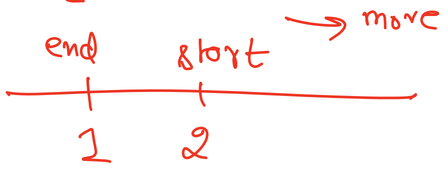
a,b = 10

In slicing, first see if its possible to reach the end from start, if not output ''

0 1 2 3 4 5

P[2:2]



as above is not possible

output ''

a=10

a , b = 10, 20 $\longrightarrow$ shortcut for multiple values

a = b = c = 20

a = b = 10 ✓

a , b = 10 ✗

a , b , c = 10 , 20 , 30 , 40 ✗

$\boxed{\text{error as python expects 3 values only.}}$

=

+ =
- =
/ =
* =
% =
** =
& =
| =
// =
^ =
>> =
<< =

$x = x + 5$

$x += 5$

c = 10
c ++ ⇒ This is error

no post increment in python

pre - increment

+(+10)

⇒ 10

post increment

$\underline{10++}$

we get error as syntax is wrong

$\boxed{\text{a++5 = a+5 but that doesn't change value of a}}$

$j = +10$

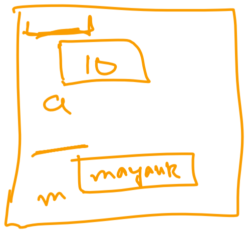
$-(-10) \Rightarrow 10$

$--j \Rightarrow 10$

2 places
1. To check if 2 references point to the same memory location

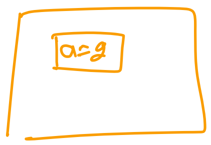
2. To determine whether a value is of certain class or type.

1.



a=10 id(a)

m=mayank id(m)



Python does some optimization and stores smaller in same location/address

★★ is operator

$\Rightarrow$ returns True if object location same

is not operator

$\Rightarrow$ returns False if operands are not identical

It is used to check whether a value or variable is part of a seq (string
list
tuple)
set
and dictionary.

Precedence	Operator	Description	Associativity
1	**	Exponentiation	Right to left
2	+x, -x	Positive, negative	Right to left
3	*, /, %, //	Multiplication, division, modulus, floor division	Left to right
4	+, -	Addition, subtraction	Left to right
5	<<, >>	Bitwise shift left, bitwise shift right	Left to right
6	&	Bitwise AND	Left to right
7	^	Bitwise XOR	Left to right
8	|	Bitwise OR	Left to right
9	<, <=, >, >=	Comparison operators	Left to right
10	==, !=	Equality operators	Left to right
11	not	Logical NOT	Right to left
12	and	Logical AND	Left to right
13	or	Logical OR	Left to right
14	if-else	Ternary conditional	Right to left
15	=, +=, -=, *=, /=, //=, %=, **=, <<=, >>=, &=, ^=, |=	Assignment and compound assignment	Right to left

x^

+ , - sign

Normal ope

+/- op

C++ , Golang

%d ⇒ int
%i ⇒ int
%f ⇒ float
%s ⇒ string

```
a=10
print("%s" %a)
```
#str

Output:

10

```
a=10
print("%f" %a)
```

Output:

10.000000

```
a=10.6
print("%f" %a)
```
Output:

10.600000

```
a=10.6
print("%.2f" %a)
```
Output: decimal points

10.60

```
a=10.6
print("%d" %a)
```
int

Output:

10

```
a=10.6
print("%s" %a)
```
Output:

10.6

```
a=True
print("%s" %a)
```
Output:

True

```
a=True
print("%d" %a)
```
/ 1

Output:

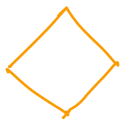
1

Gets rid of your % operator and makes
the string formatting more regular

decision control statements

★ In real life too, we have inherent decision flow.

```
If dist > 2 km
        ____ transport
  else
        ____ walk
```



input ⇒ If on basis of this input, you wanna do something, then decision control statem. will use

4 decision control :

1. if
2. if else
3. if elif else
4. nested if

: ← ☆☆

1. Python uses indentation to divide/identify code block. It doesn't use { }
2. : is imp ( ) around condⁿ are optional

```
if (condn) :
    Statement 1
    Statement 2
else :
    Statement 3
    Statement 4
ended ________________
Statement 5
```

```
[58]: stringInput = input()

       ga'
```

```
[59]: if stringInput == "apple":
          print("apple")
      elif stringInput == "orange":
          print("orange")
      elif stringInput == "banana":
          print("banana")
      else:
          print("no fruit")

      no fruit
```

we can have multiple elifs

we can choose to have else or not.

```python
# Previous Function
def out2(stringInput,num):
    if stringInput  == "apple":
        if num == 2:
            print("100")
        elif num == 3:
            print ("200")
    elif stringInput == "orange":
        if num == 5:
            print("500")
        elif num == 8:
            print ("600")
```

```python
def out(stringInput,num):
    if stringInput  == "apple":
        if num == 2:
            print("100")
        elif num == 3:
            print ("200")
    elif stringInput == "orange":
        if num == 5:
            print("500")
        elif num == 8:
            print ("600")
        else:
            print("1000")
    else:
        print("2000")
```