Python

print() → To print the content.

Variable :

a = 10
b = 20

print (a, b) // O/P: 10 20.

print (id(a), id(b)) // O/P: 140319838
1308 34689.

↓
memory address.

Rules to create variables :

- Name starts with uppercase / Lowercase/ underscore.
- can't start with number
- No length restriction.
- Reserved words are not allowed.
- case sensitive.
- Reassign is allowed.

Operators :

i) Arithmetic operators :

a = 20
b = 5

print (a + b) // 25
print (a - b) // 15
print (a/b) // 4.0.
print (a % b) // 0 (Remainder)
print (a **b) // 3200000.
(20 * 20 * 20 X 20 X20 (5 times)

print(a//b) // 4.
(Floor division).

ii) Assignment operator (= += -=)

```
x=5
print(x)   // 5

x+=7
print(x)   // 12

x-=8
print(x)   // 4.
```

iii) comparison operator
(==, !=, >, <, >=, <=)

```
a = 8
b = 10
a == b    // False.
a != b    // True.
a > b     // False
a < b     // True.
a >= b    // False.
a <= b    // True.
```

iv) Logical operator (and, or, not).

```
a = 5     x = 10.
a < 8   and   a > 6     // False
a == 5  and   a < 6     // False.
a < 8   or    a > 10.   // True.
x != a    //   True.
not x != a.   //   False.
```

Membership operator (in, not in).

```
X = 'welcome'
print ('w' in x)        // True.
print ('L' not in x)    // False.
```

Identity operator (is, isnot).

```
x=10      y=10
x is y        // True
x == y        // True.
x is not y    // False.
```

Bitwise operator. (and, or, xor).

Truth table :

A	B	A & B	A | B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1		0

```
x=10      y=8
print (x & y)     // 8
print (x | y)     // 10
print (x ^ y)     // 2.
```

Data types.

- Numeric
  - Int
  - Float
  - complex No.
- sequence type
  - string
  - List (Mutable)
  - Tuple
- Dictionary (Mutable)
- Set

Numeric Datatype :

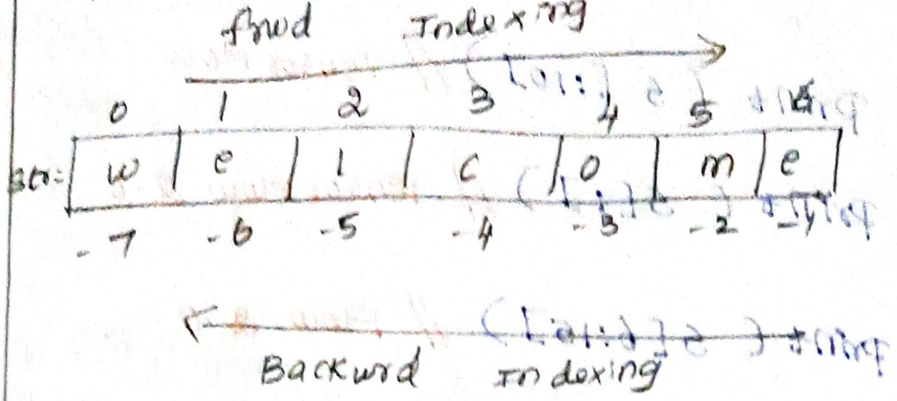
```
a = 10      type(a)     // Int
a = 2.5     type(a)     // Float
a = 1+2j    type(a)     // complex.
```

Sequence type :

a) String : # ' ' , " " , ''' '''

```
x = 'welcome'            type(x)     // str
x = "welcome"            type(x)     // str
x = '''  welcome
         Home '''        type(x)     // str

x = "welcome"
x = x * 5   // welcome welcome welcome
               welcome welcome.
```



str [0] // 'w' {1st character}

str [-7] // 'w'

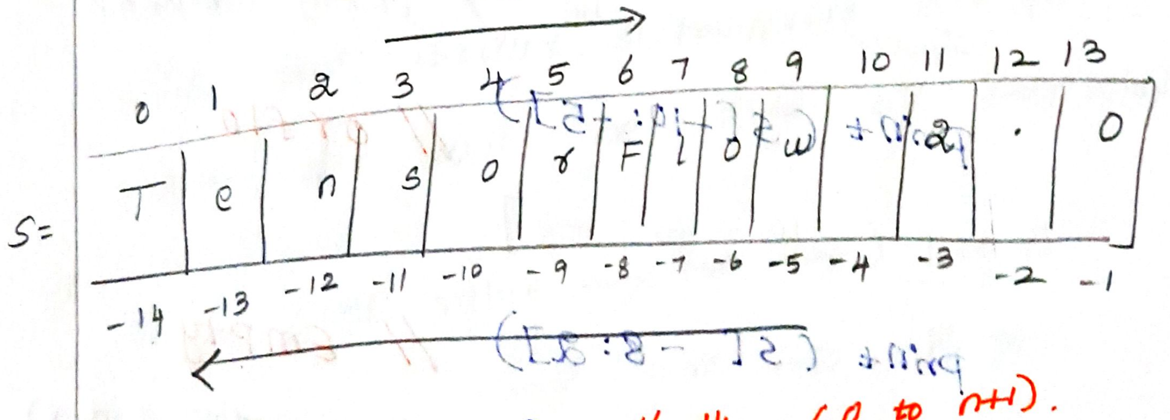
str [0:5] // 'welco'.
{Fetch all char from 0 to 5]

str [:4] // 'welc'
{Fetch 1st 4 char}.

str [-4:] // 'come'
{ Fetch last 4 char}.

String slicing.

S = "TensorFlow 2.0



print (len(s)) // 14. (0 to n+1).

print (s) // TensorFlow 2.0.

print (s [11:]) // 2.0. (From 11th Index to end).

print (s [:10]) // Tensor Flow (Before last Index)

print (s[:]) // TensorFlow 2.0 (whole string)

print (s[6:15]) // Flow 2.0.
(From 6th Index to last Index-1)

print (s[6:13 : 2]) // F0
(From 6th Index to last Index -1 ie, 12th Index with step count 1).

print(s[3:10 :4]) // s l
(From 3th Index to 9th Index, step count = 3)

print(s[-10 :]) // or flow: 2.0.

print (s[: -16]) // Tensor flow 2.0.

print (s[3:-7]) // sorf.
(In this case, starting Index must be > ending Index)

print (s[-10: -5]) // or flo.
[-10 to -4]

print (s[-8:2]) // Empty.
(Starting ~~Index~~ < Ending Index)

print (s[-7: 10]) // Low.
(-7 to 9.]

print (s[1: -10]) // en s
[1 : -11]

```
print(s[-3: -9: -1]) // a wolf.
print (s[-2: -13: -3]) // . wfs.

print (s[4: -2: 3]) // oL_

print (s[2: -1: 2]) // nofo.

print (s[-2: 1: -4]). // .DO.
print (s[ 2: : -2]) // nT.
print (s[-10: : 2]) // ofo.
print (s[5 ::3]) // roa
print (s[-2 :: -4]). // .00T.
```

Note :

If the start, step value are gn, then dir will be dete. by step value

If step value → +ve → L to R.
↳ -ve → R to L.

x. ———————— x