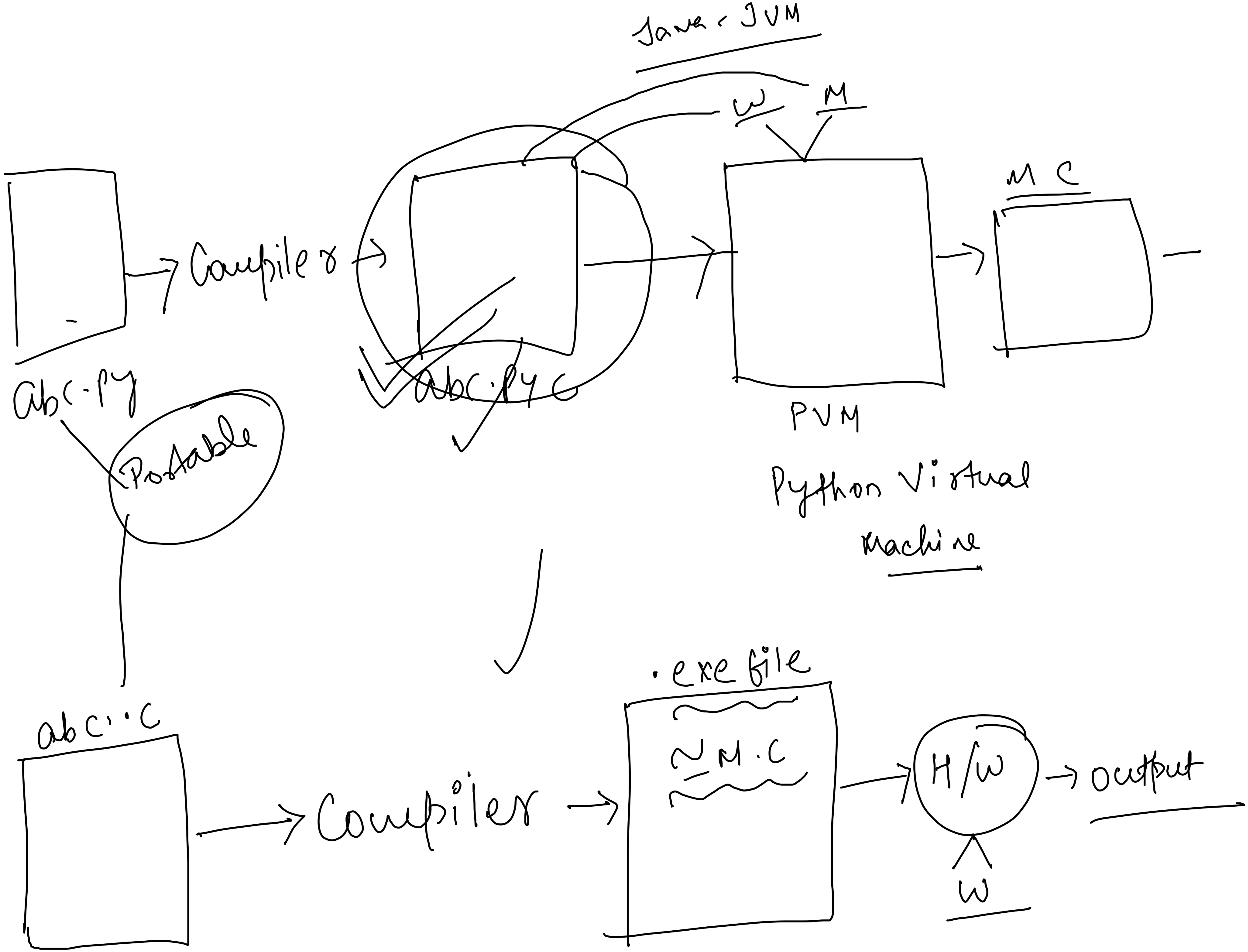
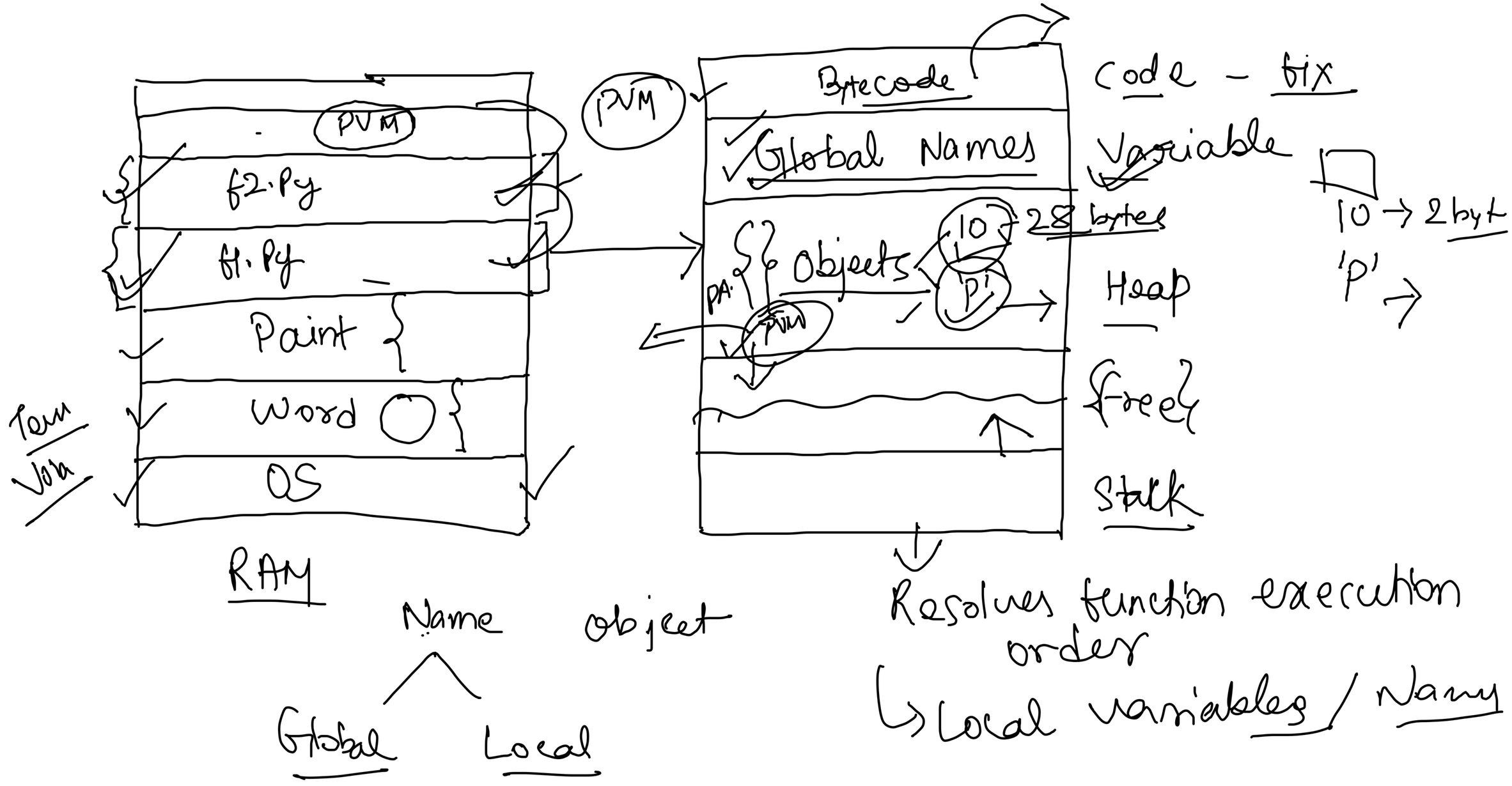
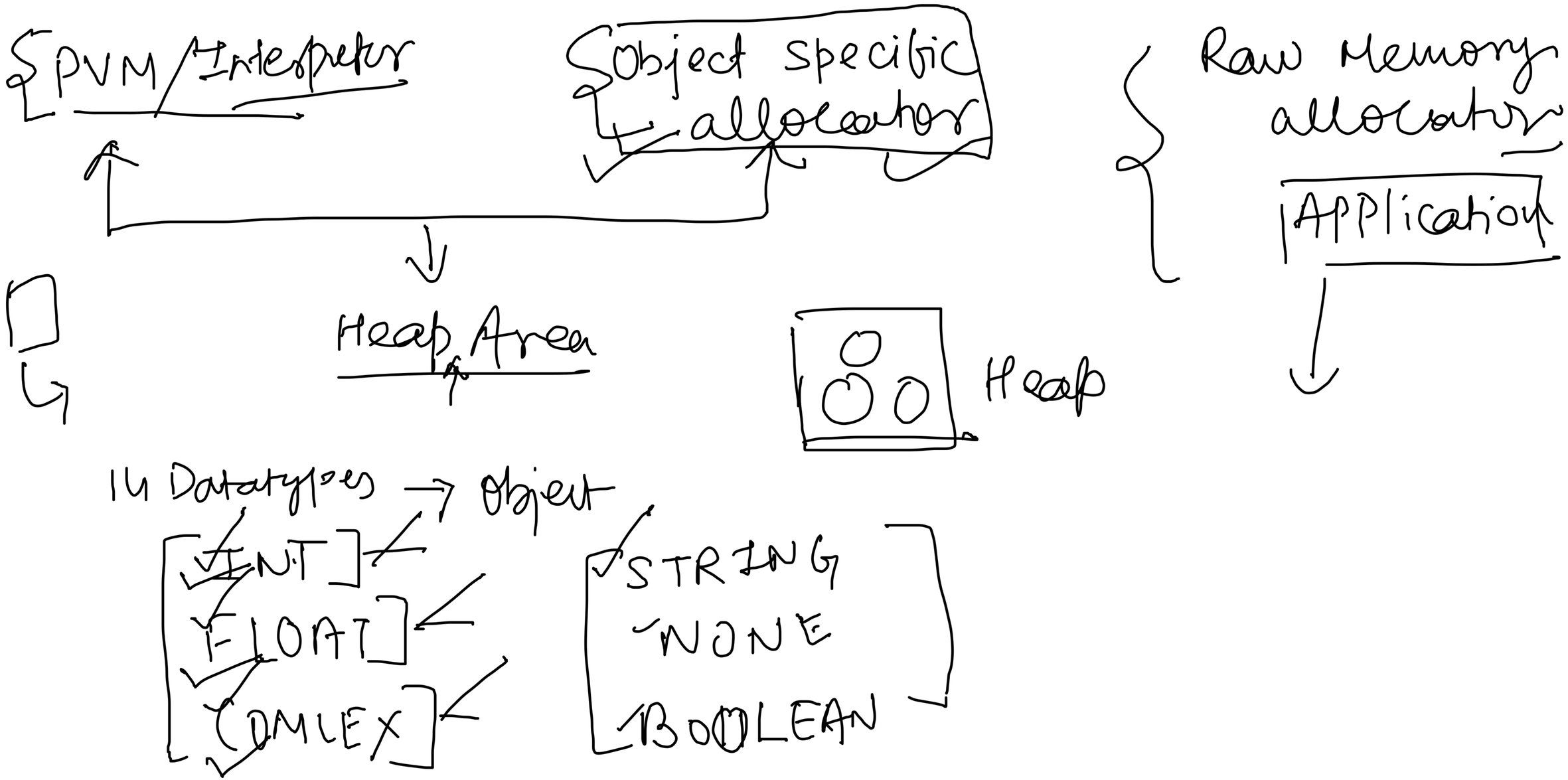

Java - JVM
W
M
Compiler
abc.py
Portable
abc.pyc
M C
PVM
Python Virtual Machine
abc.c
Compiler
.exe file
M.C
H/W
output
W

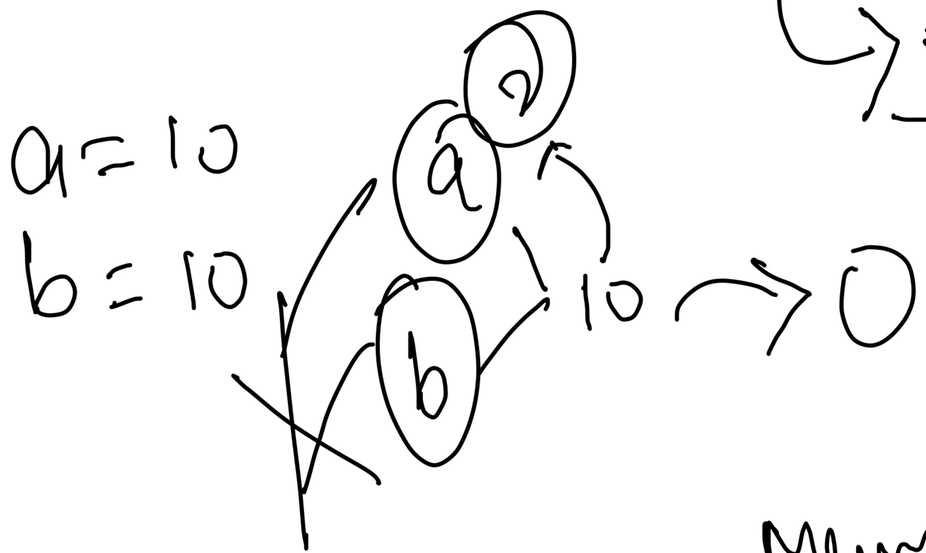

PVM
f2.Py
f1.Py
Paint
Word
OS
RAM
Tem
Var
PVM
Bytecode
Global Names
Objects
PVM
Code - fix
Variable
10 → 28 bytes
10 → 2 byt
'P'
Heap
Free
Stack
Resolves function execution order
↳ Local variables / Namy
Name
Global
Local
object


CPVM/Interpreter
Object specific allocator
Raw Memory allocator
Application
Heap Area
Heap
In Datatypes → Object
INT
FLOAT
COMPLEX
STRING
NONE
BOOLEAN

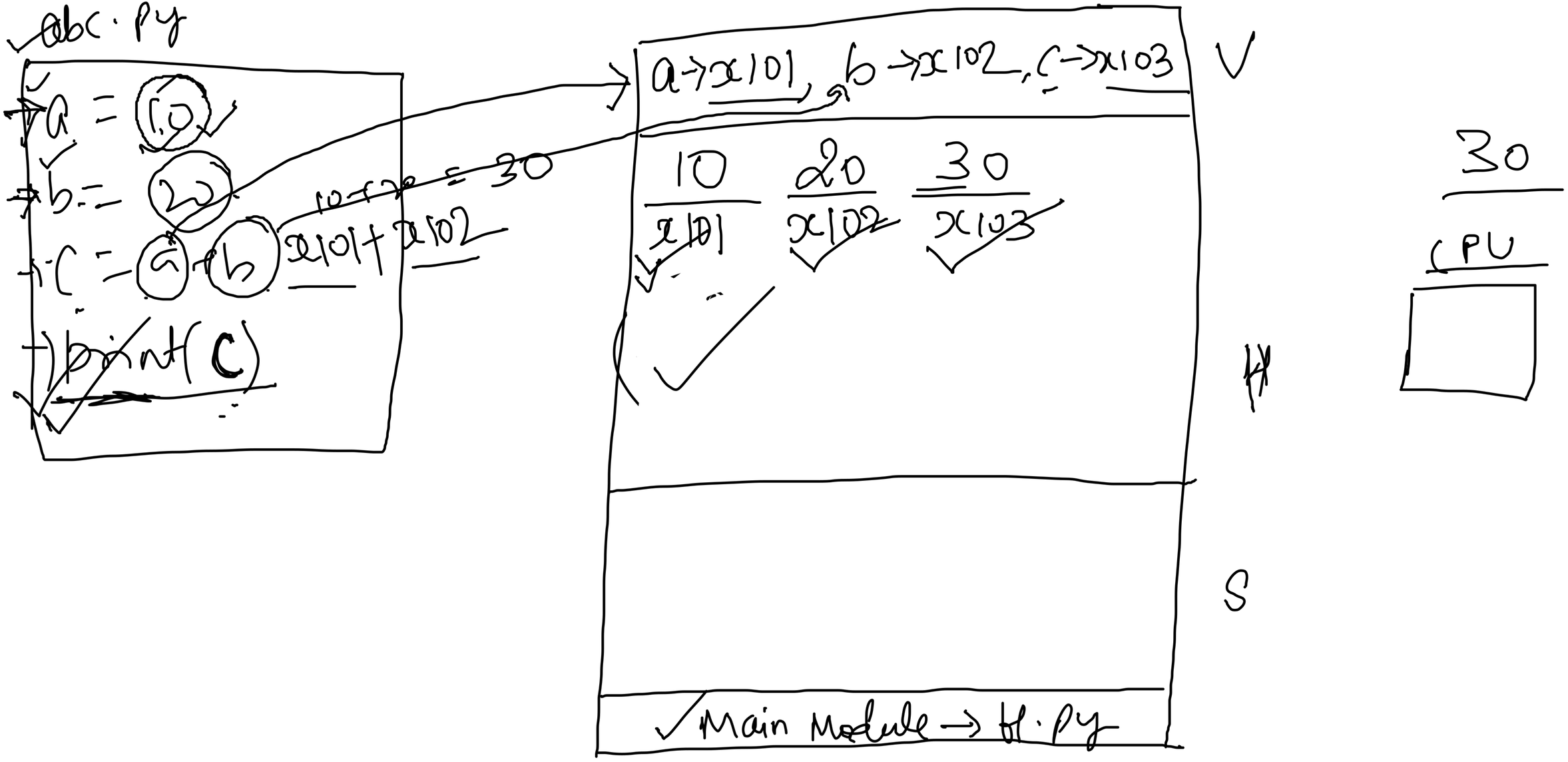
Memory Deallocation

GC → Garbage collection → Runs from time to time

↳ Reference count == 0



Memory management is automatic


abc.py
a = 10
b = 20
c = a + b
print(c)
10+20 = 30
x101 + x102
a -> x101, b -> x102, c -> x103
10
x101
20
x102
30
x103
V
H
S
Main Module -> hl.py
30
CPU

Pyth ASCII

0 127

C

a = 10

256

Pyth

a = 10
b = 10

a = 10
b = 10

a = 1000
b = 1000

a → 10 ← b

a = 1000
b = 1000

Object Reusability

- → INT → -5 to 256
- → NONE
- → True False
- → String - A-Z, a-z, 0-9, _

PyPy → a, b, c

PyPy

Int, None, Bool, String

-5 - 256, True False, A-Z a-z 0-9 _

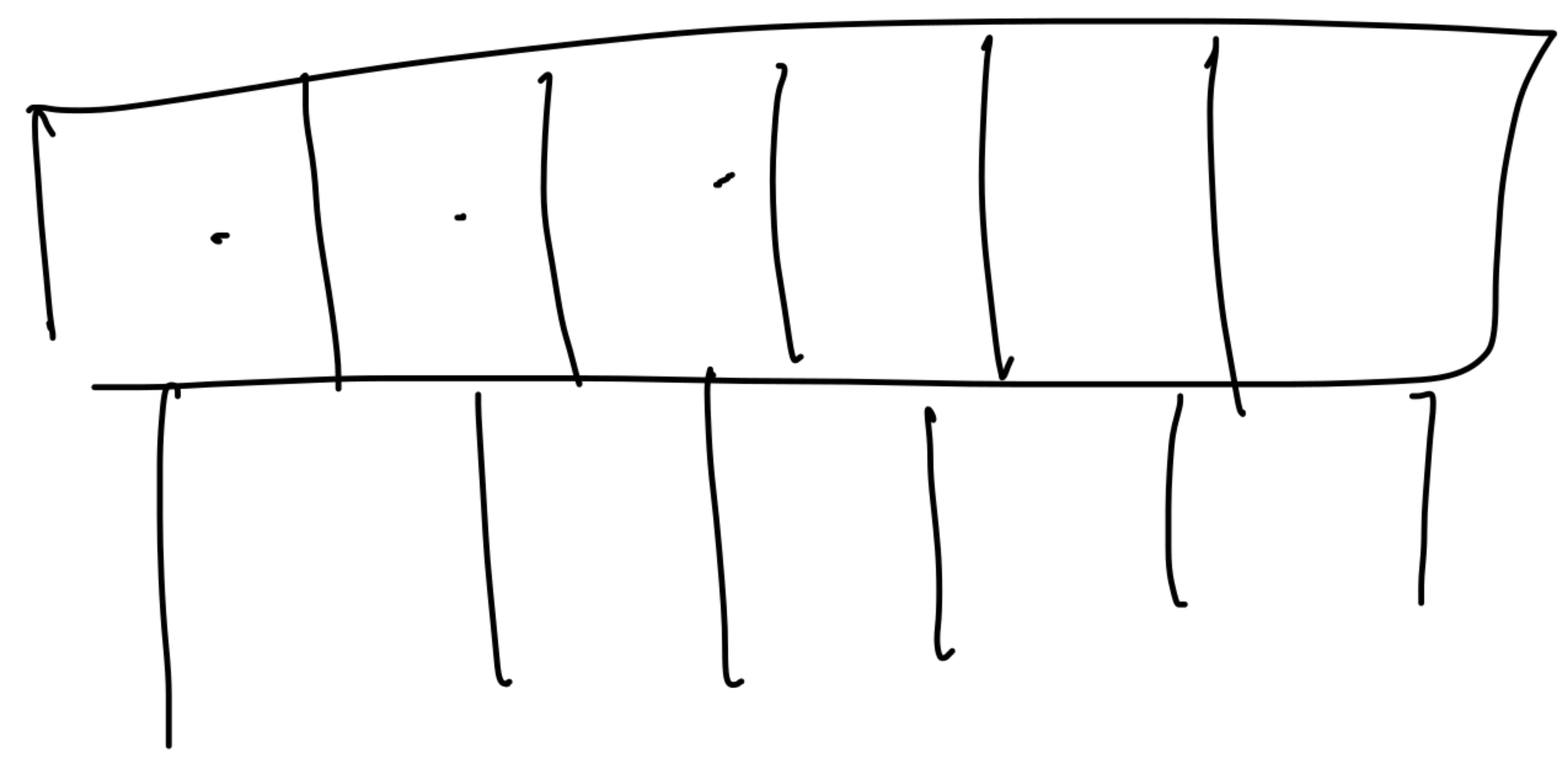
⇓

float → 0 & 1 → Infinite

Complex → a + bj

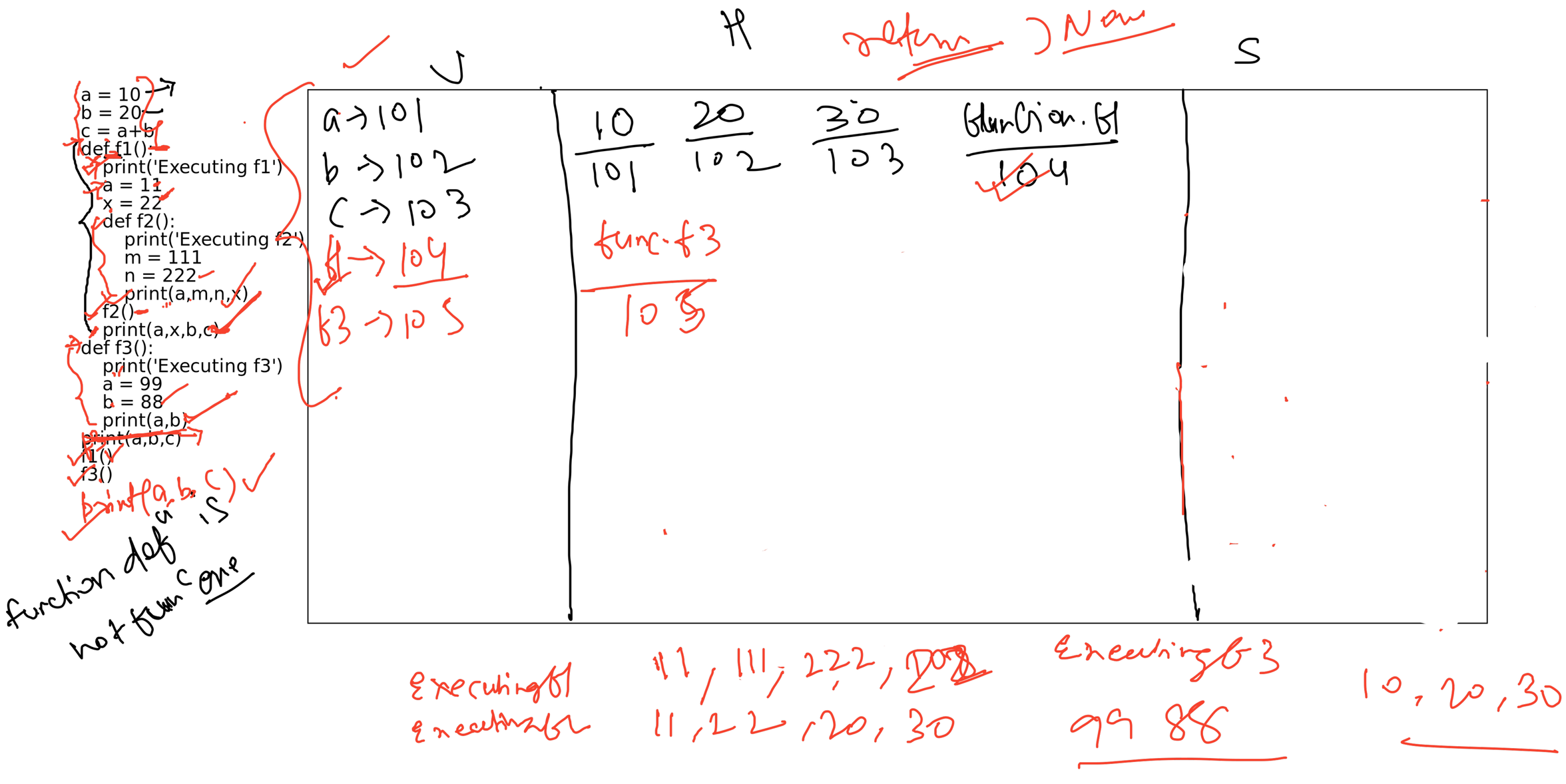
Int flo Int flo

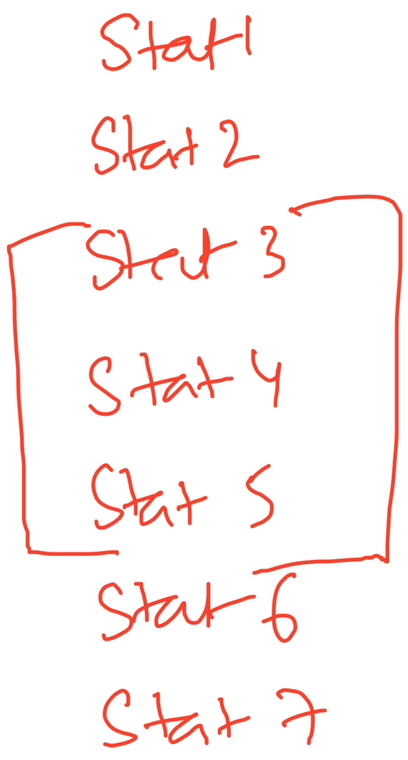
my_str = "Python"

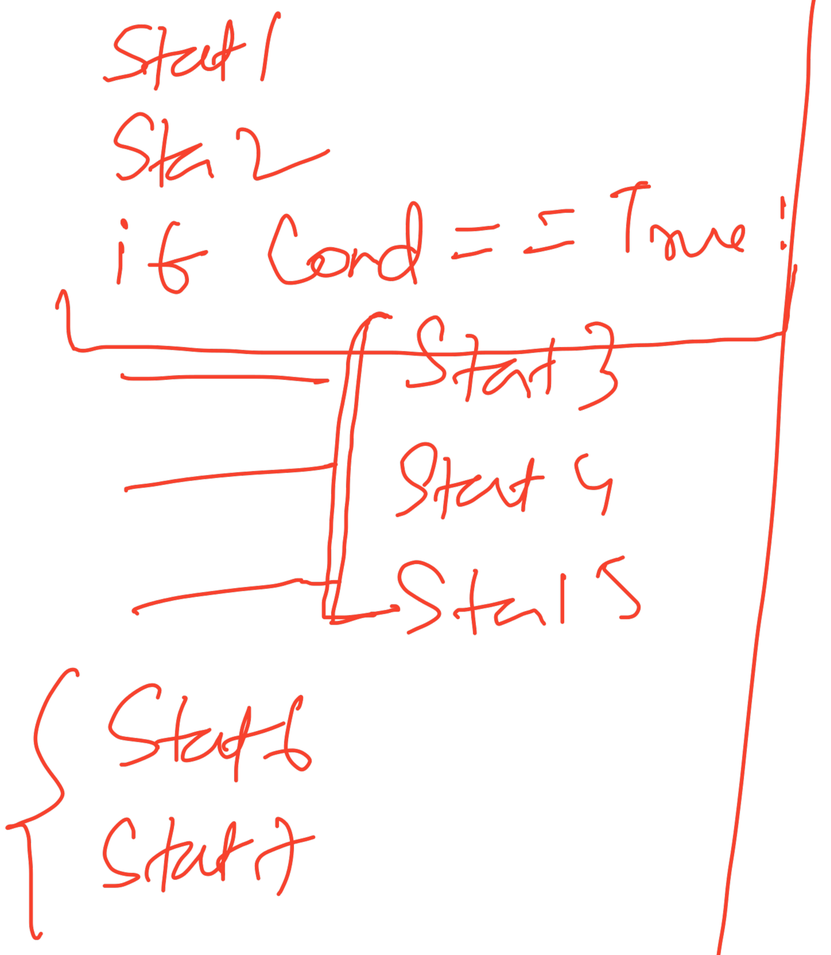


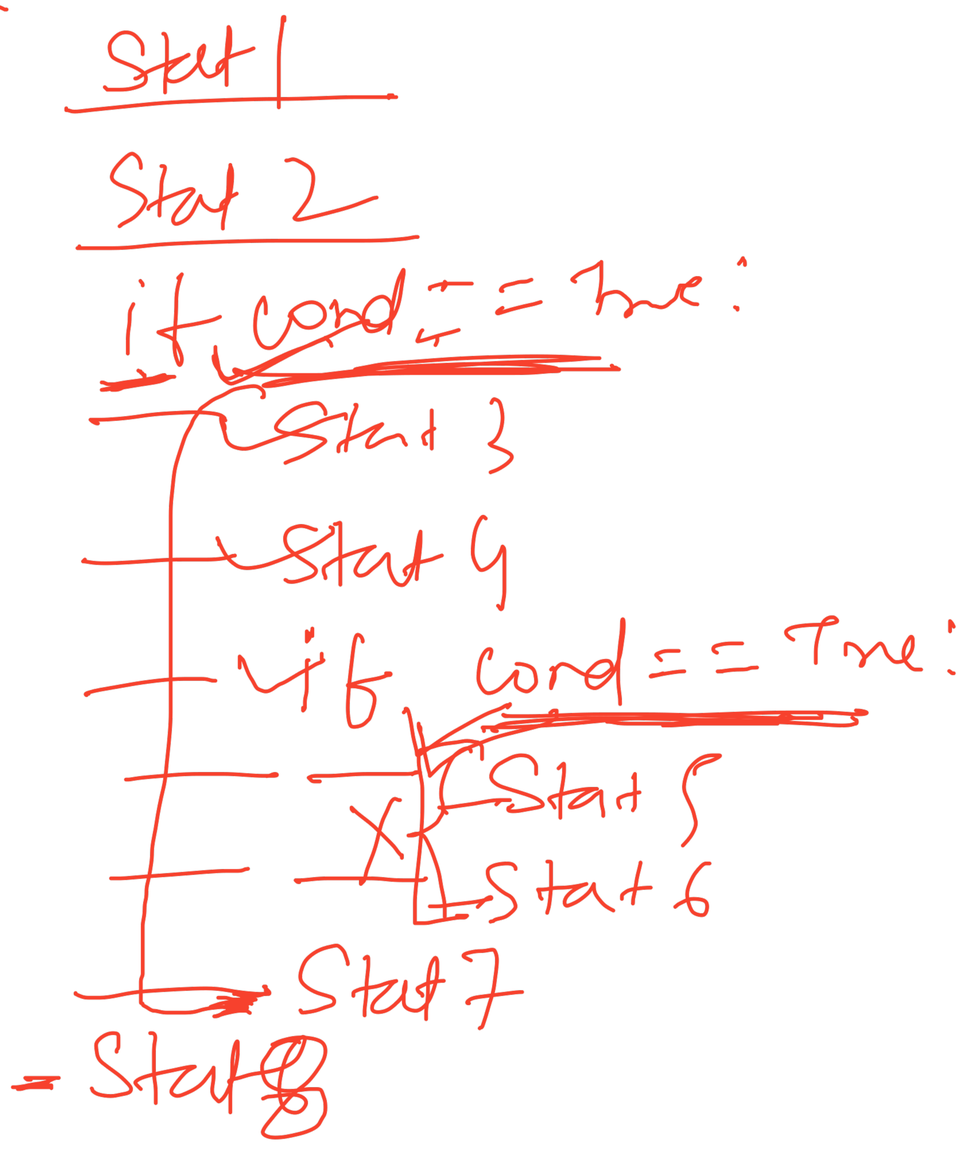
[( )]

p y t h o n


a = 10
b = 20
c = a+b
def f1():
print('Executing f1')
a = 11
x = 22
def f2():
print('Executing f2')
m = 111
n = 222
print(a,m,n,x)
f2()
print(a,x,b,c)
def f3():
print('Executing f3')
a = 99
b = 88
print(a,b)
print(a,b,c)
f1()
f3()
print(a,b,c) is function def but not function one
a→101
b→102
c→103
f1→104
f3→105
10 101
20 102
30 103
function.f1 104
func.f3 105
Executing f1
11, 111, 222, 20
Executing f3
99 88
10, 20, 30
Executing f1
11, 22, 20, 30


Stat 1
Stat 2
Stat 3
Stat 4
Stat 5
Stat 6
Stat 7


Stat 1
Stat 2
if Cond == True:
Stat 3
Stat 4
Stat 5
Stat 6
Stat 7


Stat 1
Stat 2
if cond == True:
Stat 3
Stat 4
if Cond == True:
Stat 5
Stat 6
Stat 7
Stat 8