It indicates the order in which two operators of same precedence are evaluated

Left to right or Right to left

(Which part of the expression is executed first)

Mixed-Type Expressions

A **mixed-type expression** **is an expression containing operands of different type**. The CPU can only perform operations on values with the same internal representation scheme, and thus only on operands of the same type. **Operands of mixed-type expressions therefore must be converted to a common type**. **Values can be converted in one of two ways**—by implicit (automatic) conversion, called ***coercion***, or by explicit ***type conversion***.

Type casting :

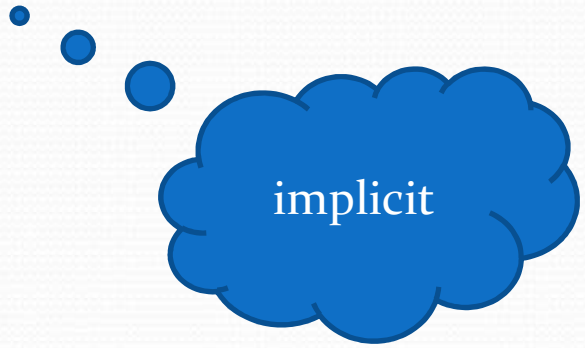
Process of changing the cast of variable or data is called type casting.

Coercion is the *implicit* (automatic) conversion of operands to a common type. **Coercion is automatically performed on mixed-type expressions only if the operands can be safely converted**, that is, if no loss of information will result.

The conversion of integer 2 to floating-point 2.0 below is a safe conversion—the conversion of 4.5 to integer 4 is not, since the decimal digit would be lost,

2 + 4.5 → 2.0 + 4.5 → **6.5** **safe** (automatic conversion of int to float)

int float float float float



Type conversion is the *explicit* conversion of operands to a specific type. **Type conversion can be applied even if loss of information results**. Python provides built-in **type conversion functions** `int()` and `float()`, with the `int()` function truncating results

2 + 4.5 → float(2) + 4.5 → 2.0 + 4.5 → **6.5** **No loss of information**

int float float float float float float

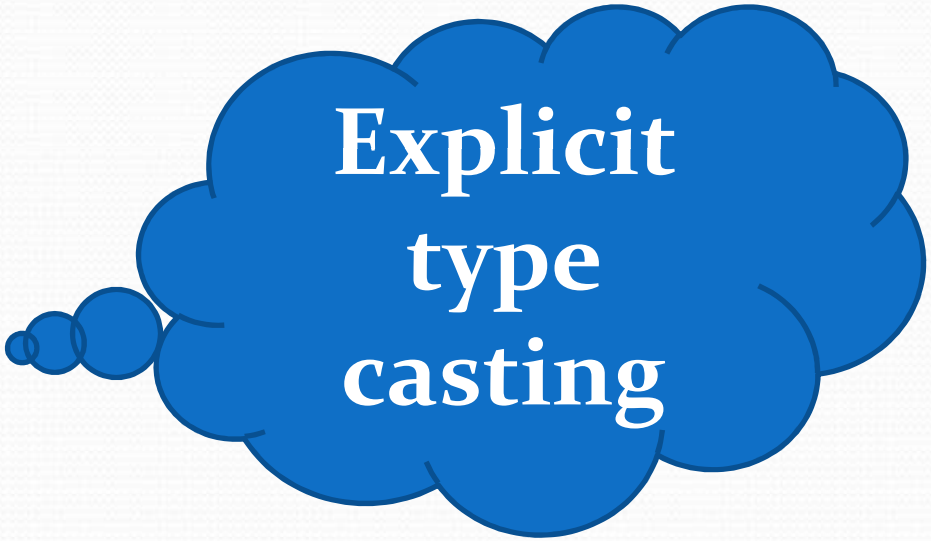
2 + 4.5 → 2 + int(4.5) → 2 + 4 → **6** **Loss of information**

int float int int int int int

print(“4”+5) ??

Print(int(“4”)+5)??

Print(“4”+str(5))



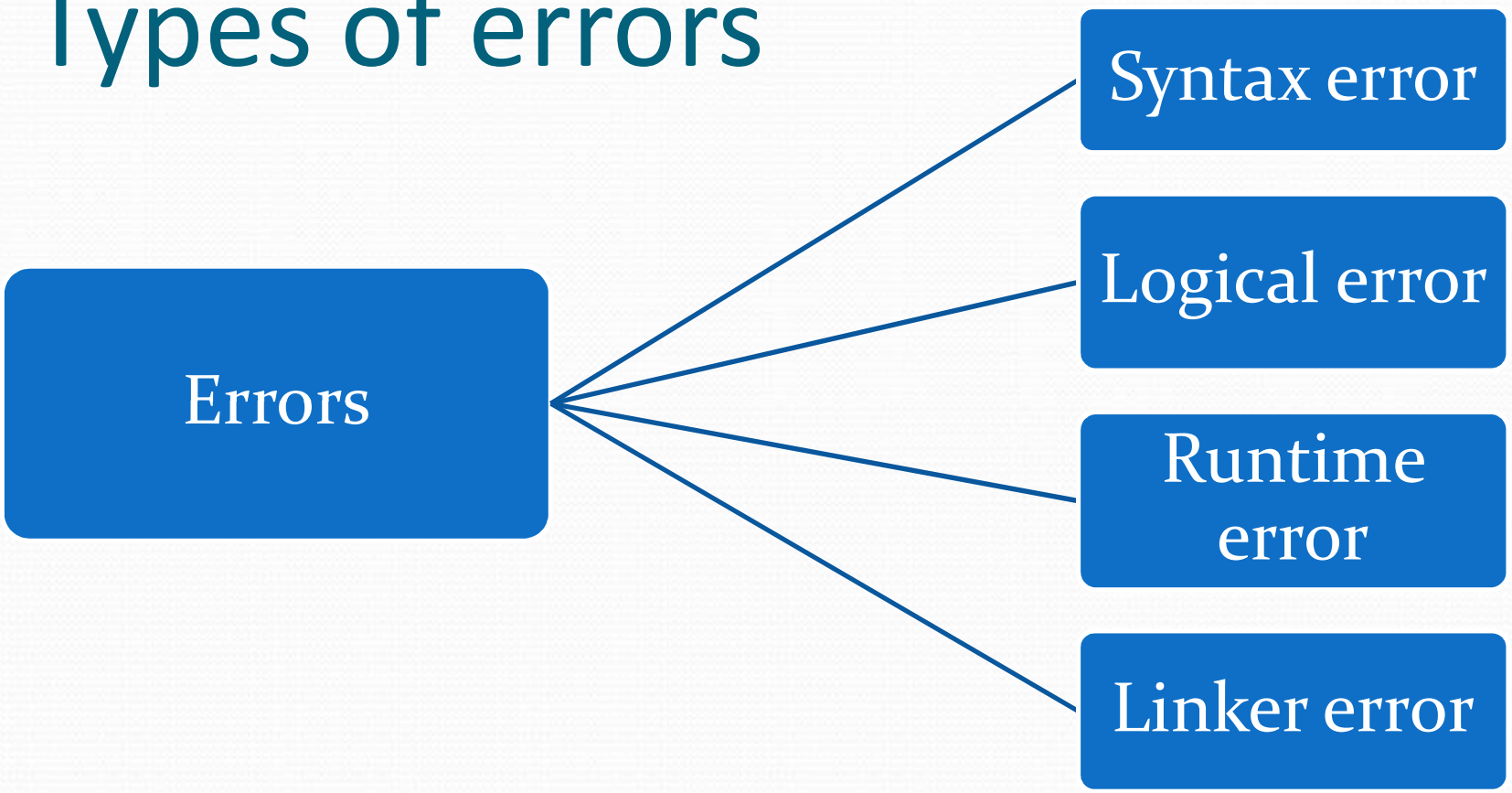
Type conversion functions int() and float()

Conversion Function		Converted Result		Conversion Function		Converted Result
`int()`	`int(10.8)`	10		`float()`	`float(10)`	10.0
	`int('10')`	10			`float('10')`	10.0
	`int('10.8')`	ERROR			`float('10.8')`	10.8

Note that numeric strings can also be converted to a numeric type. In fact, we have already been doing this when using `int` or `float` with the input function,

```
num_credits =  int(input('How many credits do you have? '))
```

Types of errors



1. Syntax error :-

   print(“hello)

   how to remove

   By compiling

2. Logical error :-

   zero errors,it takes Input but gives wrong output.

   avg=a+b+c/3; // error in Logic

   how to remove :

   By debugging and tracing vaues

3. Runtime error

zero errors, zero warnings, takes input but no output because error happens at run time

Example:

```
a=int(input("enter the value of a"))
b=int(input("enter the value of b"))
res=a/b
print(res)
```

How to remove?

By debugging and Exception handling.