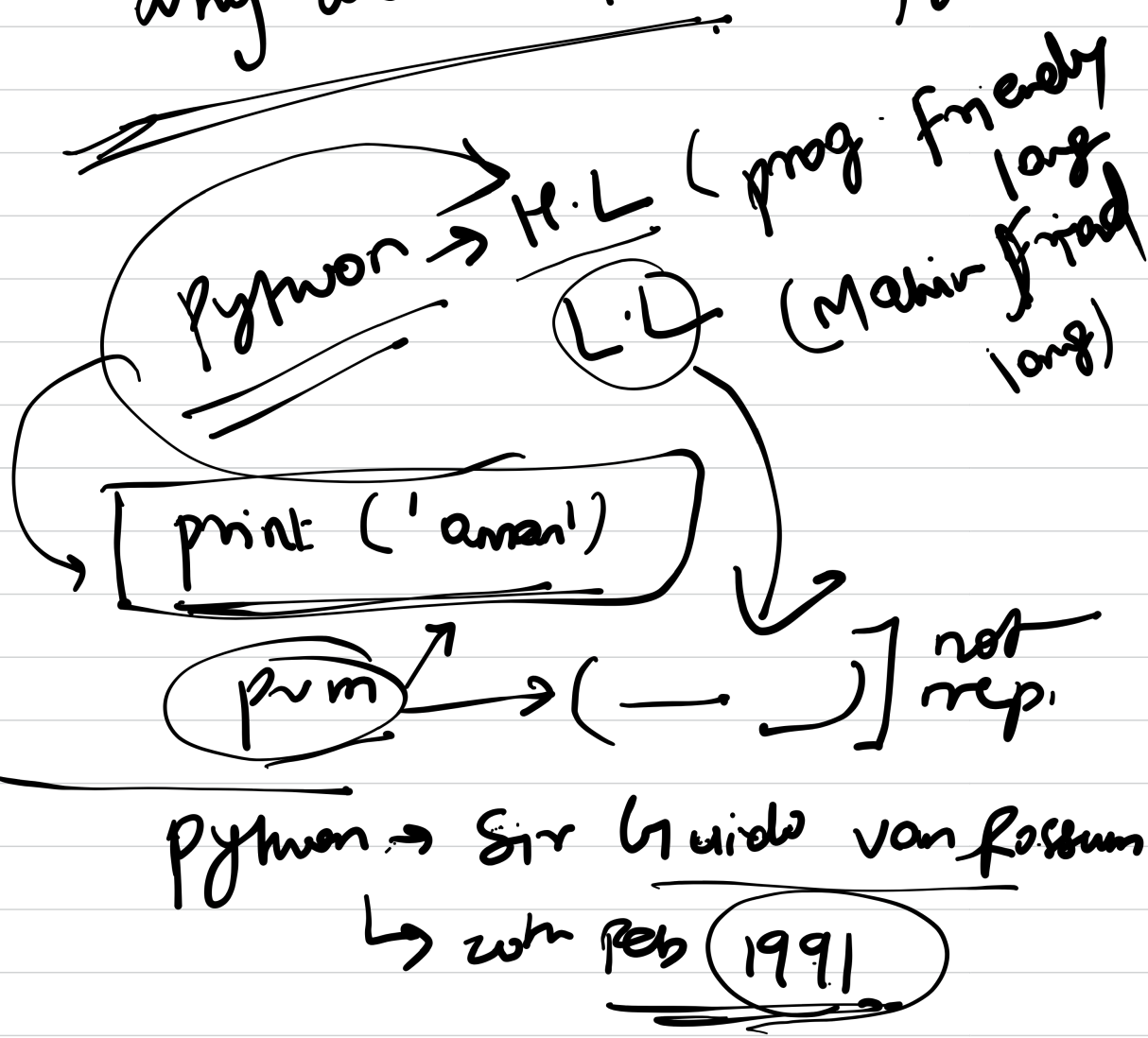
Why we are learning python?



Python → Sir Guido van Rossum

↳ 20th Feb 1991

Java → compiler interpreter	Python → interpr

In python how code is compiled?

p vm → interpreter →

Java data types

d = 10

d = {int}

python data types

d = 10 → int

[d = int] ? No

why No?

'string' = str

what is dynamically typed?

d = 'aman' ⇒ str

d = 10 ⇒ int

a = 10

b = 20

print (a+b) # 30

⇒ a, b = 10, 20 ⇒ packing / unpacking

print (a+b) # 30

python

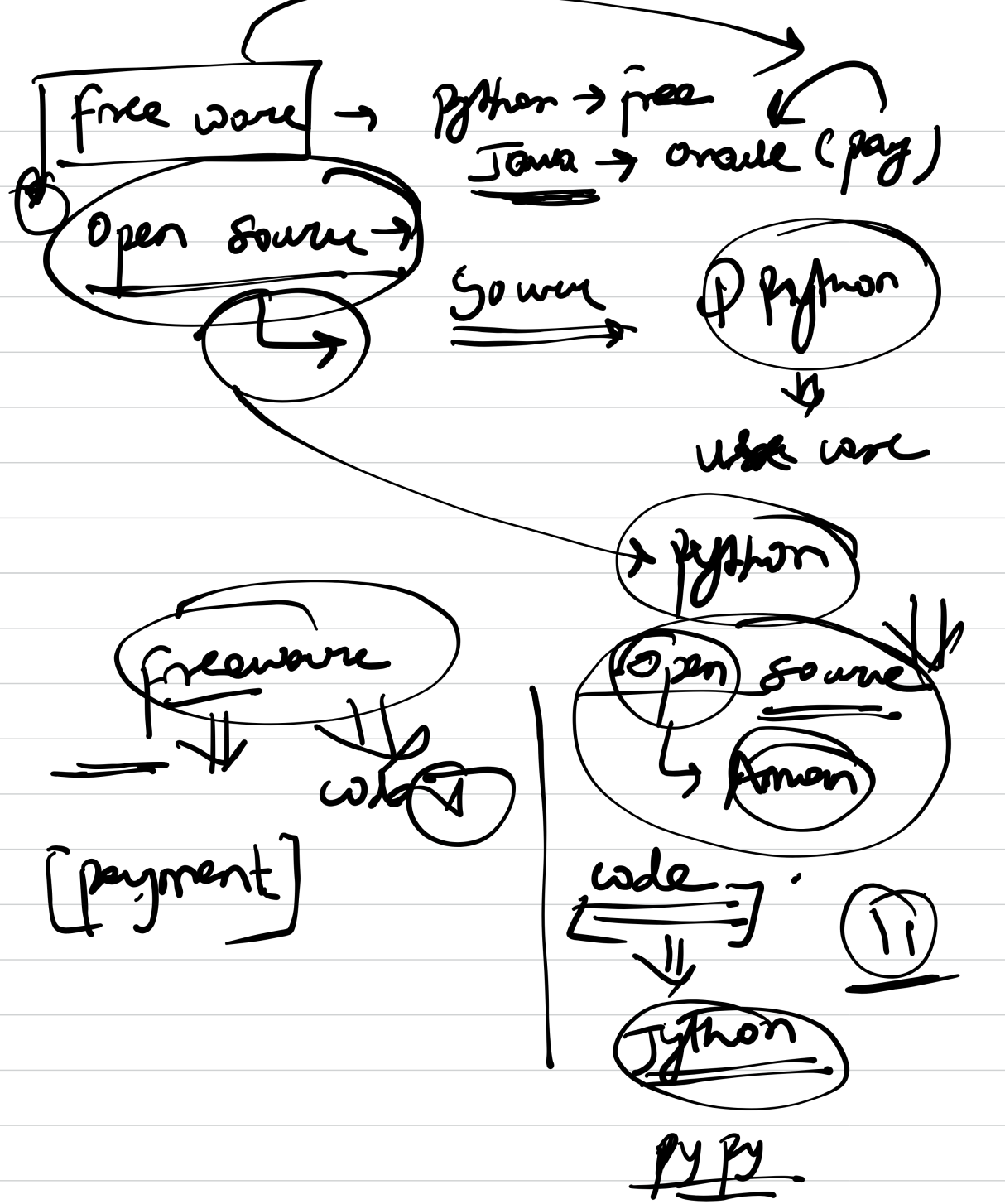
a, b, c, d = 1, 2, 3, 4

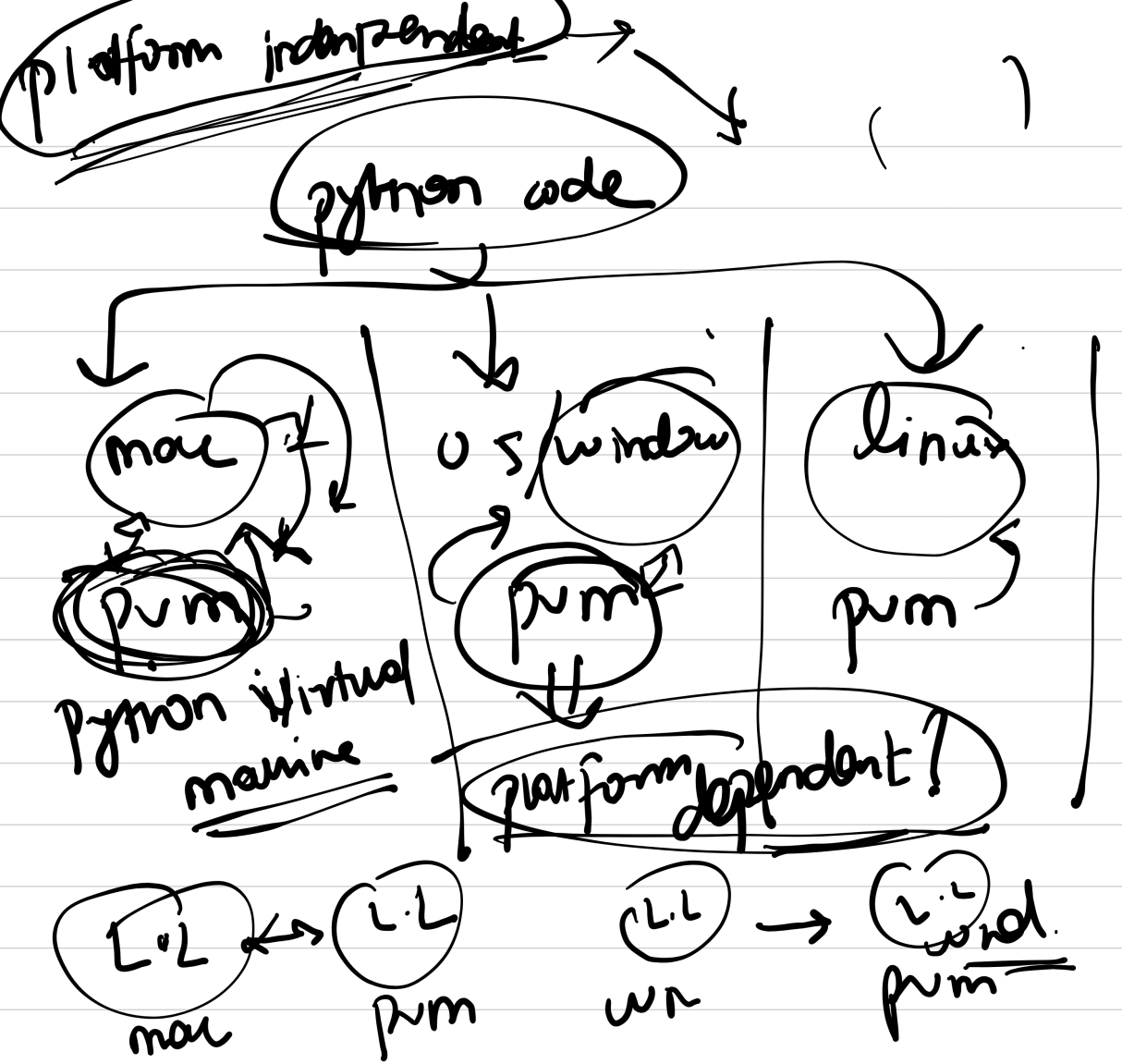
#8

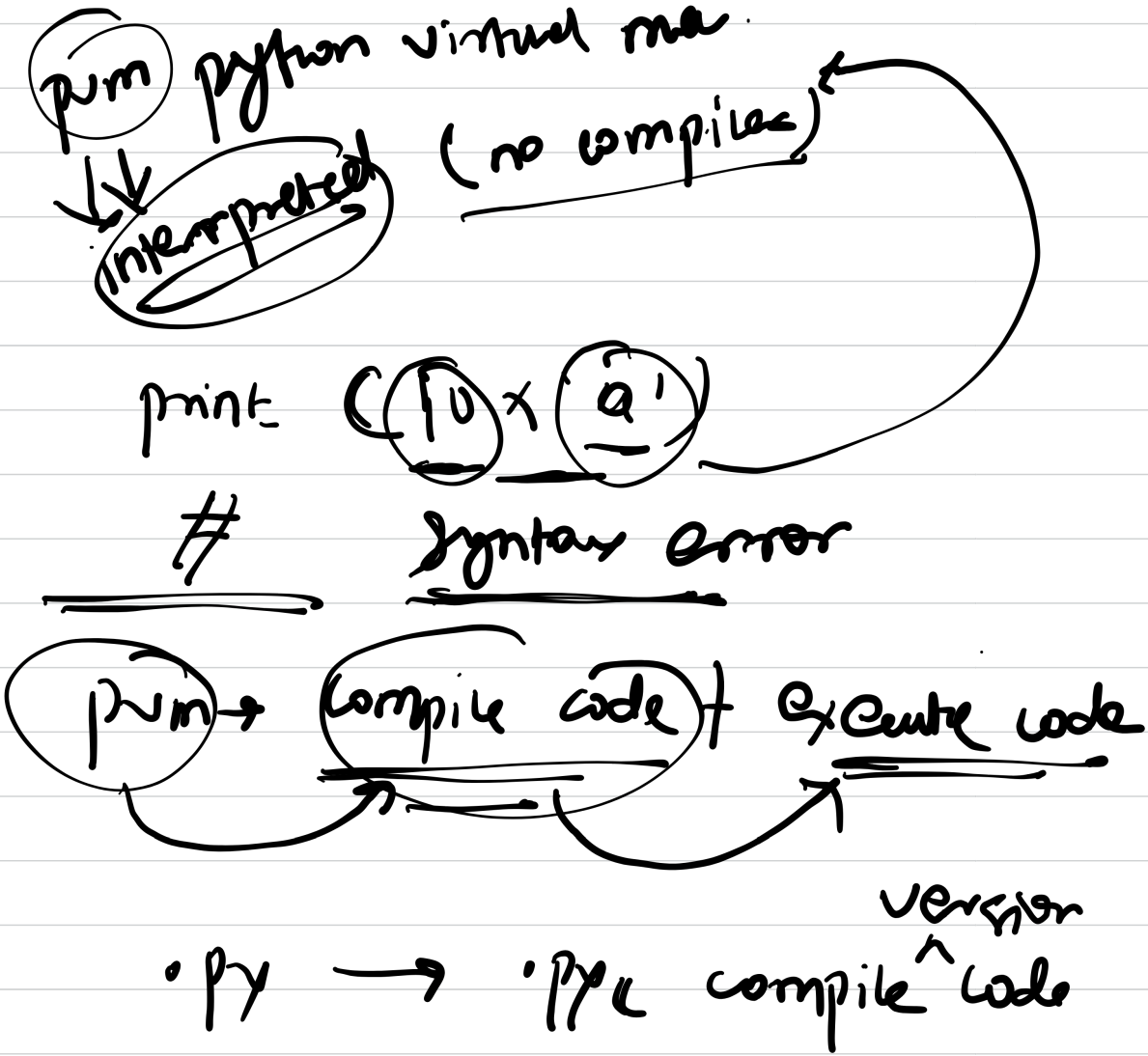
Features of python:-

1) Simple & easy to learn. ✓
2) Free ware & open source ✓
3) H.L. ✓
4) dynamically typed ✓ oracle
5) procedure / object oriented ✓
6) extensible library. ✓

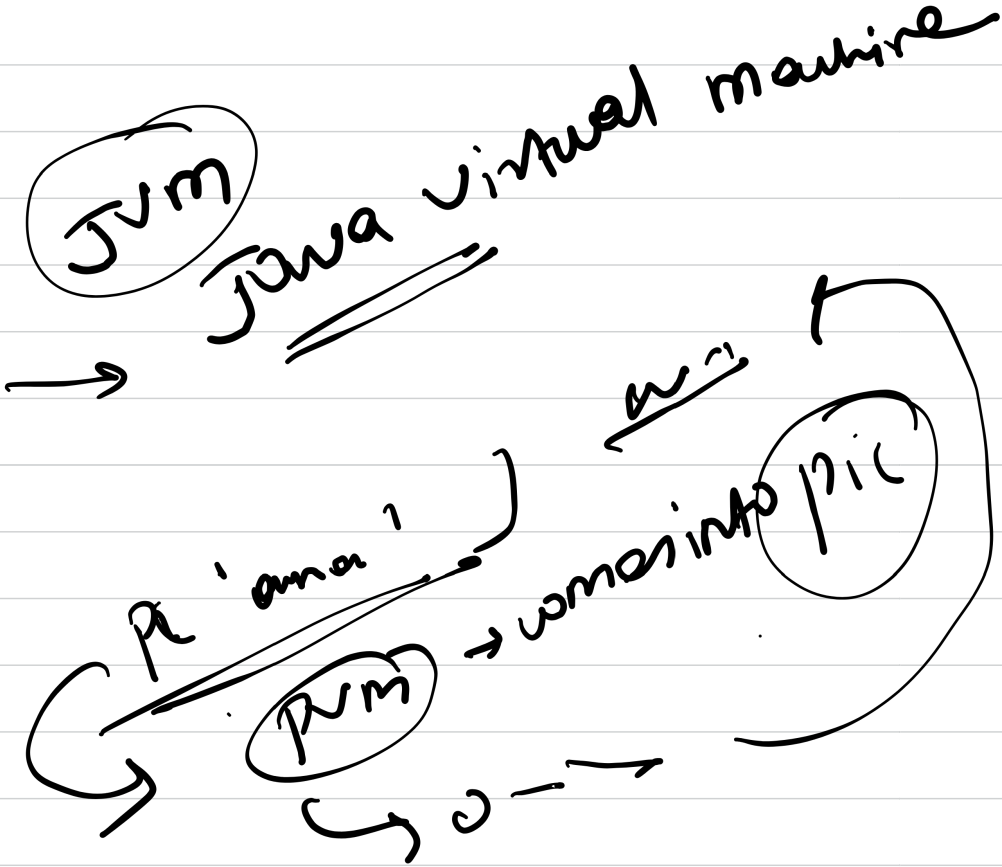
oops → If everything can be done using object (oops) why procedure ?

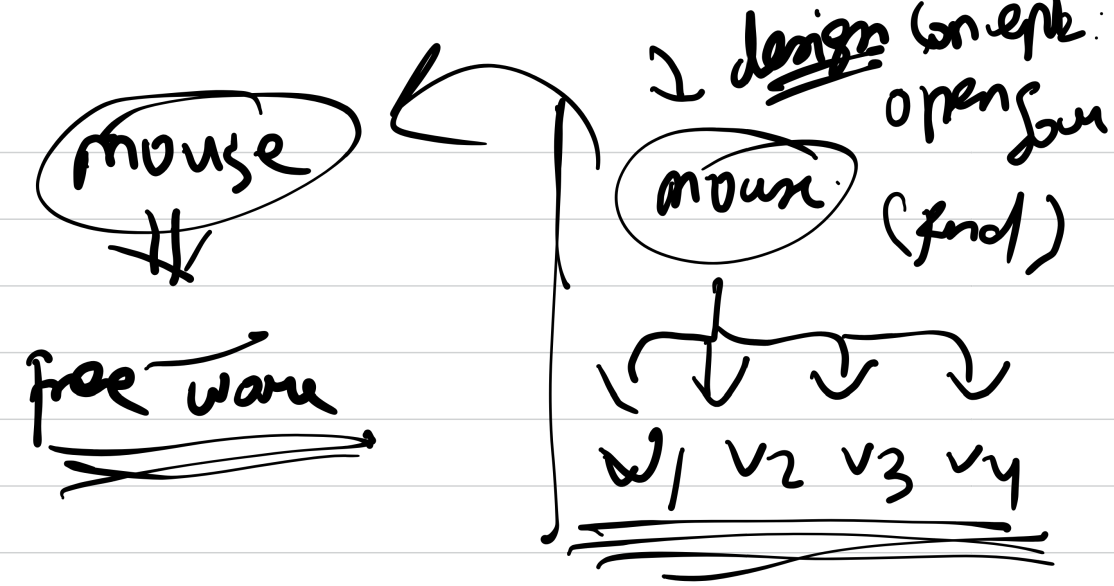




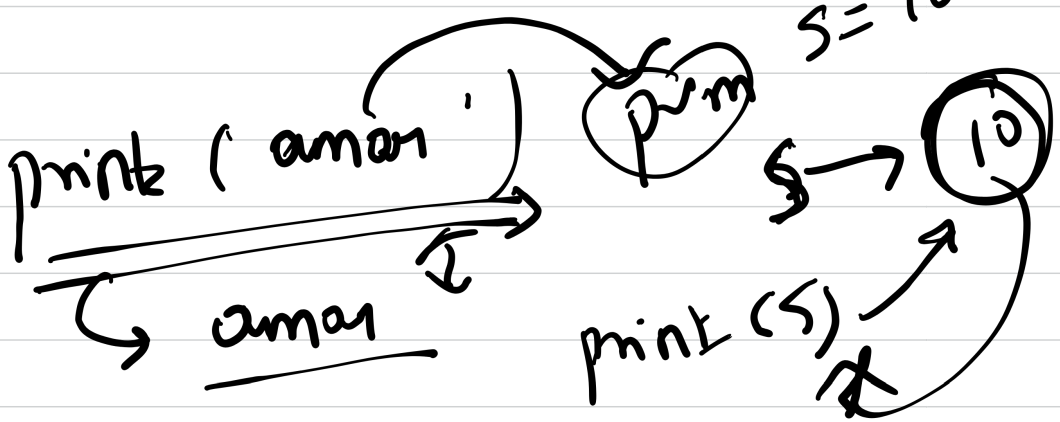


extensive libraries



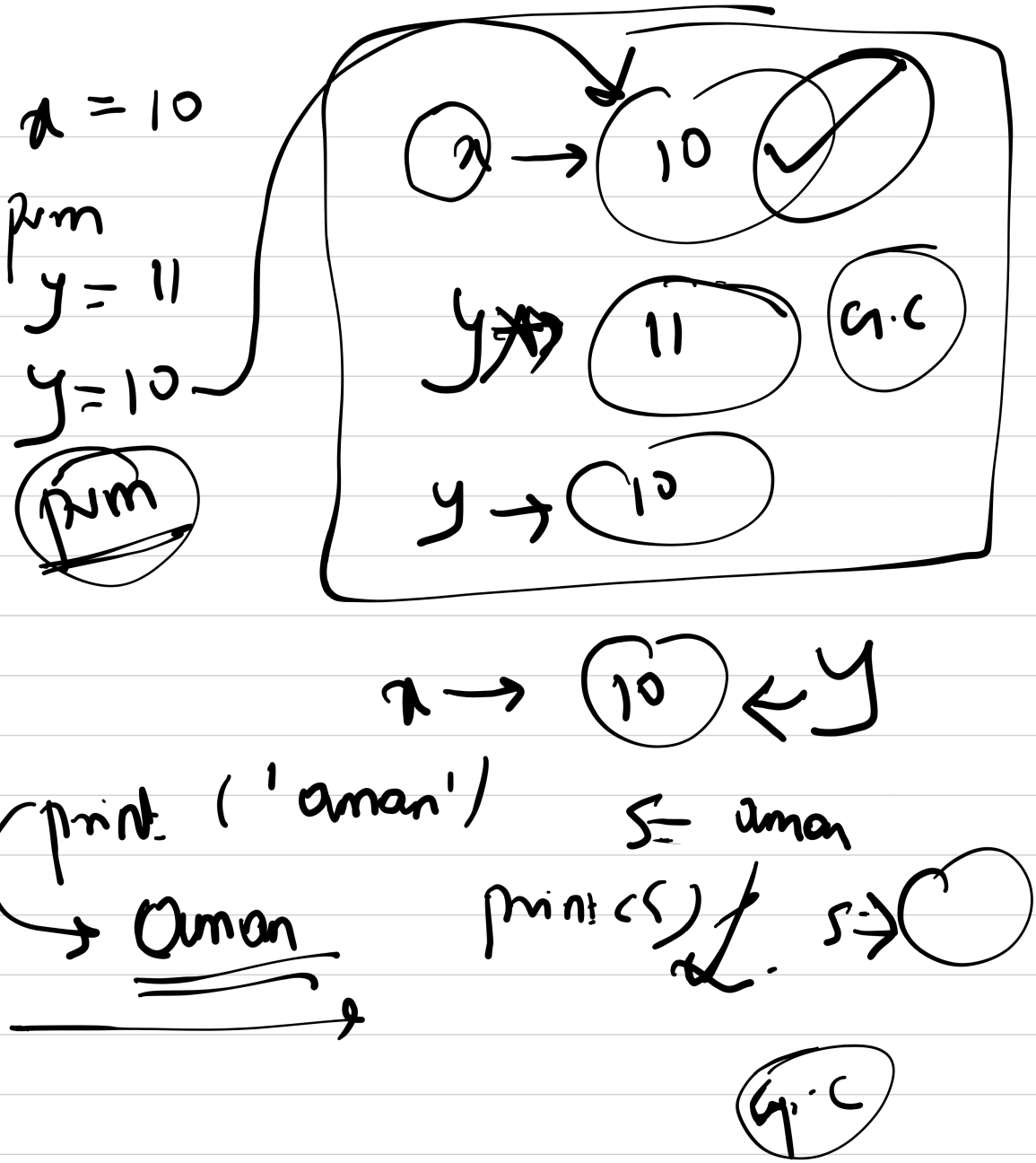


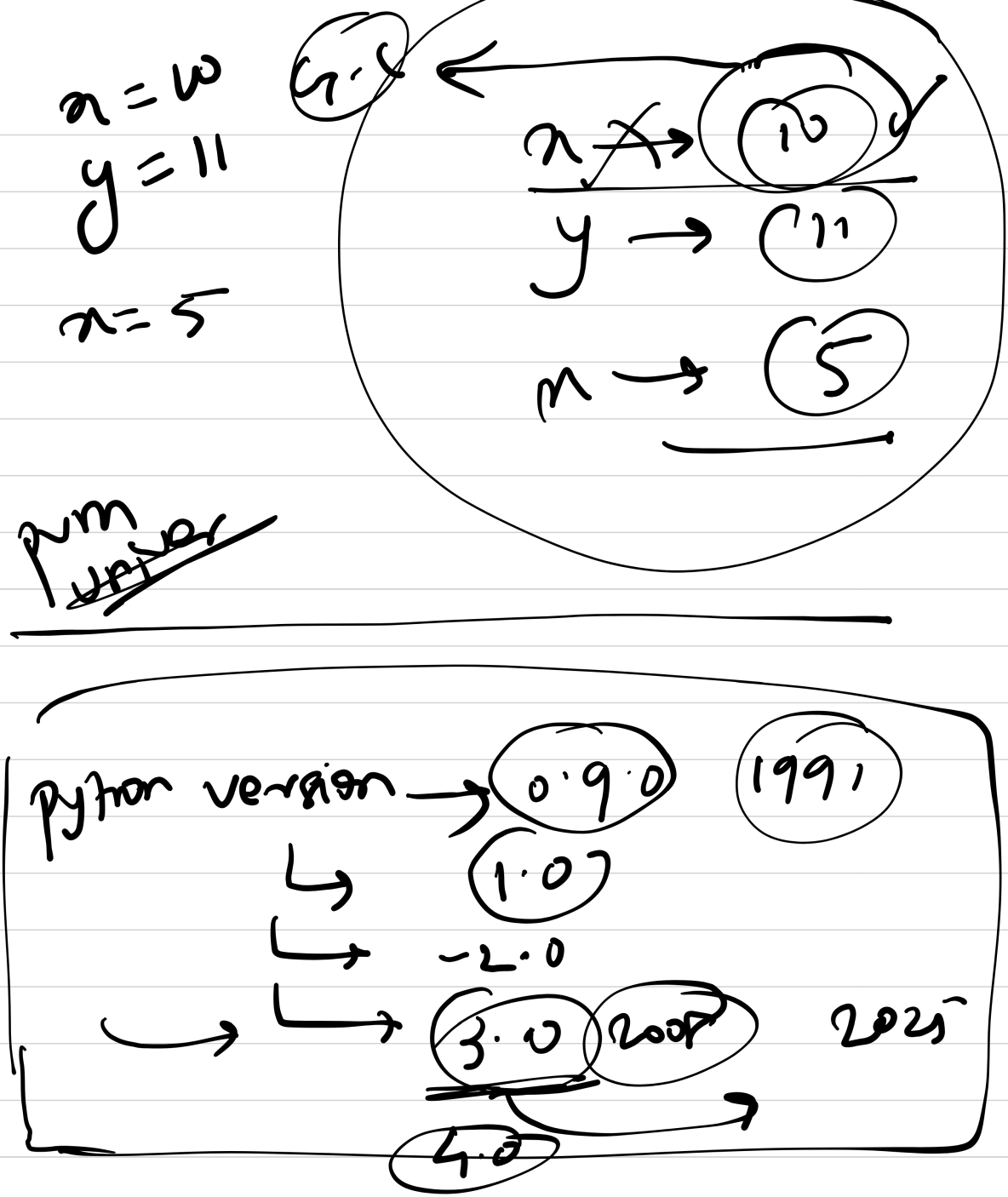
Python works internally!

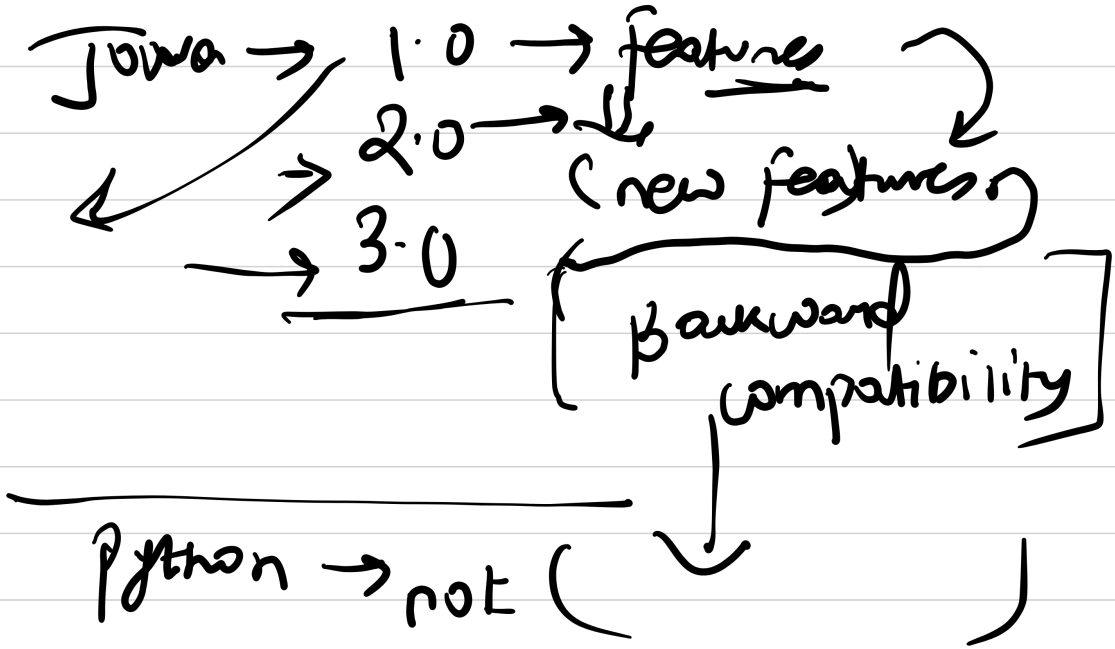


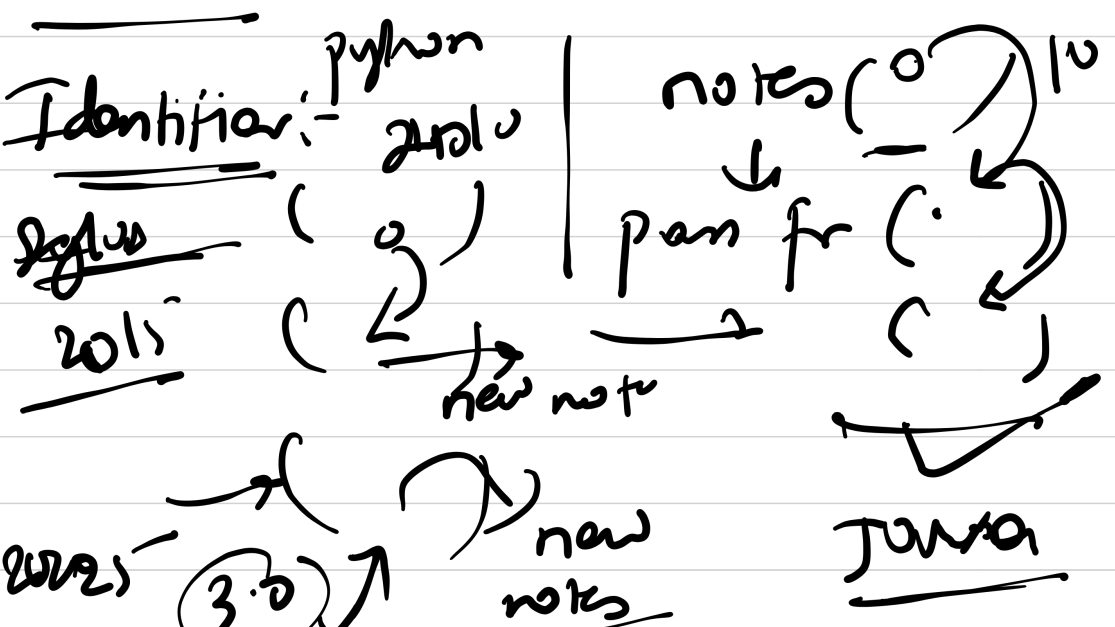
x = 1

Note: In Python everything is treated as obj









Identifier →

a to z
A to Z
0 to 9
_ (underscore)

} rules for identifier

→ Gaman = " " Error

amang8 = " "

→

aaaaaa ——————

——— aaa) = 10

Print ( ) # 10

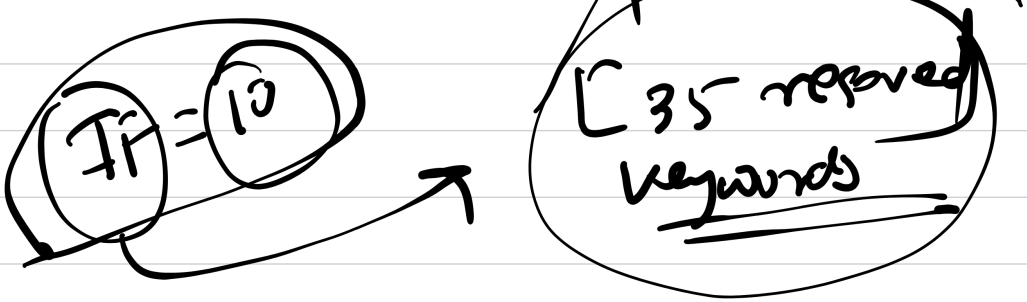
If = 10 will it work? No

then why?

keywords → 35

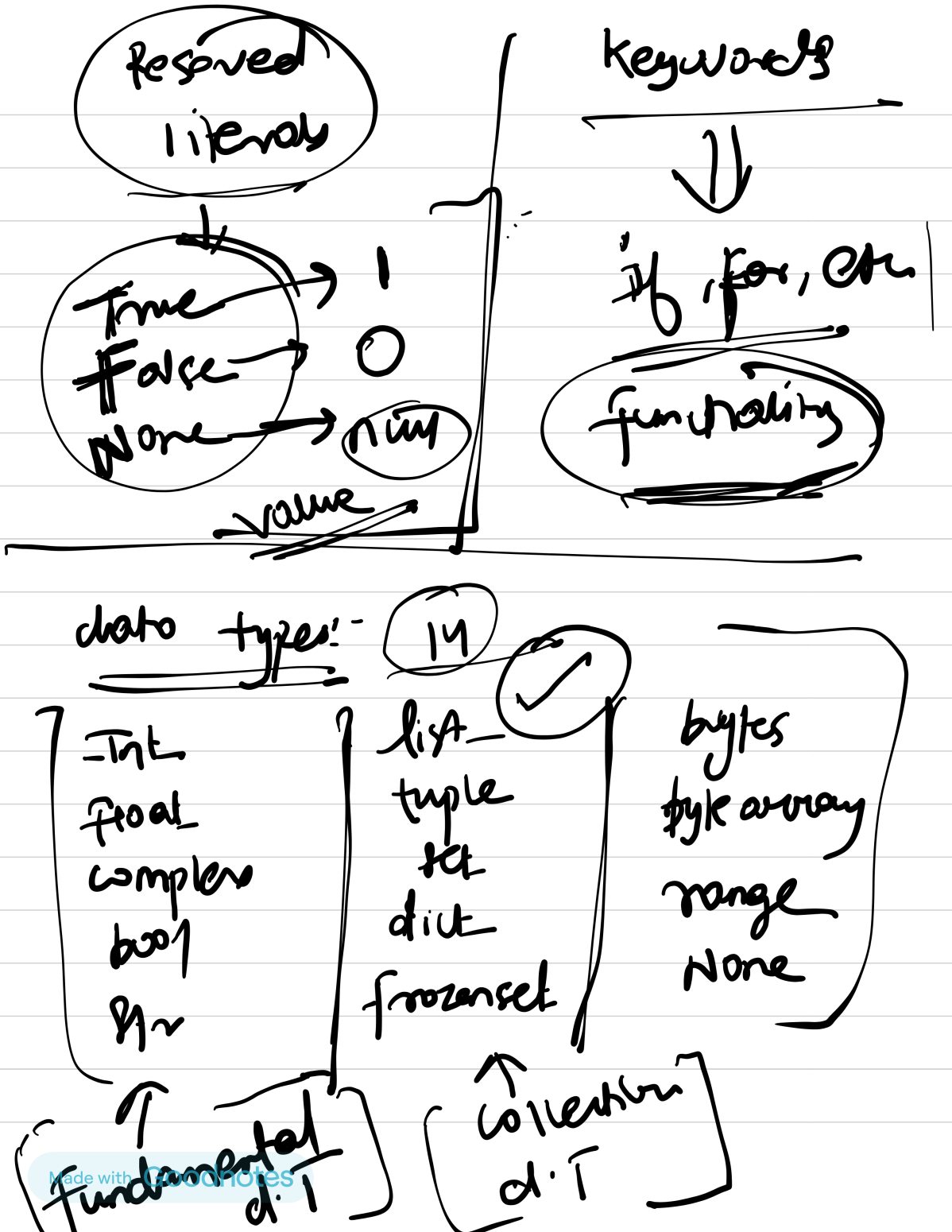
import keyword

keyword.kwlist



aman = 10

35 keywords


Reserved literals
True
False
None
1
0
NULL
value
Keywords
if, for, etc
functionality
data types:-
Int
float
complex
bool
Str
list
tuple
set
dict
frozenset
bytes
byte array
range
None
Fundamental d.t
collection d.t

dynamically typed

x = 10

x = int

print(x) # 10

1) Int (integral values (non decimal)

x = 10

20

30

2) float (decimal values)

x = 10.5 type(x) # class float

1) | python 2 (vs) python 3 |

Int = 10

long = 11111111

Int

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

real imaginary

4) Bool (logical values) comparision purpose

T/F

If age > 10 : → T/F

print ('younger')

Str :- (String

s = aman

s = 'aman'

s = "aman"

variable

x = Aman

x → aman

multi line string literal

```
s = """python is
easy to learn"""
```

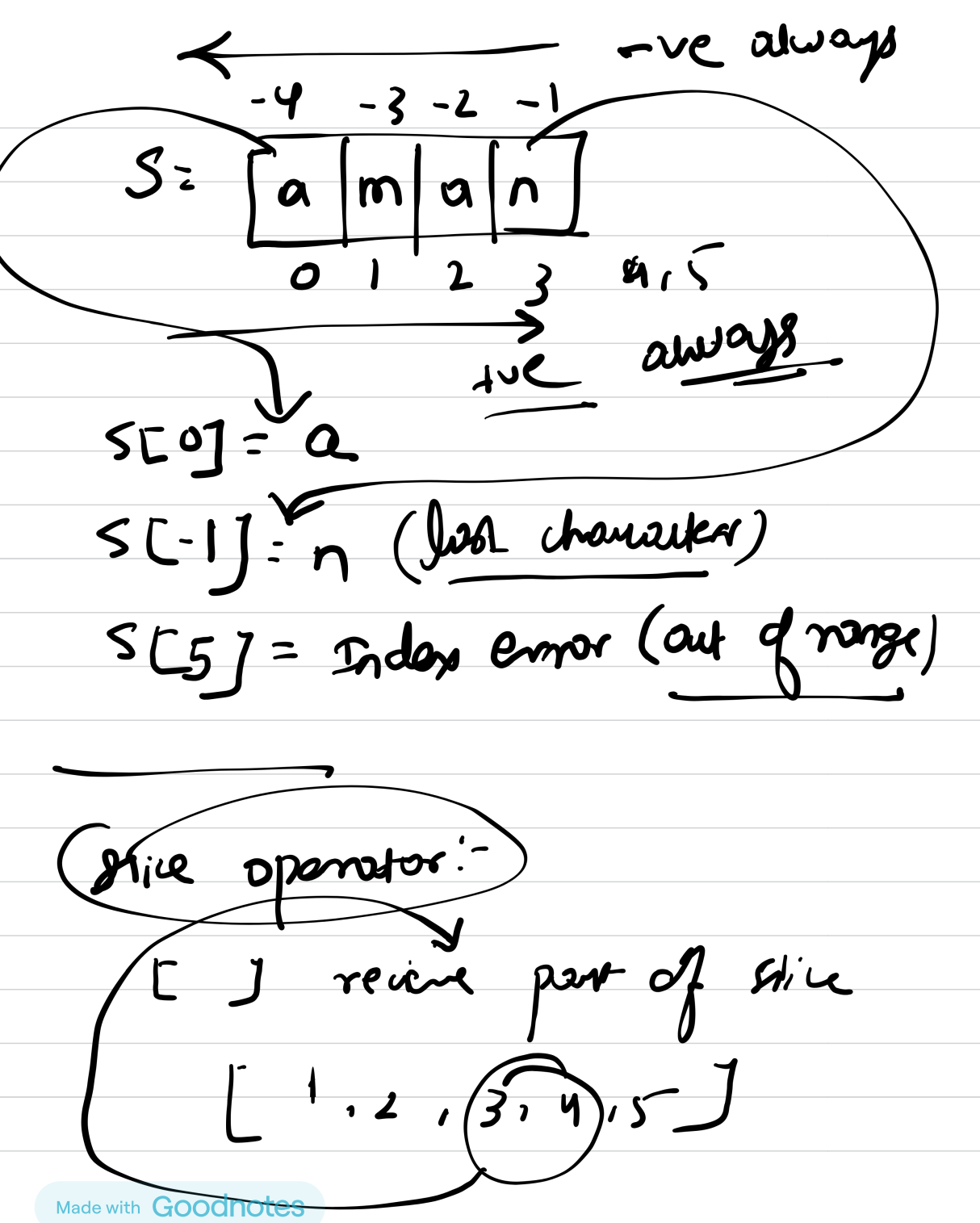
s = """ 'python' is "easy" to learn """

s = 'python is easy'

s = ''"python"'' is easy' # error

Slice operator :-

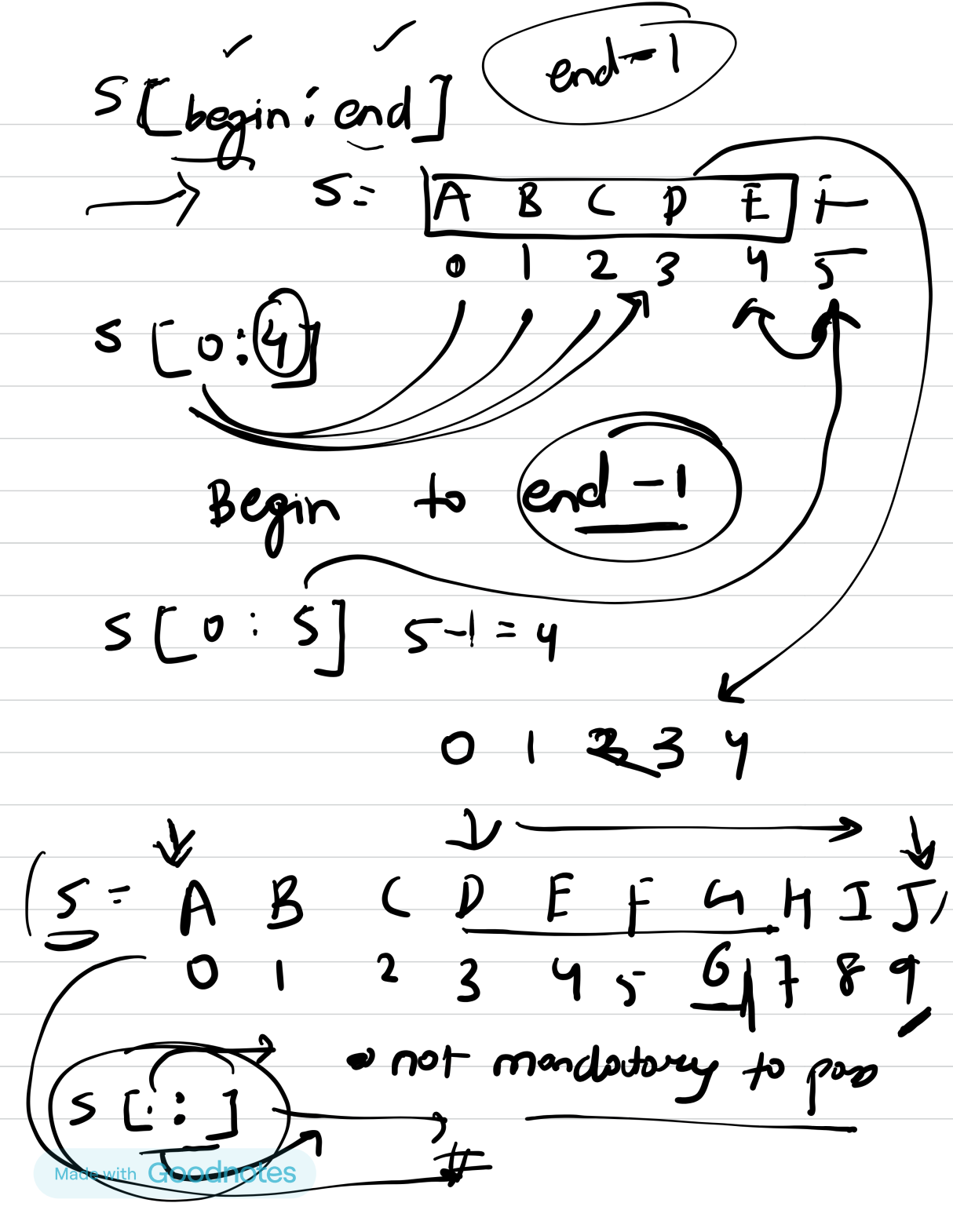
s = aman



Slice operator :-

[ ] return part of slice

[1, 2, 3, 4, 5]


s[begin : end]
end - 1
s = A B C D E
0 1 2 3 4 5
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
#
Made with Goodnotes

S[:] ✓ entire String

S[3:] ⇒ DE ... - J

S[:7] ⇒ [A .... F G]

7-1 = G

play

S[0:11] # Slice operator won't raise index error.

[A to J] # No error

S[2:7] 7-1 = G

S = [A B C D E F G H]
0 1 2 3 4 5 6 7 → -2 -1

s[7:2]

7 8 9 10 11 12

s[7:2] # empty/blank

no Error

s[2:3]

3-1 = 2

-1-1 = -2

s[:-1]

-2

s[2:2]

s[2] → [2:1] 2:2 2-1=1

s = empty

s = 'aman' # Aman
0 1 2 3 AmaN

s_1 = s[0].upper() + s[1:] arg/

A + man

0 1 2 3

s = amaN

s_1 = s[-1].upper() + s[0:len(s)-1]

= Nama

0 to 2

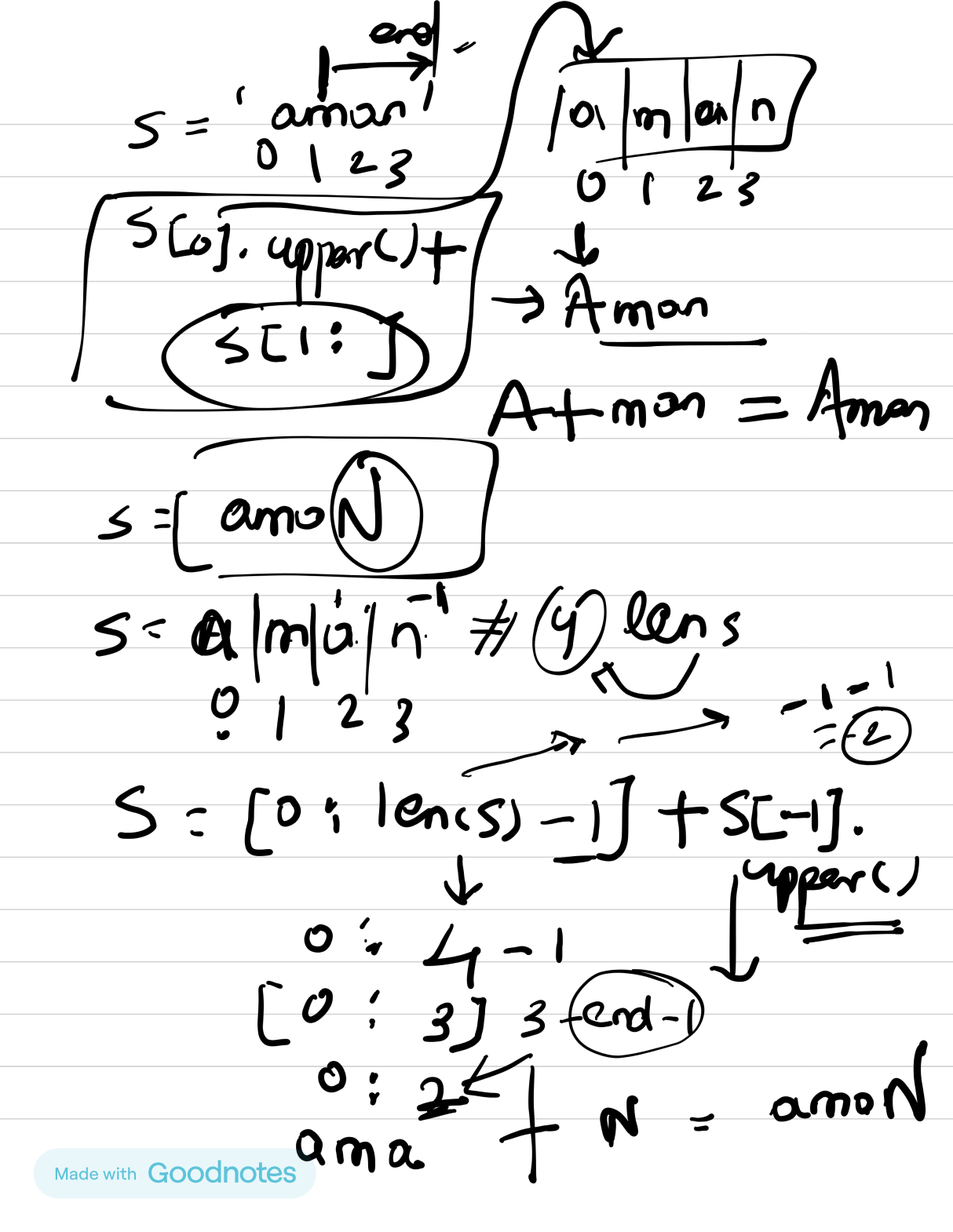
[0 : 4-1]

[0 : 3] 3-1

[0 : 2]

s[ ] + s-1

amaN



len(s) -2

4 - 2

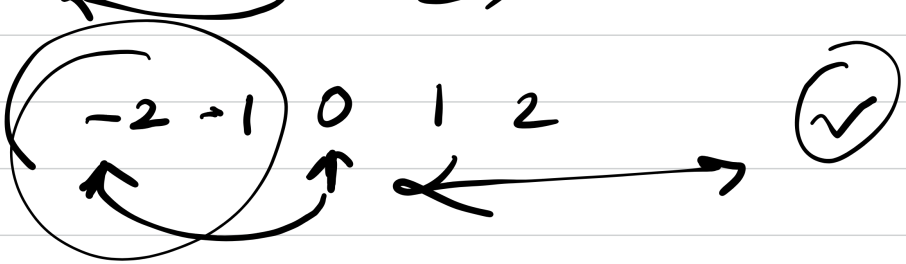
[2] → end-1

2 - 1 = 1

[0:1]

s[0:-1] → end-1 -1-1 -2

s[0:-2]

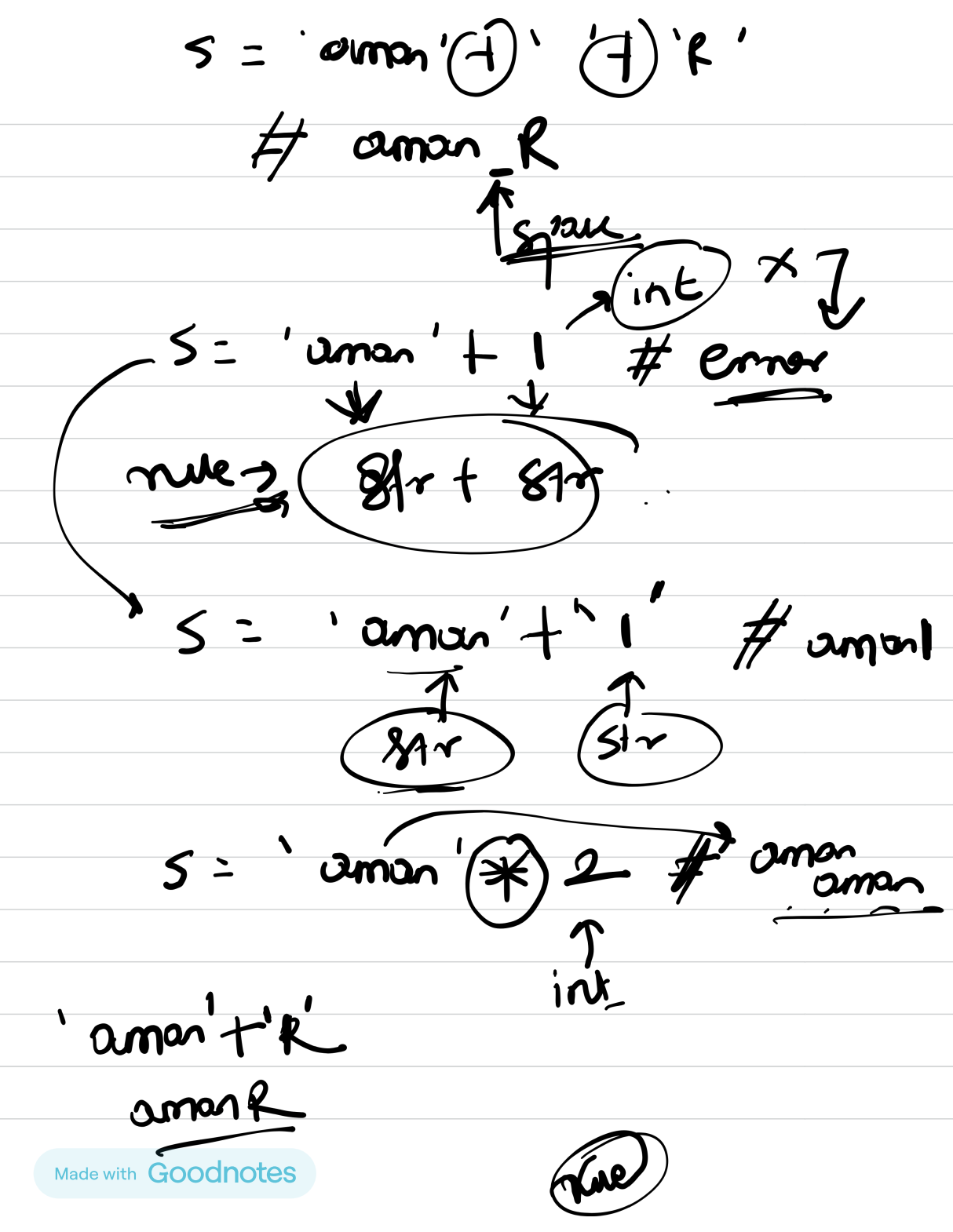


---

main operators.

s = 'aman' + 'R'

amanR



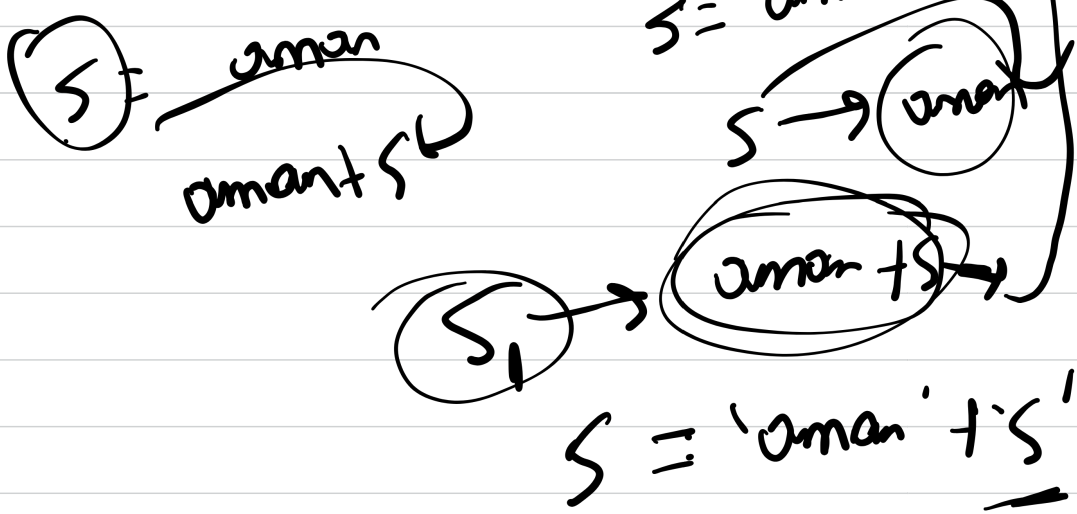
s = "aman" * "aman" # error

error

+

Str + Str

*



s = 'aman' + 's'

s1 = 'aman'

s2 = 'python'

I

$S_1 + S_2$

$S_2 =$ ' aman ' + S_1

try it !

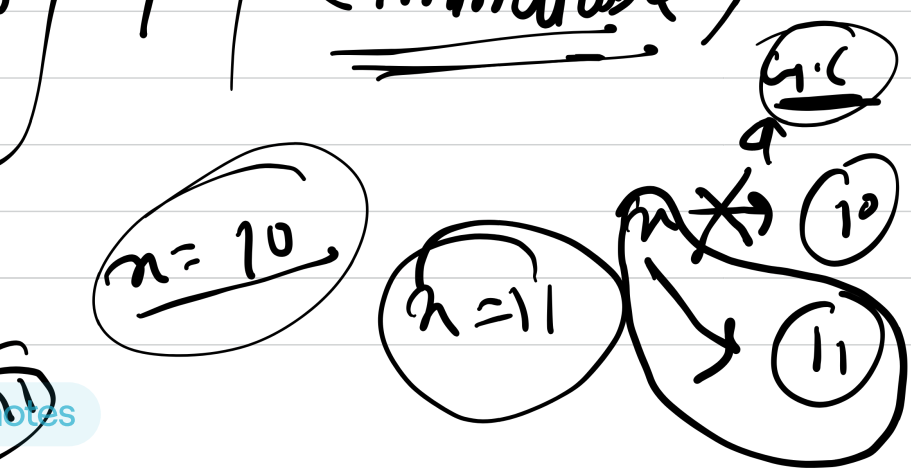
Fundamental data types vs Immutability

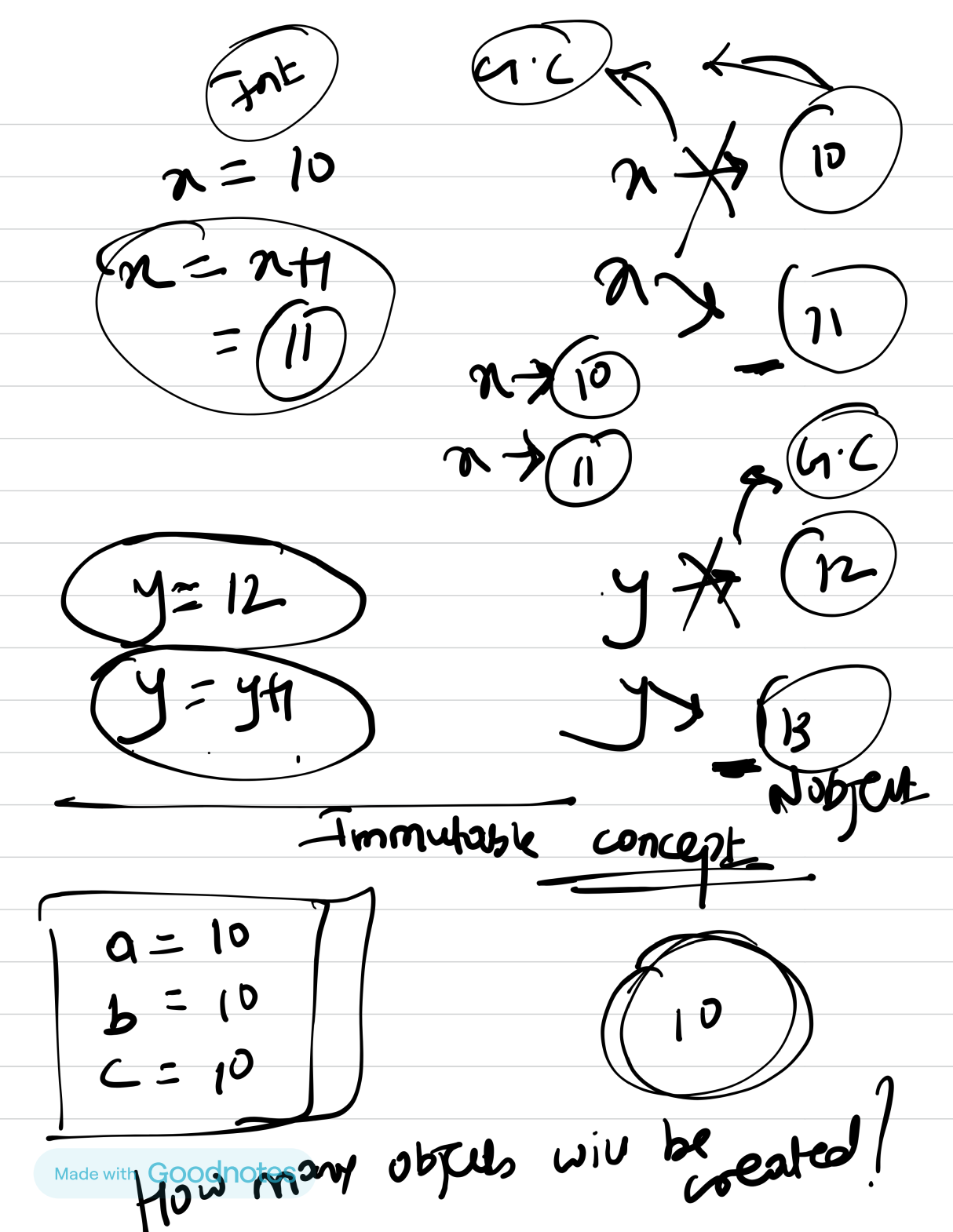
Int
float
complex
Bool
Str

fund (immutable)

n = 10

n = 11

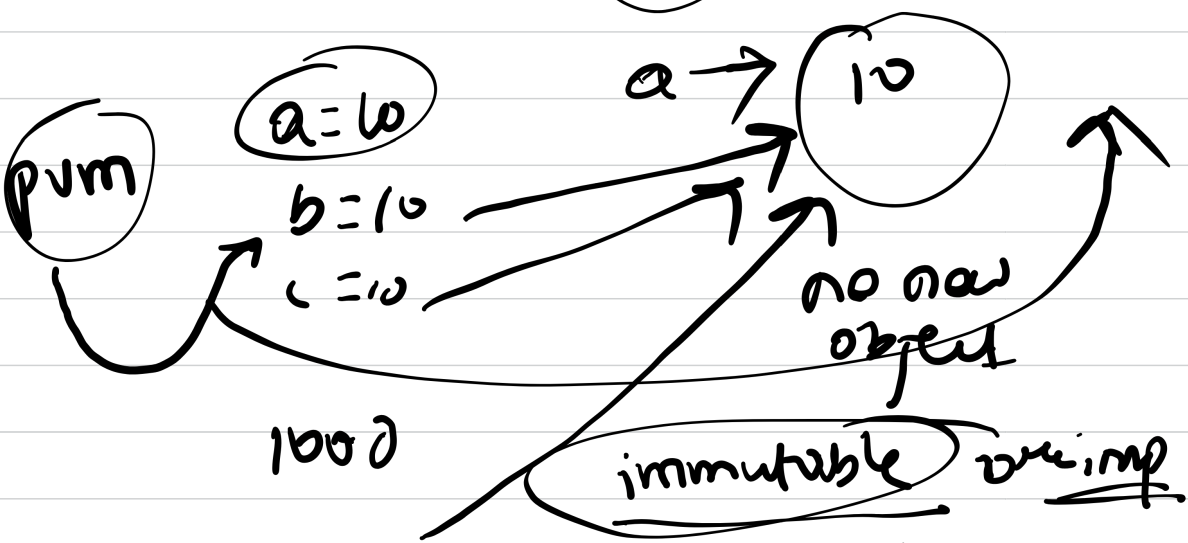



int
x = 10
x = x+1
= 11
G.C
x
10
x
11
x → 10
x → 11
y = 12
y = y+1
G.C
y
12
y
13
Object
Immutable concept
a = 10
b = 10
c = 10
10
How many objects will be created!

3 vs 1

✗ ✓

already existing



immutable string

1) memory utilization will be improved

2) object creation is costly → no new object created

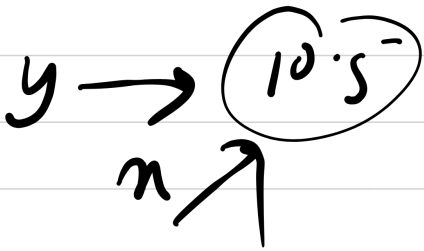
enhanced performance

vs

mutable

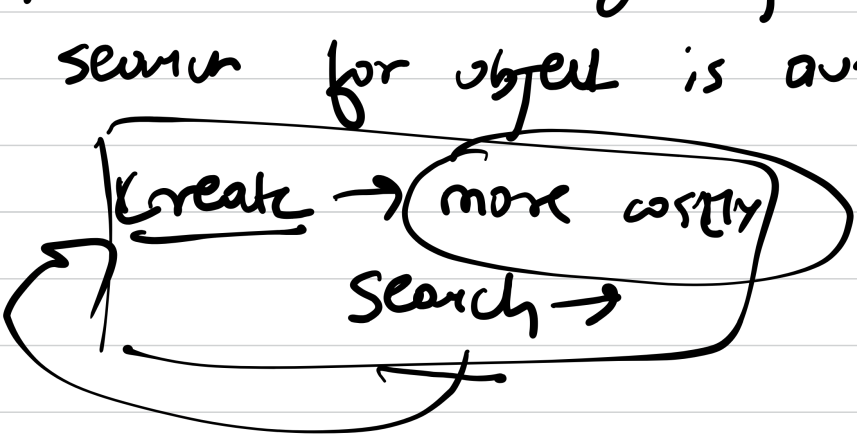
- $x = 10 \cdot 5^{-}$
  $y = 10 \cdot 5^{-}$

discuss later



Pvm reuse existing objects

search for object is available/ not.

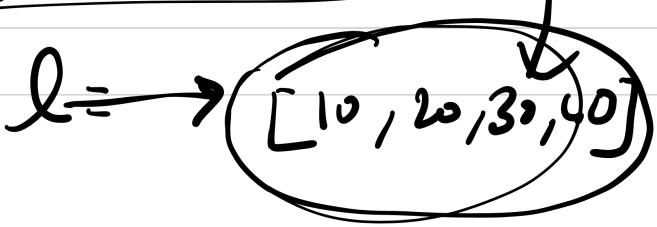


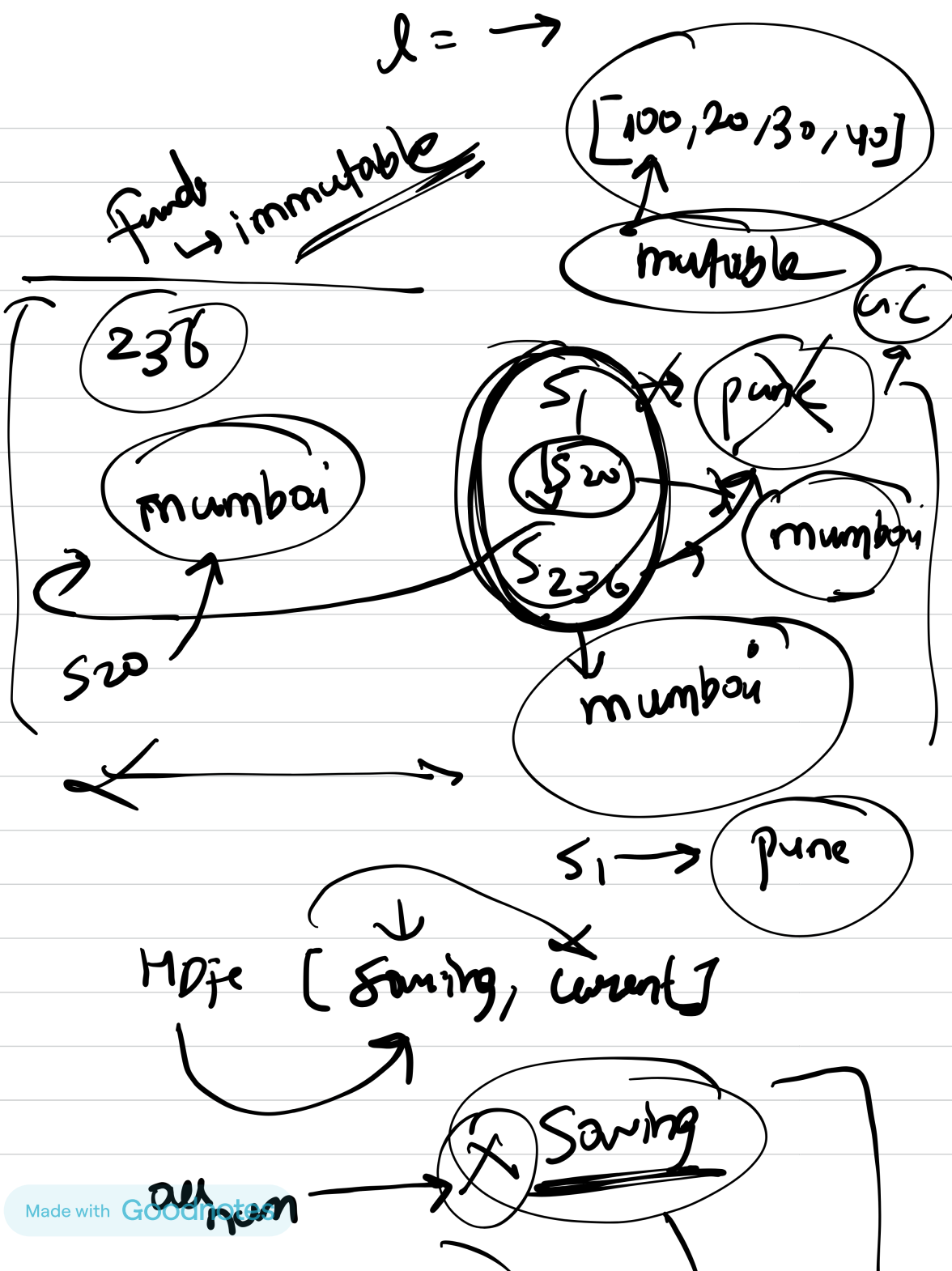
mutable

$l = [10, 20, 30, 40]$
0 1 2 3

$l[0] = 100$

print(l) [100, 20, 30, 40]





Immutability

counter

tuple

a = (1, 2, 3)

b = (1, 2, 3)

list

a = 10
= 10
= 10

a is b # True
false

a = 10.8

b = 10.8

float 10.0008

10
20
30

a = (1, 2, 3)

b = (1, 2, 3)

?

Installation

content

a = [1, 2]

b = [1, 2]

a is b # True

True

address of object

a == b

True

comparision ?