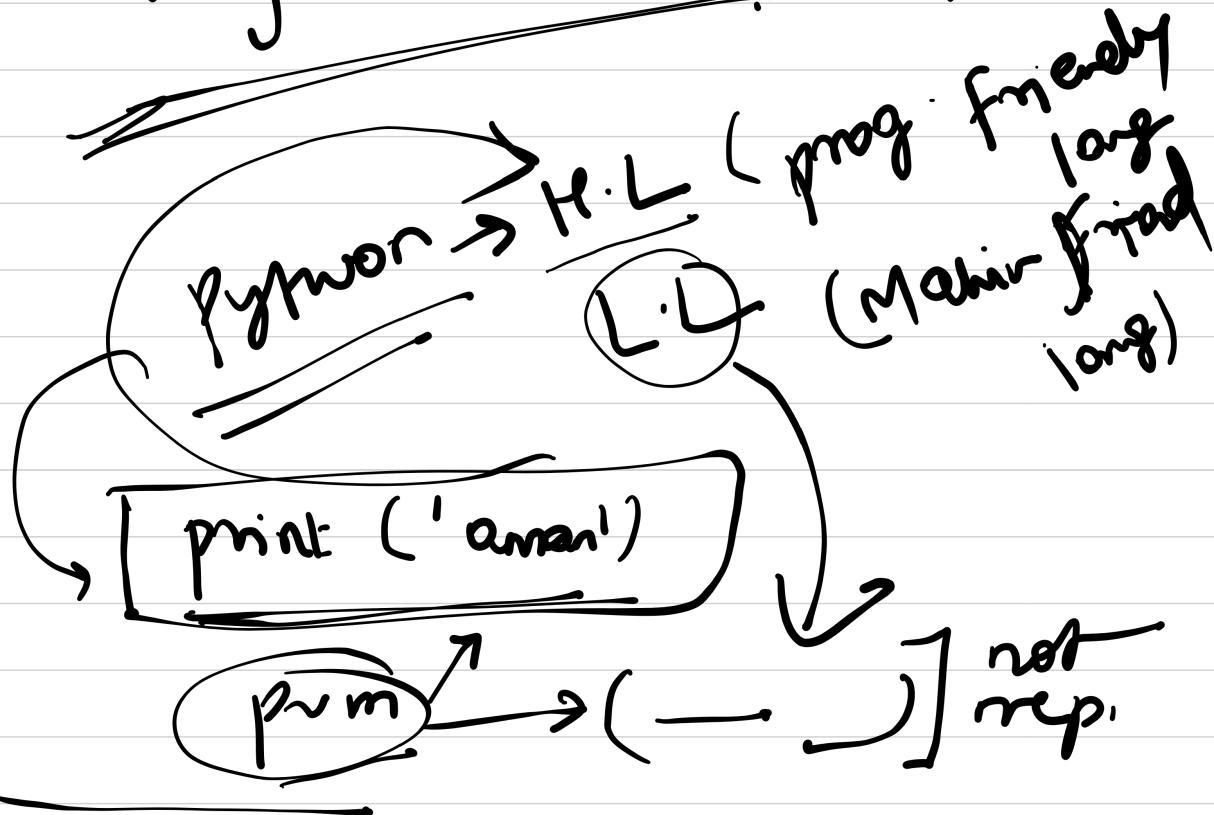




why we are learning python.



python → Sir Guido van Rossum

↳ born Feb

1991

Java → compiler  
interpreter

Made with Goodnotes

Python → interpreter

# In python how code is compiled?

Pvm → interpreter →



Java data types

$d = 10$

$d = \underline{\text{int}}$

python data types

$d = \underline{10} \rightarrow \text{int}$

[ $d = \text{int}$ ] ? NO

why No?

"String" = String

# what is dynamically typed?

[  
d = 'man'  $\Rightarrow$  str]  
d = 10  $\Rightarrow$  int  
]

$$a \neq b \quad a = 10 \\ b = 20$$

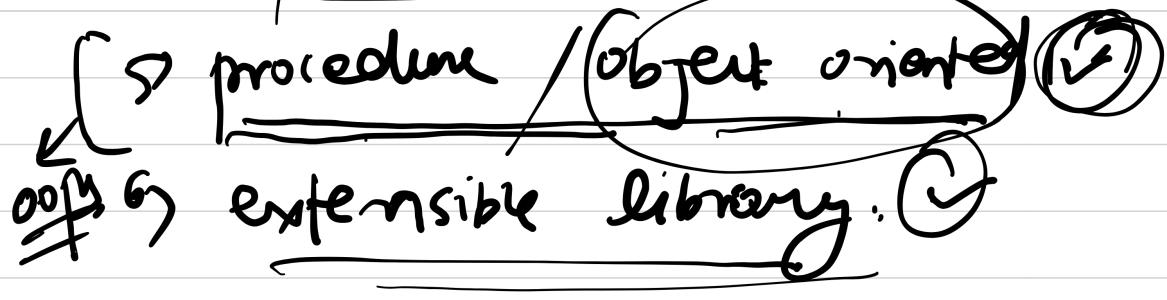
print(a+b)  $\neq$  30

$\Rightarrow$  [ a, b = 10, 20 ]  $\Rightarrow$  packing / unpacking  
print. (a+b)  $\neq$  30

python  
a, b, c, d = 1, 2, 3, 4

# Features of python:-

- 1) Simple & easy to learn. ✓
- 2) Free ware & open source ✓
- 3) H.L. ✓
- 4) dynamically typed ✓ *oracles*



~~oopm~~ → If everything can be done using object(OOP) why procedure?

free ware →

Python → free

Java → oracle (pay)

Open source →

source

Python

use case

freeware

\* Python

open

source

python

[payment]

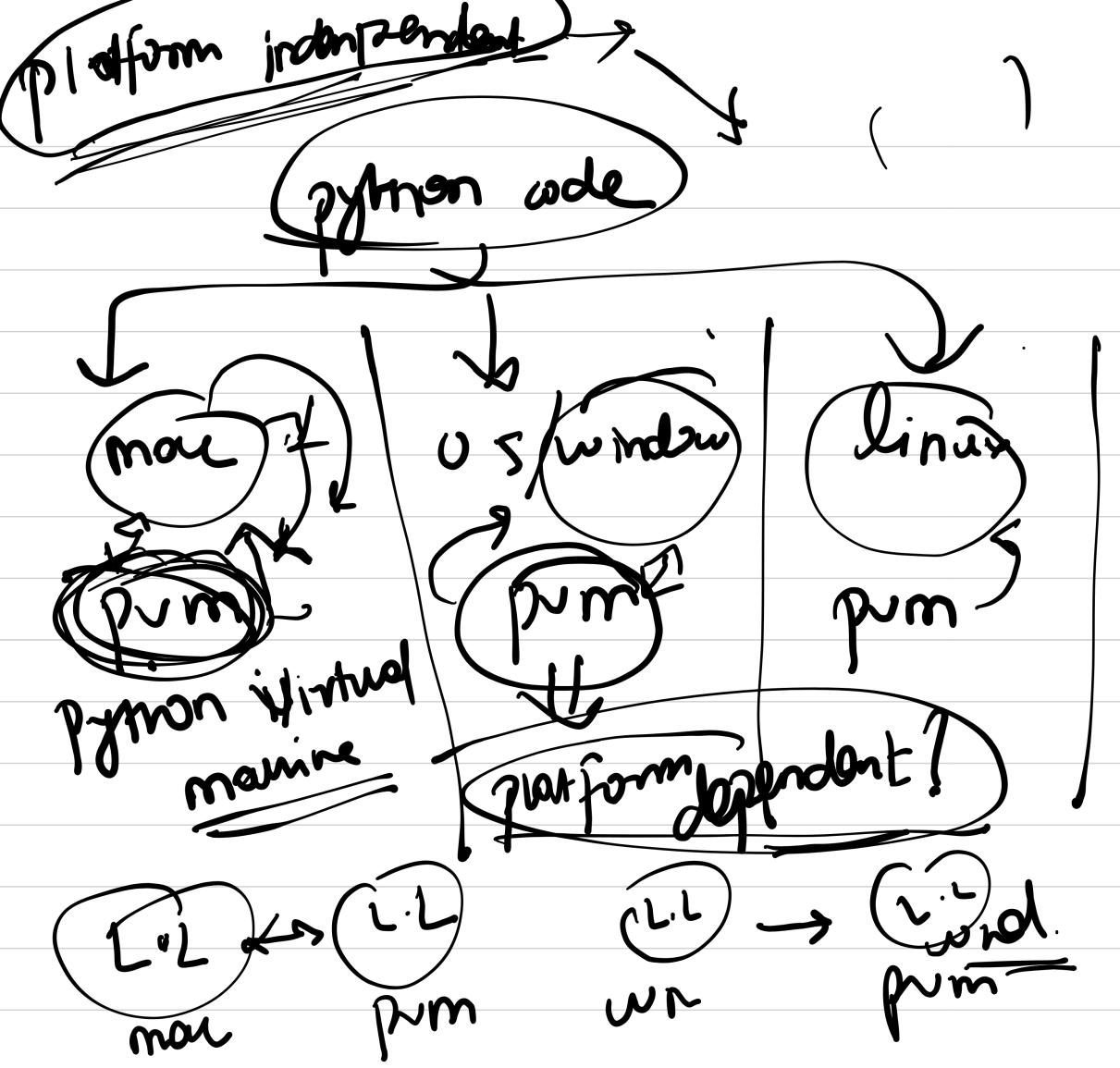
ware

code

11

Python

Py Py



pyvm python virtual machine  
(no compiler)  
↓  
interpreted

print (10 \* a)

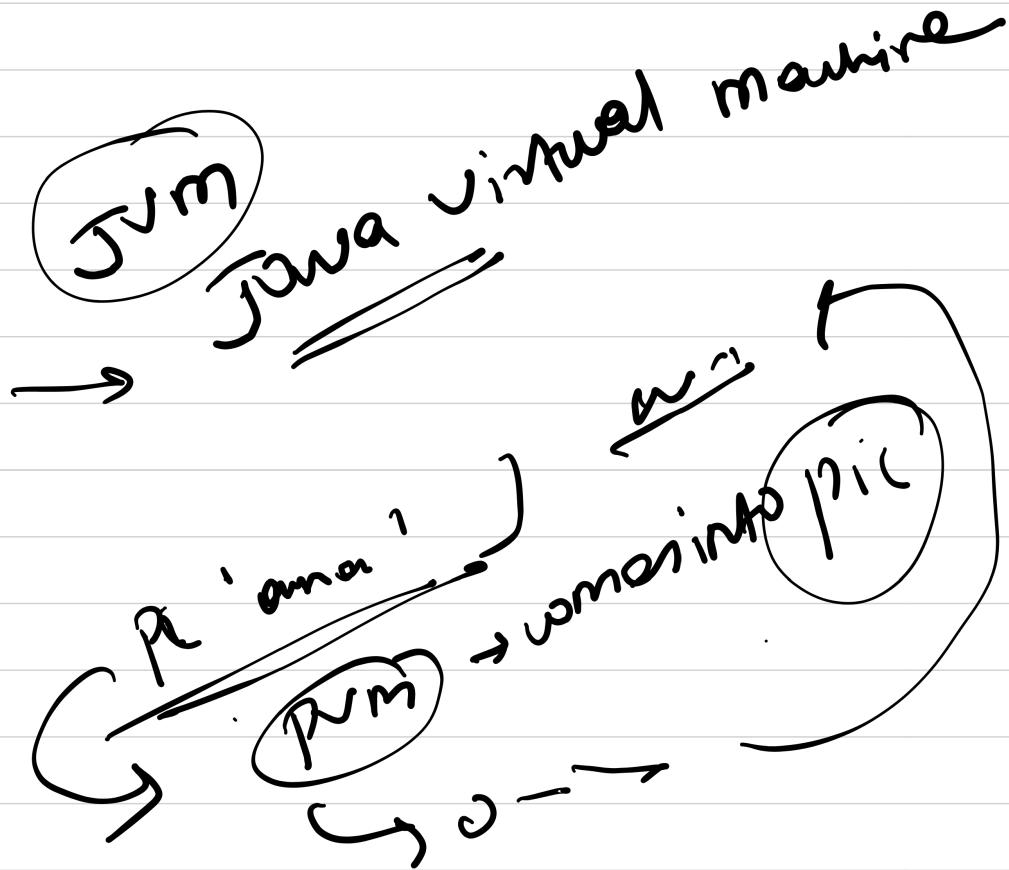
#

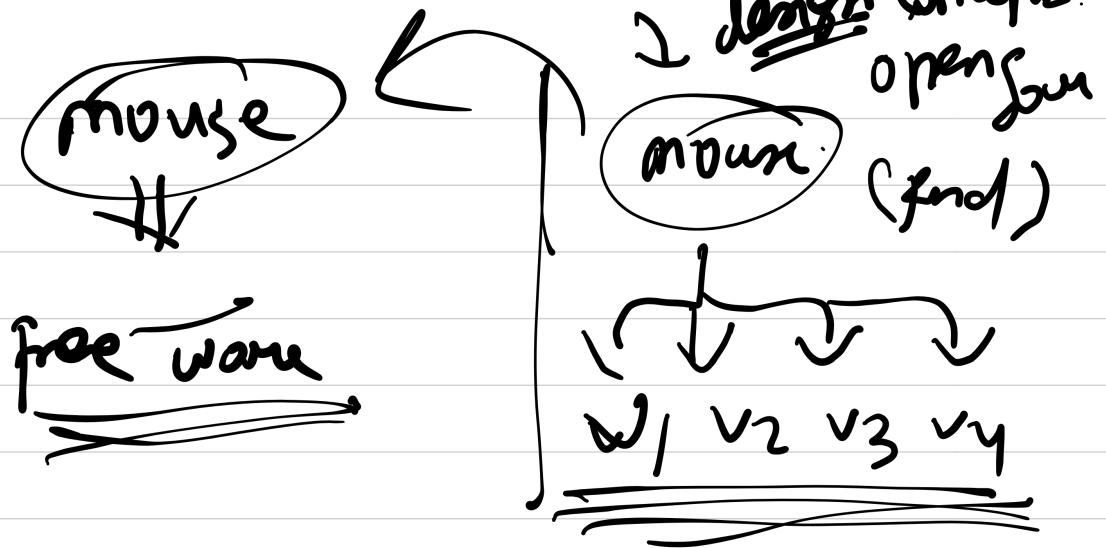
Syntax error

pyvm → compile code + execute code

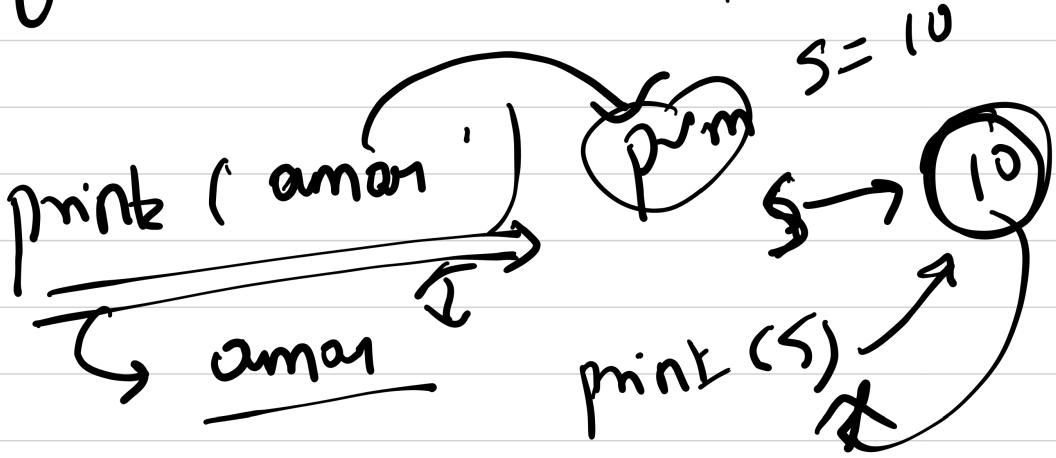
• py → • pyc <sup>version</sup> <sup>compile</sup> code

# extensive libraries





python works internally,



$n = 1$

**Note:** In Python everything is treated as obj

$a = 10$

num

$y = 11$

$y = 10$

~~num~~

2

10

✓

~~y~~

11

g.c

$y \rightarrow 10$

$x \rightarrow 10 \leftarrow y$

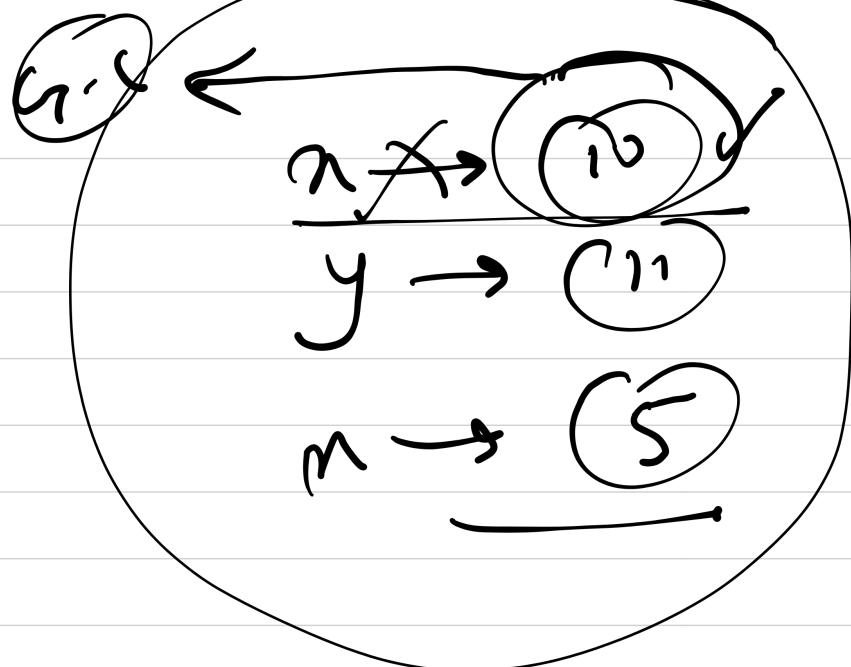
print ('aman')  
 $\rightarrow$  aman

print(s) f. s →  $\circ$

g.c

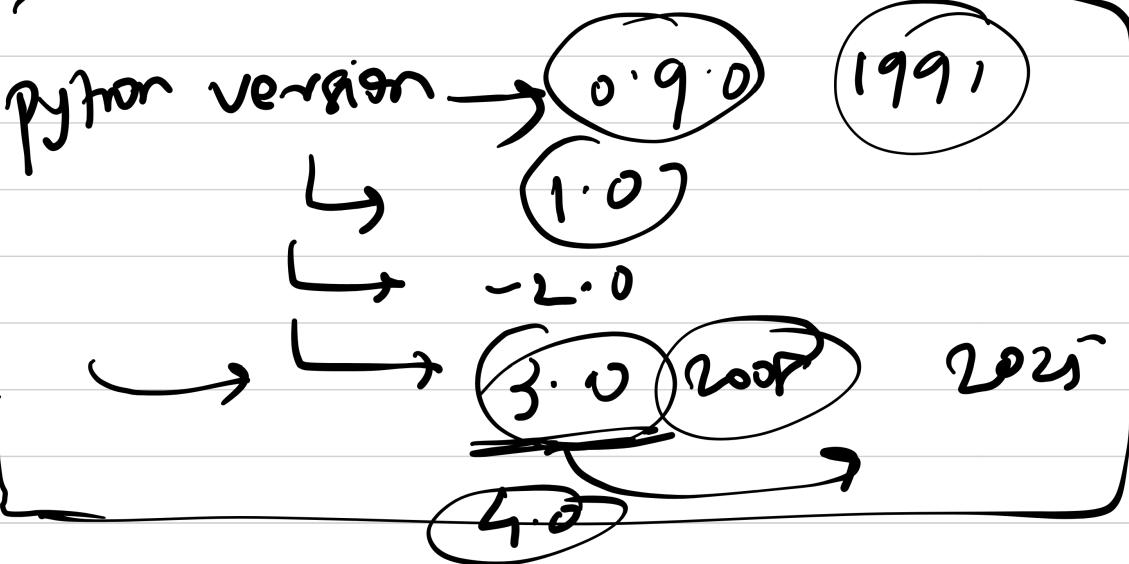
$x = w$   
 $y = 11$

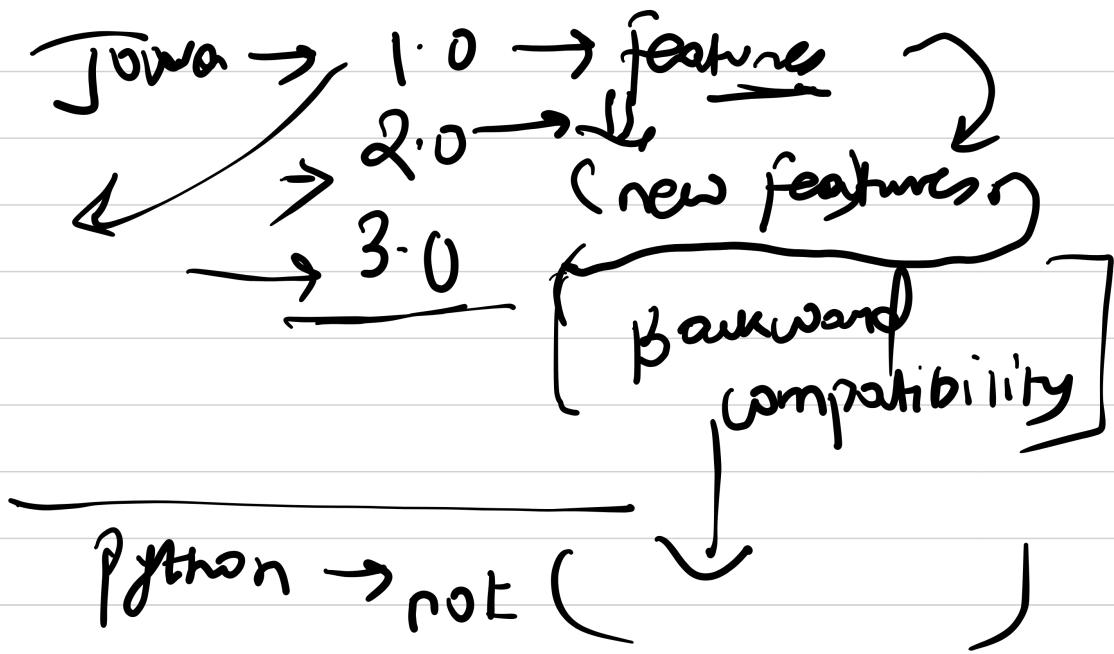
$x = 5$



~~num  
unipar~~

python version





Identifiers: python 2010 | notes (0) 10

keywords (g)

2011 (c) → new notes

2025 (3.6) → new notes

pass for (.

JUNIOR

Identifier →

{ a to z  
A to Z  
0 to 9  
\_ (underline)

Rules for identifier

→ gamen

"

"

Error

aman98 = "

"

aaaao —————  
aaa) = 10

Print (

) # 10

If = 10 will it work? No  
then why?

keywords → 35 ✓

import keyword  
Keyword · kwlist )

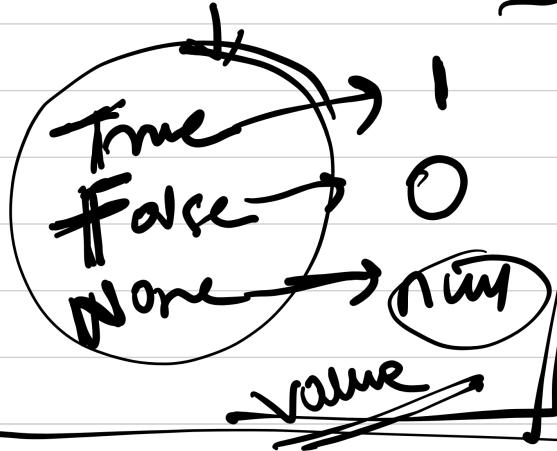
If = 10

[ 35 reserved  
keywords ]

answer = 10

35 keywords

reserved  
literals



Keywords



if, for, etc

functionality

data types:

14

as

int  
float  
complex  
bool  
str

list  
tuple  
set  
dict  
frozenset

bytes  
bytearray  
range  
None

Fundamental dt.

↑  
Collection  
dt

dynamically typed

$n = 10$

$n = \underline{\text{int}}$

print only  $\neq 10$

1) Int (integral values)

$n = \underline{\text{10}}$   
 $\underline{20}$   
 $\underline{30}$

(non decimal)

2) float (decimal values)

$n = \underline{10.5}$

type only  $\neq$  class float

1) \python 2 \python 3

Int = 10  
long = 11111111 → Int

3) complex :-  $a + b\hat{j}$   $\hat{j} = \sqrt{-1}$

real imaginary

4) Bool (logical values) comparison  
purpose

T/F → T/F  
 If age > 10:  
 print ('Younger')

str :- (string) variable

s = "aman" → n = Aman

s = 'aman' → n → aman

s = "aman" ✓

(multi line string literal →)

$s = " " \left( \begin{array}{l} \text{Python is} \\ \text{easy to} \\ \text{learn} \end{array} \right) " "$

$s = (" " \cdot \text{Python} \cdot " " \text{is easy to learn} \cdot ")$

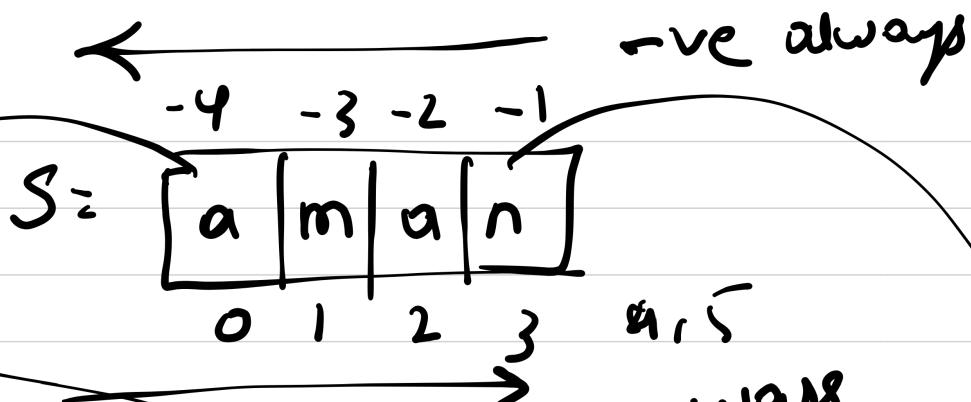
✓ ↓

$s = \text{Python is easy}$  ✓

$= " " \text{Python} " " \text{is easy} " "$  /error

slice operator:-

$s = \text{aman}$



$s[0] = a$

$s[-1] = n$  (last character)

$s[5]$  = Index error (out of range)

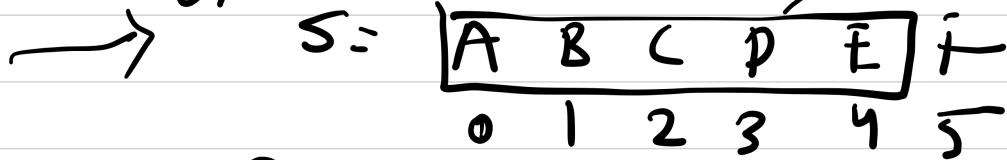
Slice operator :-

$[ ]$  receive part of slice

$[1, 2, 3, 4, 5]$

$s[\underline{\text{begin}} : \underline{\text{end}}]$

end - 1



$s[0:4]$

Begin to end - 1

$s[0:5]$   $5-1=4$

0 1 2 3 4

$s = [A B C D E F G H I J]$

0 1 2 3 4 5 6 7 8 9

$s[:]$

• not mandatory to pass

$s[:]$  ✓ entire string

$s[3:] \Rightarrow D E \dots - J$

$s[:7] \Rightarrow [A \dots f g]$

$$7-1 = 6$$

play

$s[0:11]$  # slice operator  
won't raise index  
error.

$[A : J]$  # no error

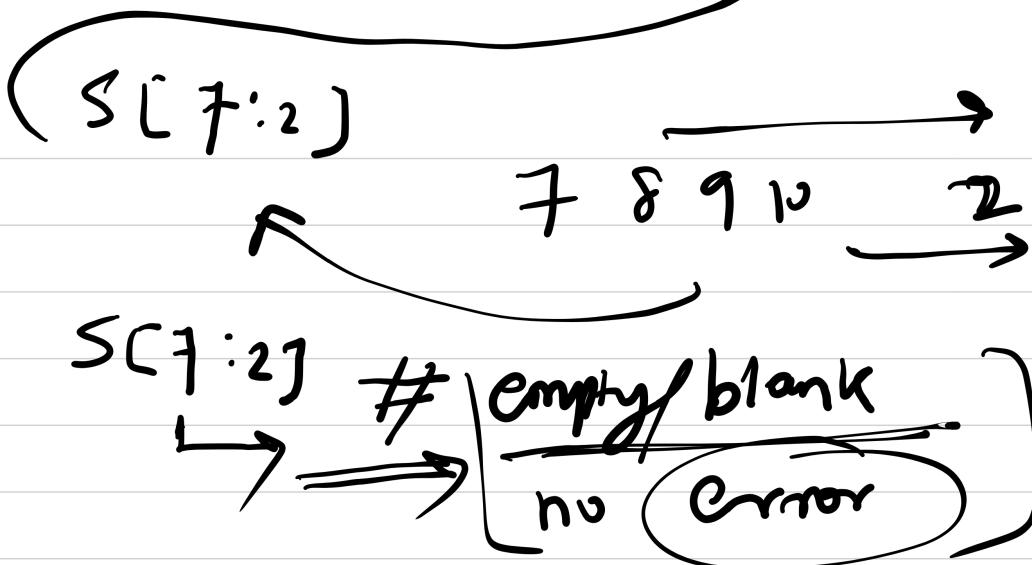
$s[2:7] \quad 7-1 = 6$

$s = [A \ B \ C \ D \ E \ F \ G \ H]$

0 1 2 3 4 5 6 7 →

← → 1 ↗

Diagram showing a string  $s$  with indices from 0 to 7. The character at index 2 is circled and labeled 'C'. The character at index 1 is circled and labeled 'B'. The character at index 0 is circled and labeled 'A'. The character at index 7 is circled and labeled 'H'. The character at index 6 is circled and labeled 'G'. The character at index 5 is circled and labeled 'F'. The character at index 4 is circled and labeled 'E'. The character at index 3 is circled and labeled 'D'. The character at index 2 is circled and labeled 'C'. The character at index 1 is circled and labeled 'B'. The character at index 0 is circled and labeled 'A'.



$s[2:3]$

$3 - 1 = 2$

$s[: -1]$

-2

$$-1 - 1 = -2$$

$s[2:2]$

$s[2] \rightarrow [2:-1]$

$2 - 2 = 0$

$s = \underline{\text{empty}}$

$s = \text{Aman} \begin{smallmatrix} 0 \\ 1 \\ 2 \\ 3 \end{smallmatrix}$  # Amon  
AmaN

$s_1 = s[0].upper() + s[i:]$

A + man

ang/

$s = \underline{\text{AmaN}}$  ↘

$s_1 = \underline{s[-1].upper()} + \underline{s[0:len(s)-1]}$

= Nama

0 to 2

$[0:4-1]$

$s[ ] + s_1$

$[0:3]$  ↘ 3-1

$[0:2]$

amON

$s = \text{'aman'}$

0 1 2 3

a	m	a	n
0	1	2	3

$s[0].upper() +$

$s[1:]$

$\downarrow$   
 $\rightarrow \underline{Aman}$

$A + man = Aman$

$s = \underline{\text{amoN}}$

$s = \underline{A|m|u|n} \neq (y) \underset{\nwarrow}{\text{lens}}$

0 1 2 3

$\rightarrow \rightarrow -1 -1$

$s = [0 : \underline{\text{lens}} - 1] + s[-1].$

$\downarrow$

$[0 : 3] 3 - \text{end} - 1$

$\downarrow \underline{\text{upper()}}$

$0 : 2 \leftarrow$

$ama + N = amoN$

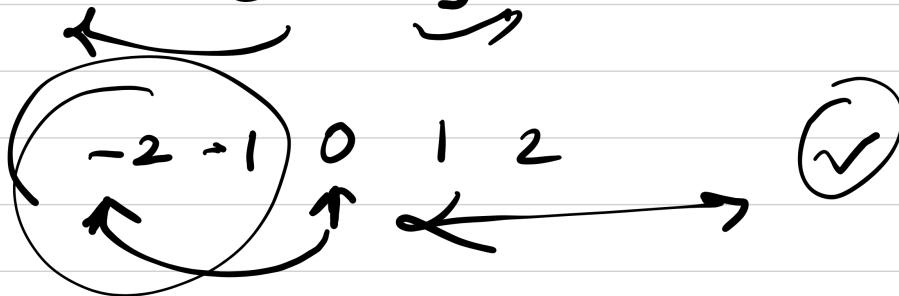
$\text{len}(s) - 1$

[  
  4 - 2  
  2]  $\rightarrow$  end-1

[0:1]  $2 - 1 = 1$

s[0:-1]  $\rightarrow$  end-1  $-1 - 1$

s[0:-2]



math operators.

$s = \text{'aman'} \oplus \text{'R'}$

H amanR \*

$s = 'aman' \langle T \rangle \langle T \rangle 'R'$

$\# \underline{\text{aman}} R$

~~space~~

int  $\times$  ]

$s = 'aman' + I \# \underline{\text{error}}$

$\downarrow \quad \downarrow$

rule  $\rightarrow S1r + S2r$

$s = 'aman' + 'I' \# \underline{\text{aman}}$

$S1r$

$S2r$

$s = 'aman' \langle * \rangle 2 \# \underline{\text{aman aman}}$

$\uparrow$   
int

'aman' + 'R'

aman R

Kue

$S = "aman"$ ,  $"*$ ,  $'aman'$   $\#$  err.

$\#$  error

$+$

$S_0 + S_1$

$*$

G.C

$S$  →  $'aman'$   
 $'aman' \leftarrow S$

$S = 'aman'$   
 $S \rightarrow 'aman'$   
 $'aman' \leftarrow S$

$S_1 \rightarrow$

$S = 'aman' + S'$

$S_1 = 'aman'$

$S_2 = 'python'$

I

$s_1 + s_2$

$s_2 = ' aman' + s_1$

try  
it?

## Fundamental data types vs Immutability

int

float

complex

Bool

str



fund

(immutable)

$n = 10$

$n = 11$

11

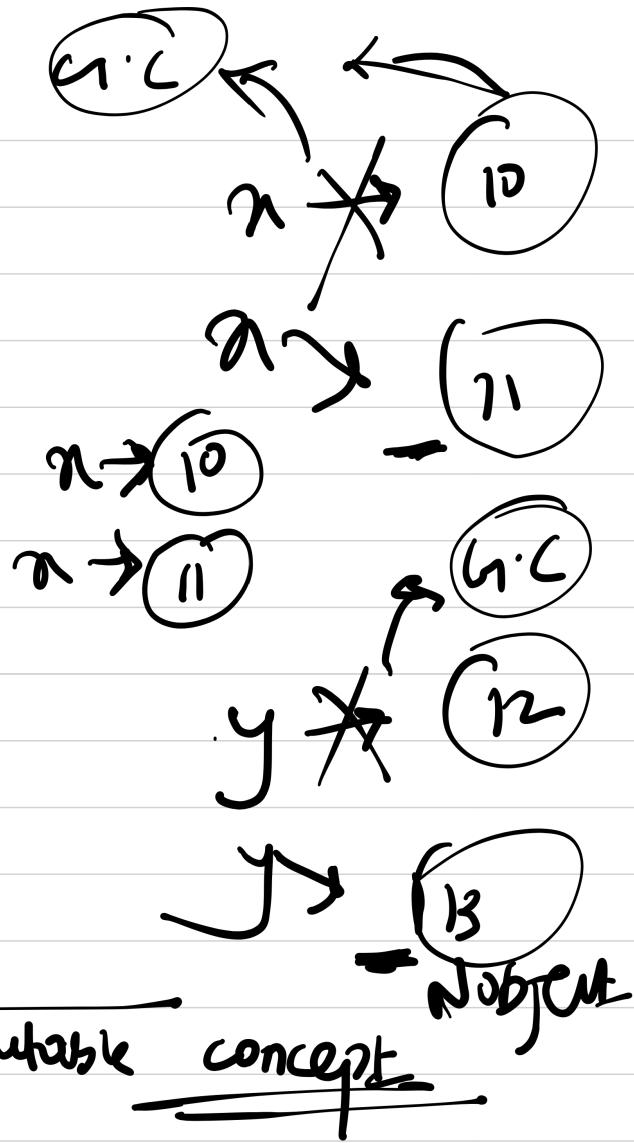
10

11

$$x = 10$$

$$n = n + 1$$

$$= 11$$

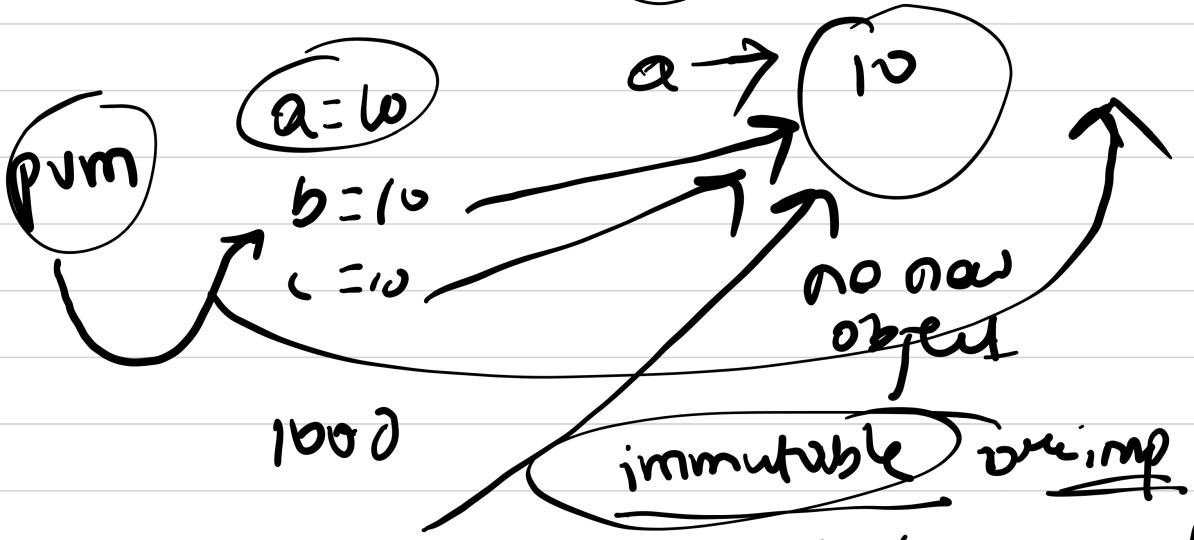


$a = 10$
$b = 10$
$c = 10$

10

How many objects will be created?

3 vs 1 ✓ already existing



- 1) memory utilization will be improved
- 2) object creation is costly  $\rightarrow$  no new object created

enhanced performance ← →  
vs  
mutable

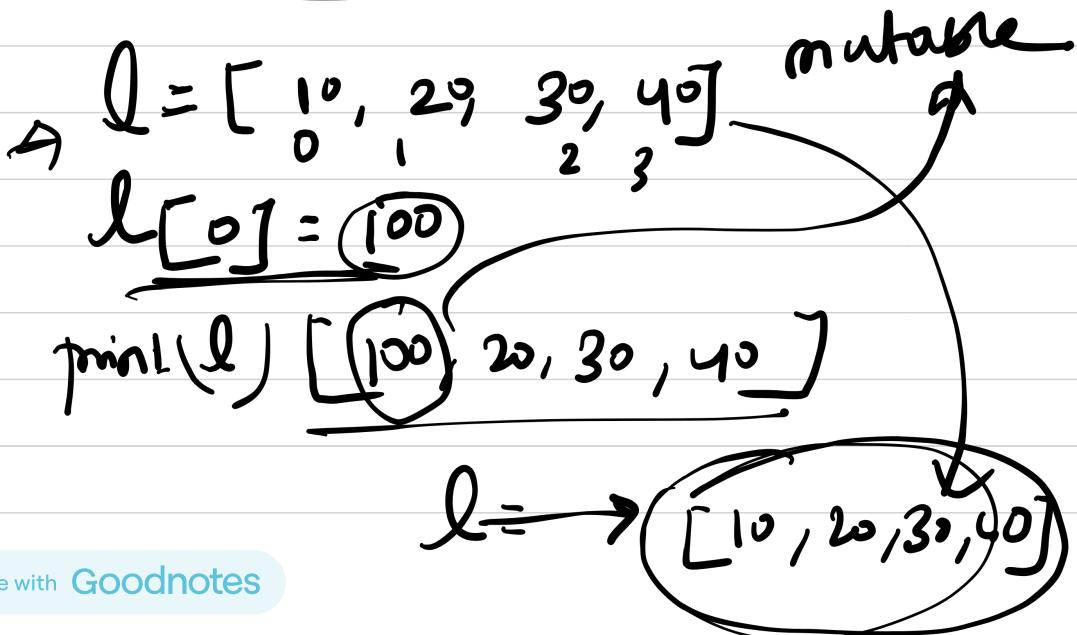
$$n = 10 \cdot s^-$$

$$y = 10 \cdot s^-$$

disum later  
 $y \rightarrow$   $10 \cdot s^-$   
 $n \nearrow$

From reuse existing objects

server for object is available / not

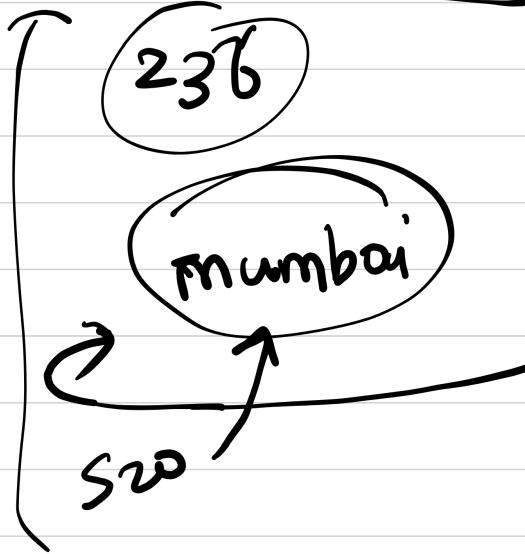


$\lambda = \rightarrow$

funds  $\hookrightarrow$  immutable

[100, 20, 30, 40]

mutable



$S_1 \rightarrow$  pune

MDfe [ Saving, current ]

all ~~new~~ item

Saving

Immutability

→ Cannot change

$a = (1, 2, 3)$  → tuple  
 $b = (1, 2, 3)$  list

$$01 = \begin{matrix} 10 \\ = 10 \\ = 10 \end{matrix}$$

$a \neq b$  # True  
# False

$a = 10 \cdot 8$

$b = 10 \cdot 8$

float  $1.0 + 0008$

10  
20  
30

$a = (1, 2, 3)$

$b = (1, 2, 3)$

?

Installation

content

$a = [1, 2]$

$b = [1, 2]$

also

$a = b$

# True  
True  
address of  
object

compromising?

True

