Type Casting -

Implicit Type Casting

Python automatically does this without programmers intervention.

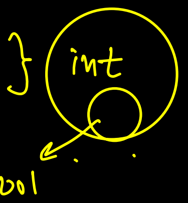
Don't need any programmer intervention

Eg. $\underset{\text{int}}{5} + \underset{\text{float}}{10.5} = \underset{\text{float}}{15.5}$ ⇐ automatically

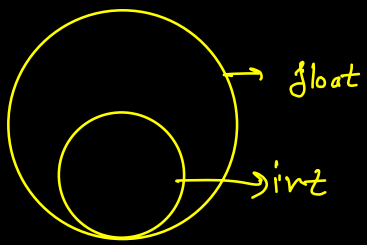
⇒ data type int is ~~at~~ converted internally to larger one to avoid data loss
$\Downarrow$
data type

Larger data type ⇒ int float bool

T - 1
F - 0 }



5 + 10.5 ⇒



Explicit Type Casting

When done explicitly by the programmer

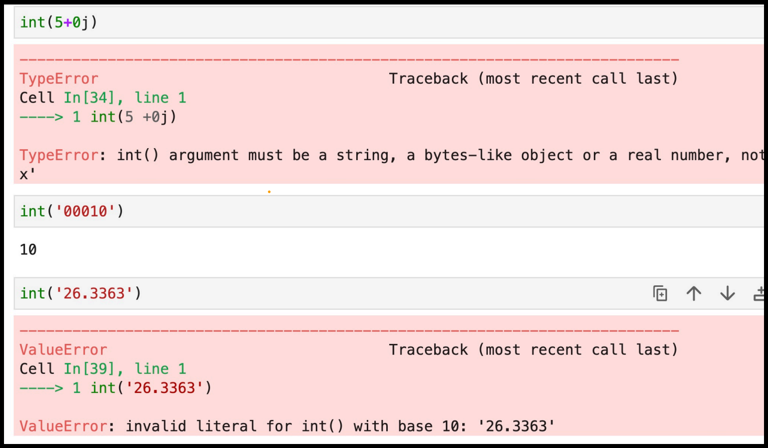
$\boxed{\text{'999999'}} \Rightarrow$ int

int ( )
float( )
bool ( )
str()
complex ( )

int ('5') ⇒ 5

int ( )

⇒ takes a data type & convert it to integer.



```
int(5+0j)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 int(5 +0j)

TypeError: int() argument must be a string, a bytes-like object or a real number, not
x'
```

```
int('00010')

10
```

```
int('26.3363')

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[39], line 1
----> 1 int('26.3363')

ValueError: invalid literal for int() with base 10: '26.3363'
```

Float ()

takes an input & changes it to
float

Bool $\Longrightarrow$ $\underset{F}{0}$, $\underset{T}{1}$

takes a number & change to bool

Only 0 & 0.0 are false   0.0   -0
                                         -0.0

Only 0 + 0j is false
0.0 + 0j
0 + 0.0j
0.0 + 0.0j

$x - x = 0 = $ false

Strings

Only "" [empty string] is false

Only string with len = 0 is false

'0'   '0.0'   '0+0j' all True

$$\left.\begin{array}{l} [\;] \Rightarrow \text{list} \\ \{\;\} \Rightarrow \text{set} \\ (\;) \Rightarrow \text{Topple} \end{array}\right\} \text{empty} \Rightarrow \text{False}$$

$$\boxed{\begin{array}{l} 0\ ,\ 0.0\ ,\ 0+0j\ ,\ 0j \\ \text{"" , or string with length } 0 \\ [\;]\ ,\ \{\;\}\ ,\ (\;) \end{array}}$$

↑

all above are False

String

⇒ any data type to string

⇒ no special case

input() takes anything you enter as a string.

String concatenation

If we have two strings & we do '+' on them.

⇒ strings get concatenated.

Examples.

1. First name + Lastname
2. Address.
3. url.

* ⇒ works with int (True, false)

Slicing

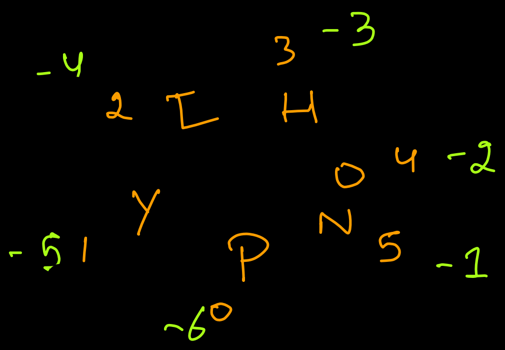
Str   but its valid on all seq
list
range
tupple

	-6	-5	-4	-3	-2	-1
s1 =	P	Y	T	H	O	N
	0	1	2	3	4	5

s1[1] = Y

s1[-3] = H

P	Y	T	H	O	N	= str
0	1	2	3	4	5	
-6	-5	-4	-3	-2	-1	



slice out of work

cake slice

$\boxed{+91}$ - 999...9 ⇒ Country code

3/Feb/98 ⇒ month

url ⇒ www. google. com ⇒ domain

⇒ Slicing is pulling out a part from a sequence

⇒ [ ] ⇒ this is used for slicing

⇒ [start : end]

⇒ End takes one pos^n before the specified index.

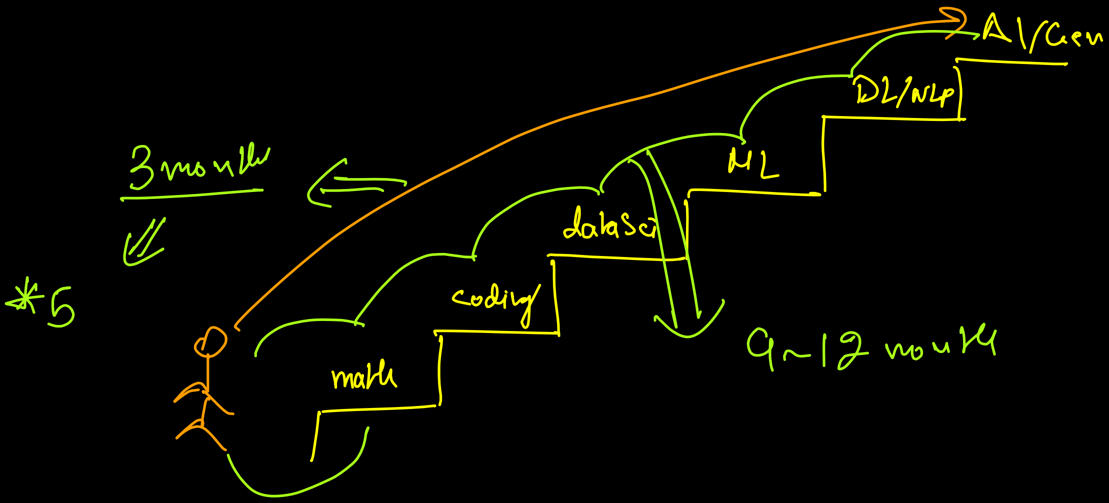
⇒ [start :] default end is len

[: end] default start is 0

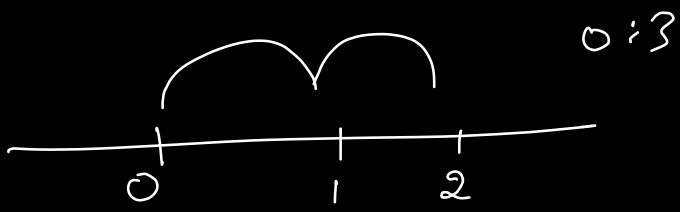
end > len ⇒ end = len

⇒ empty

a >= b for positive step,
start stop it is always empty



Step / Stride [default is 1]



[start : stop : step]

$\overset{\parallel}{1}$