Passing variables through function -

```
def increment(a):
    a = a + 2
    return a (or return)
```

→ // a - increment = 2

Both are different

```
a = 2 → Value passed ⟹ // a - main = 2
increment(a)
print(a) // o/p = 2
print(increment(a)) // o/p = 4
```

~~o/p 2 2~~

a - main ~~is~~ → 2

When passed,

a - increment

after running the fn code,

a - increment ~~→ 2~~ → 4

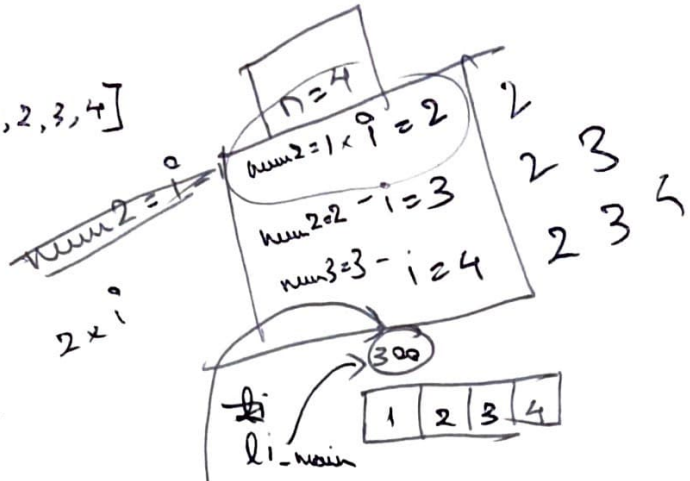
```
a = 2 // a - main
a = increment(a) // a - main = increment(a - increment)
print(a) // o/p = 4 //
```

(a - main)

Passing list through functions -

```
def increment(li):
    li[0] = li[0] + 2
    return
```

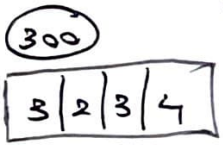
```
li = [1,2,3,4]
increment(li)  // o/p = [3,2,3,4]
print(li)
```



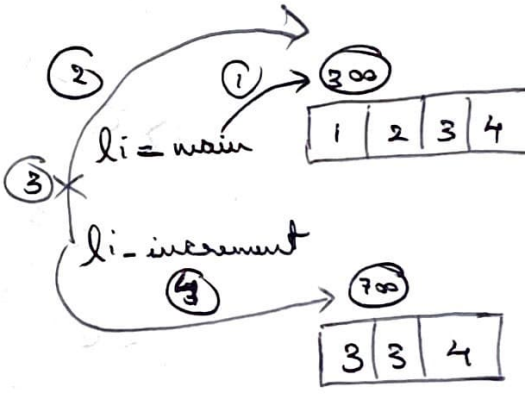

```
li = [1,2,3,4]  // li_main
increment(li)  // li_increment
```

Since, both pointing to same reference.
We see the changes even without storing it in a seperate variable.

li_increment[0] = li_increment[0] + 2

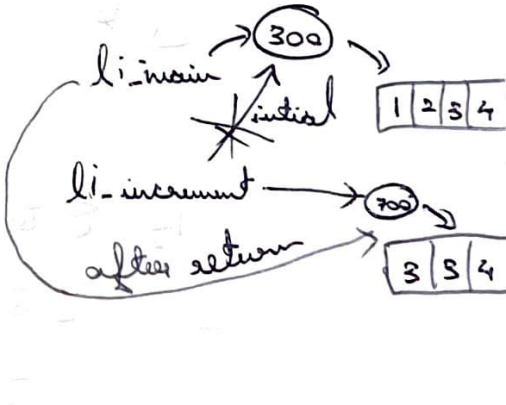


→
```
def increment(li):
    li = [3,3,4]
    return
// same as above
print(li)  // o/p = [1,2,3,4]
```



```
def increment(li):
    li = [3,3,4]
    return li

li = [1,2,3,4]
li = increment(li)
print(li)  // o/p = [3,3,4]
```



Strings

String → a sequence of characters (unicode) → ASCII (Java/C++)
↓ encoding to 0 & 1.
Python

variable name = "substance value" / 'value', """value"""

↓ ↓ ↓

All content in one line

Content can be printed in multiple lines

multi line string

Indexing in string-

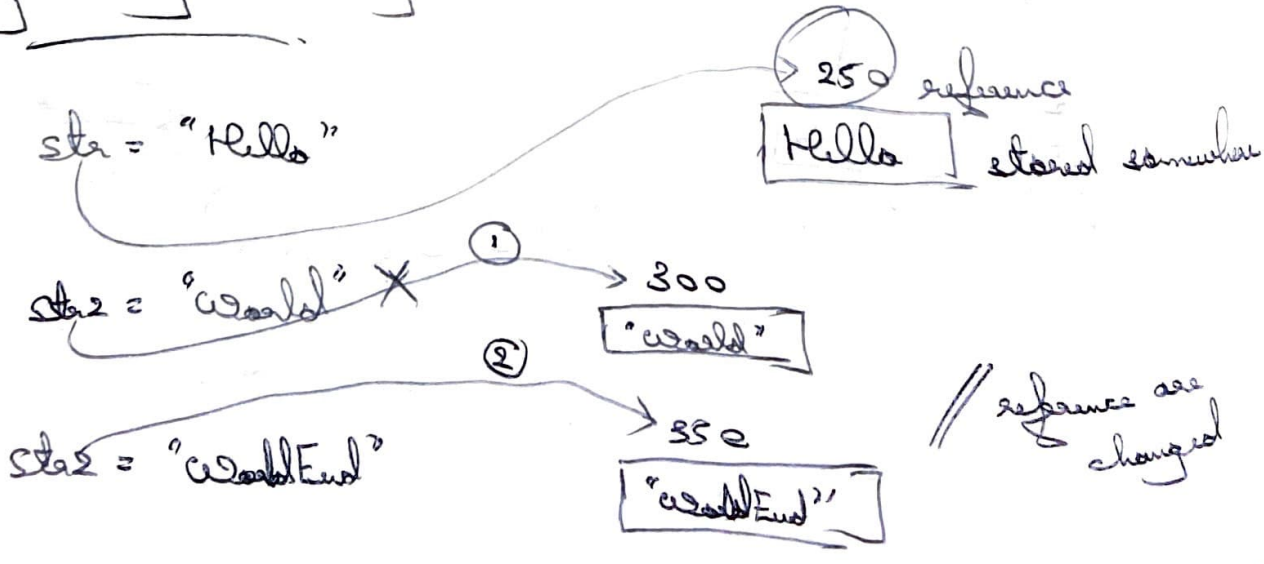
string_variable [index_value]

↳ should be within range

In case of """ """ → \n is also used as an index value.

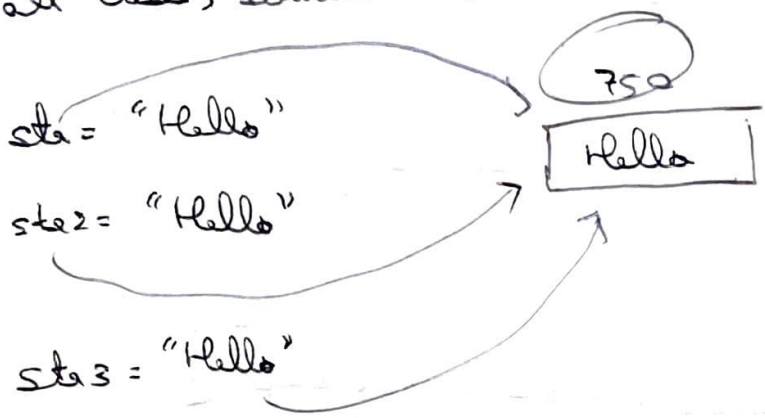
Negative indexing also present.

String storing (internally)



Python optimization:

Not all cases, sometimes. // For, smaller strings.



Strings ~~are~~ are immutable.

```
sta = "Vishwa"
sta[0] = "C"  X  doesn't work.
```

Concatenation of strings:

```
result = stringname1 + stringname2
```

also,

```
stringname1 = stringname1 + "Hello"
stringname1 += "Hello"
```

```
sta = "Hello"
sta = sta * 3
```

o/p = HelloHelloHello

But, a string & a integer can't be appended (in Python)

Solu: str = string + str(integer)

Slicing on strings

[start_index : end_index : step]

if greater than end_index ⇒ empty string

Negative indexing:
s[-1] → last index

index of out of length → returns till last element from start

reversing → [: : -1]

Iterating on string

Using for loop-

character wise → for character in string:

index wise → for index in range (len(string):

in and not in operation

substring method

↳ no skipping

↳ left to right direction

in → checks if a substring of is part of given string

not in → " " " " " " " " " " "

Comparison in case of strings (>=, <=, !=, >, <, ==)

compares each character based on ASCII value.

A → 65.. a → 97..

len(str1) = 6 , len(str2) = 5

↳ Greater value.

Operations / functions (in built) on strings

split(,) → separates the i/p by space (default) / delimiter

↳ delimiter.

↳ space, comma

returns a list of string

↳ no. of items by which it should split

(eg: 1 word, 2 word)...

replace(,) - replace a string with another string.

↳ old char, new char

if old char → not present → no ~~change~~ error

" " → present multiple times → specify no. of old char

which you want to replace.

replace (oldchar, newchar, no. of oldchars)

↓ 1, 2, 3 . . .

↳ starting from first.

find() - finds substring in a string.

returns index from where substring starts, if not found returns -1.

→ substring (s),

→ substring, start index, end index → search ss in that particular part only

(ss)

lower() & upper() -

↳ lower → capitalize

↳ small letter

startswith() → whether starts with a substring or not.

→ substring

→ substring, start index, end index

returns true / false.