② a = 10
print (a).
print (a, end = " ") → ends with new line by default

↱ delimiter argument

print function

(*) The last line result is treated as output of the cell (in jupyter notebook only)

a = 10
a + 10 → prints 20, i.e. line at last as output
↓ final answer of the cell

variables
↓ type declaration is not req. in python
↳ automatic type inference

↓ not used for many variables.

(*) type (a). ↳ showcases the type of the variable
↓
<class 'int' >
↓
this can change with each change of value or assign → dynamic nature
↓
can even be assigned complex values.
↓
a = 2 + 3J → <class 'complex'>

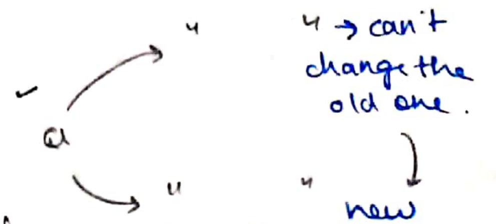
(*) Basic operators from other languages exist here as well. (+, -, *)
↓
variable of one cell can be used in other cell, within same scope
↓
(order of execution)

string → Immutable
list → mutable
tuple → immutable
dictionary → mutable
set → mutable

(*) a / b → float
a // b → int (floor division)

a , b = 2 , 3
(valid initialisation)

(*) a * b (multiply)
a ** b (exponentiation)



(referencing to different strings)

(*) strings :

a = " this is a string "

→ strings are immutable ↓ particular character cannot be changed

· len (a) → length of a
type (a) → str.
a [0] → accessing particular element -
a [2] = " s " → not valid ×
a = " def " → valid ✓

python does not have chars. ↓ ' s ' → type ' str '

* concatenation is allowed
* string + (integer) (won't work) ↳ (but can be typecasted)

a = ' Abc '
print (a . upper ( ).)   (returns a string, not inplace)
↳ a is not changing but is getting stored in temp. and getting printed

print (a . lower ( )) → (just printing. doesn't change the variable)

print (a . strip ( ))
↓
(**) removes spaces (→ white space) at end and start of a string. (all whitespace) ↓ (enter, tab)

print (a . isalpha ( ))
↓
if all elements of a string are alphabets.

↱ corresponding isdigit ( ) available .

a = input (" Please enter input : ")

(*) user input by default is a string ✓

a = int (input (" Please enter input : ")) → convert input to int

(* : console running symbol)

④ Slicing → works for strings as well as arrays (called lists in python)

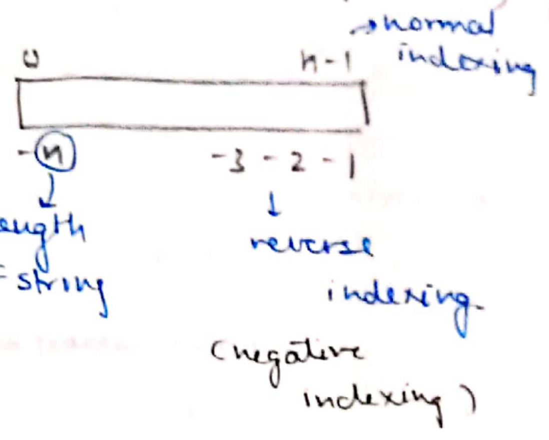
(getting substring of a string)

s = "abdefg"   s[-1] = 'g'

t = s[1:3]

1 → included   3 → not included

p = s[-3:-1]



print(s[1:])

↓

index 1 to end

print(s[:3])

↓

everything till 3 → i.e index 2 will be the limit to include in slicing.

print(s[:])

↓

no constraints on beginning and ending index

↓

slicing basically means fetching a substring with constraints on beginning and ending index.

* Tuple . (not an array) → (immutable lists) → objects stored can be mutable

↓ used to store multiple things together

t = (1, 2, 3)

type (t) → tuple

t [0] → normal access method.

* can store different kinds of entities bundled together (even lists)
* immutable.
* slicing can be used.
* operators on tuples.

t = (1, 2)

k = ("d", 1)

print (t+k) → (1, 2, 'd', 1)

print (t*k) → error.

print (t* 2) → possible (1, 2, 1, 2) → makes scalar copies of the tuple in one tuple

print (1 in t)

↳ boolean function to check presence of element in tuple

lists → [ ]

tuple → ( )

sets → {,}

Dictionary → { } (key, value pair)