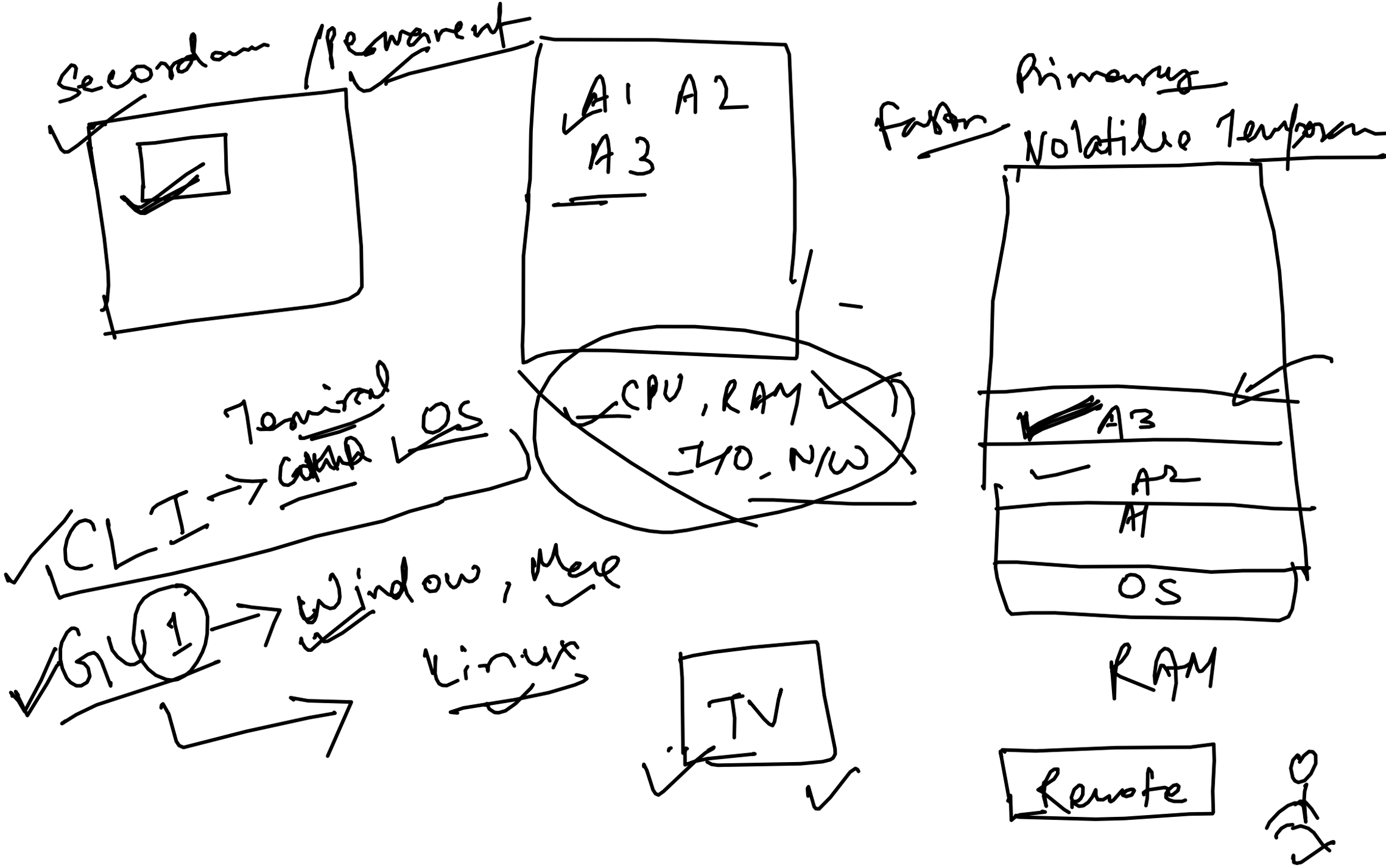
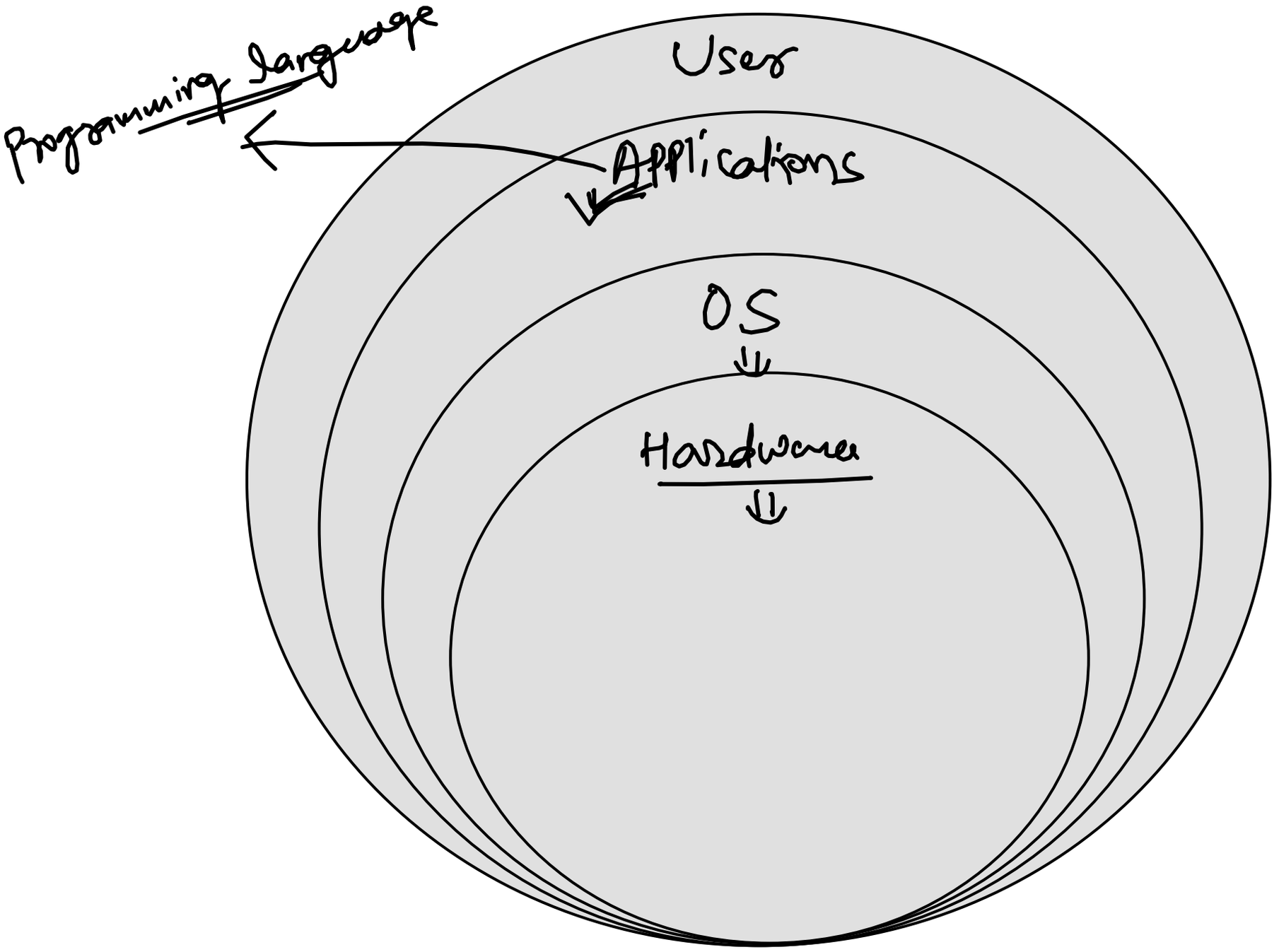

Secondary
Permanent
A1 A2
A3
Primary
Volatile Temporary
CPU, RAM
I/O, N/W
Terminal
CLI → Command
OS
GUI → Window, Mac
Linux
TV
A3
A2
A1
OS
RAM
Remote


Programming language
User
Applications
OS
Hardware

a = 10

b = 20

c = a + b

print( )

Low

Sign

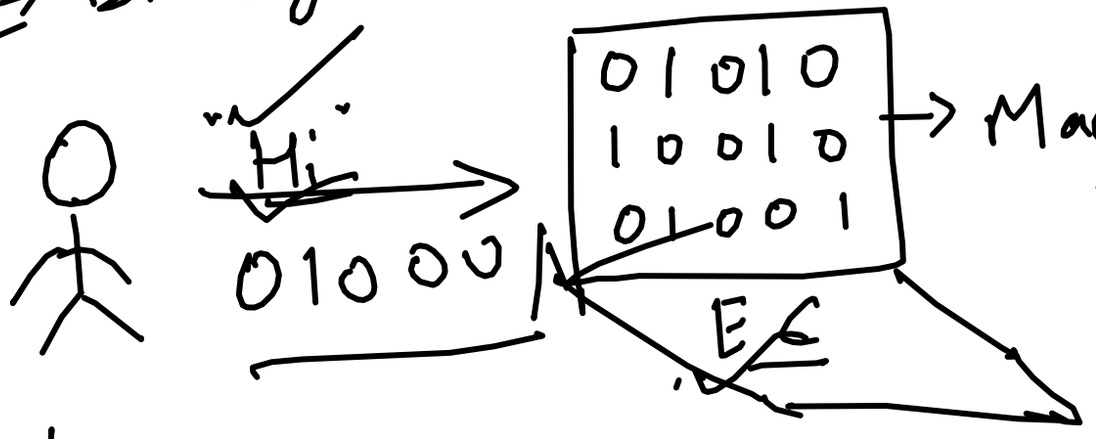
Machine language → Binary language → 0 1

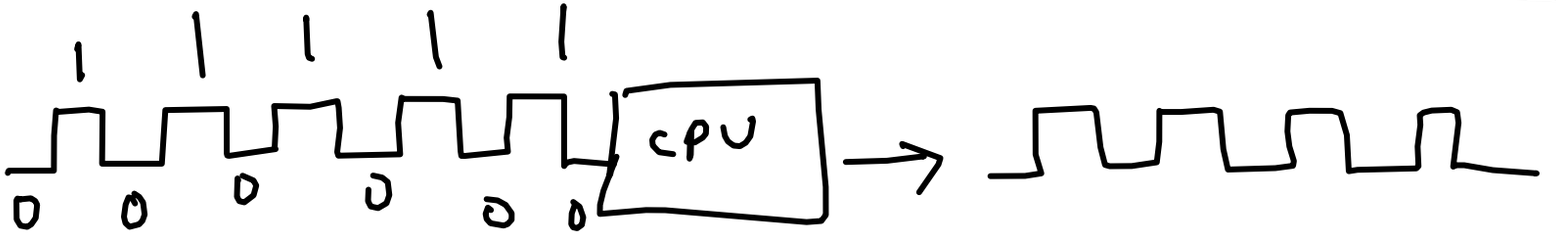
Communication

Exchange of ideas

High

English





a=10

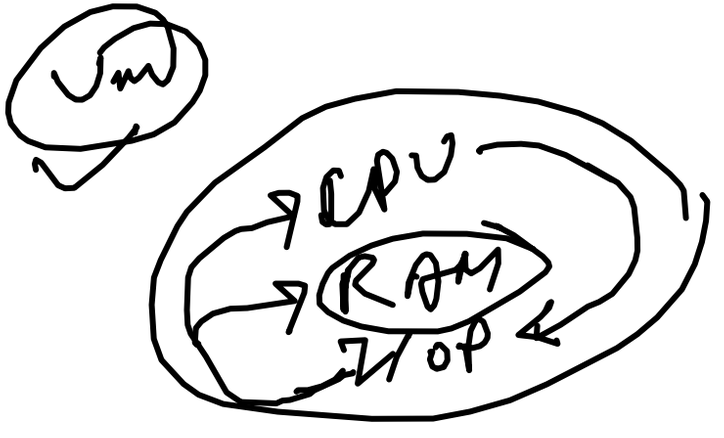
ASCII → a → 97 , = → 61

Binary

Num	D by 2	Q	R
97	2	48	1
48	2	24	0
24	2	12	0
12	2	6	0
1	2	3	0
3	2	1	1
1	2	0	1

61	2	30	1
30	2	15	0
15	2	7	1
7	2	3	1
3	2	1	1
1	2	0	1

11 00 0011 11 1101 1010



L.L.P.L. are not portable

Low Level Source code

↓

Set of H/W components

Assembly language

8085 → 8086

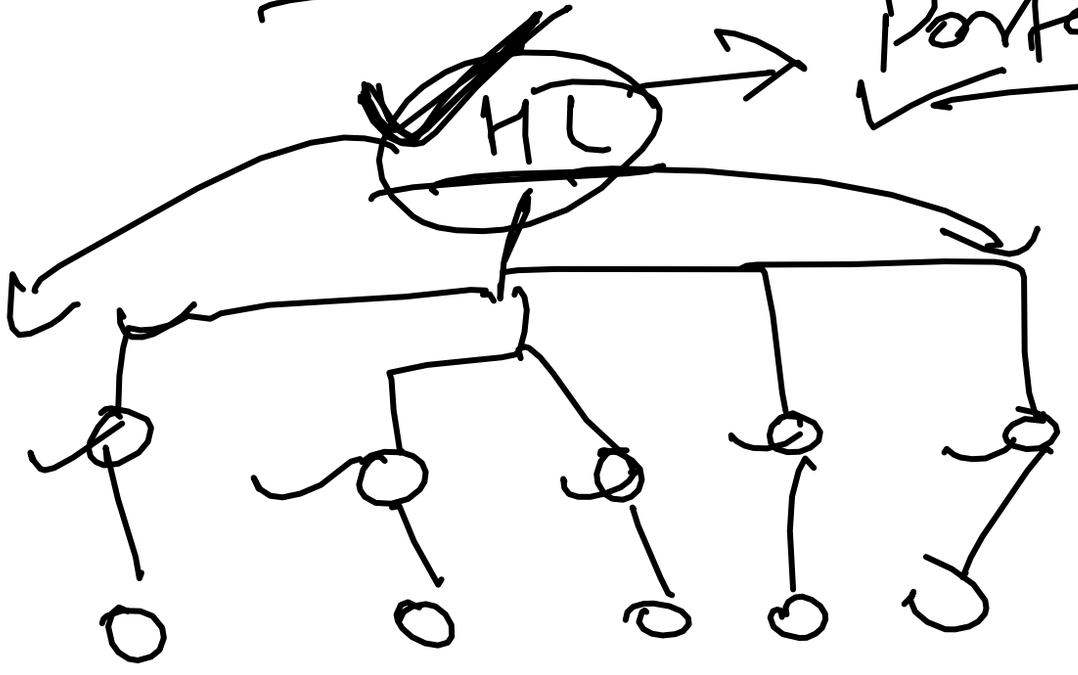
↓ ↑

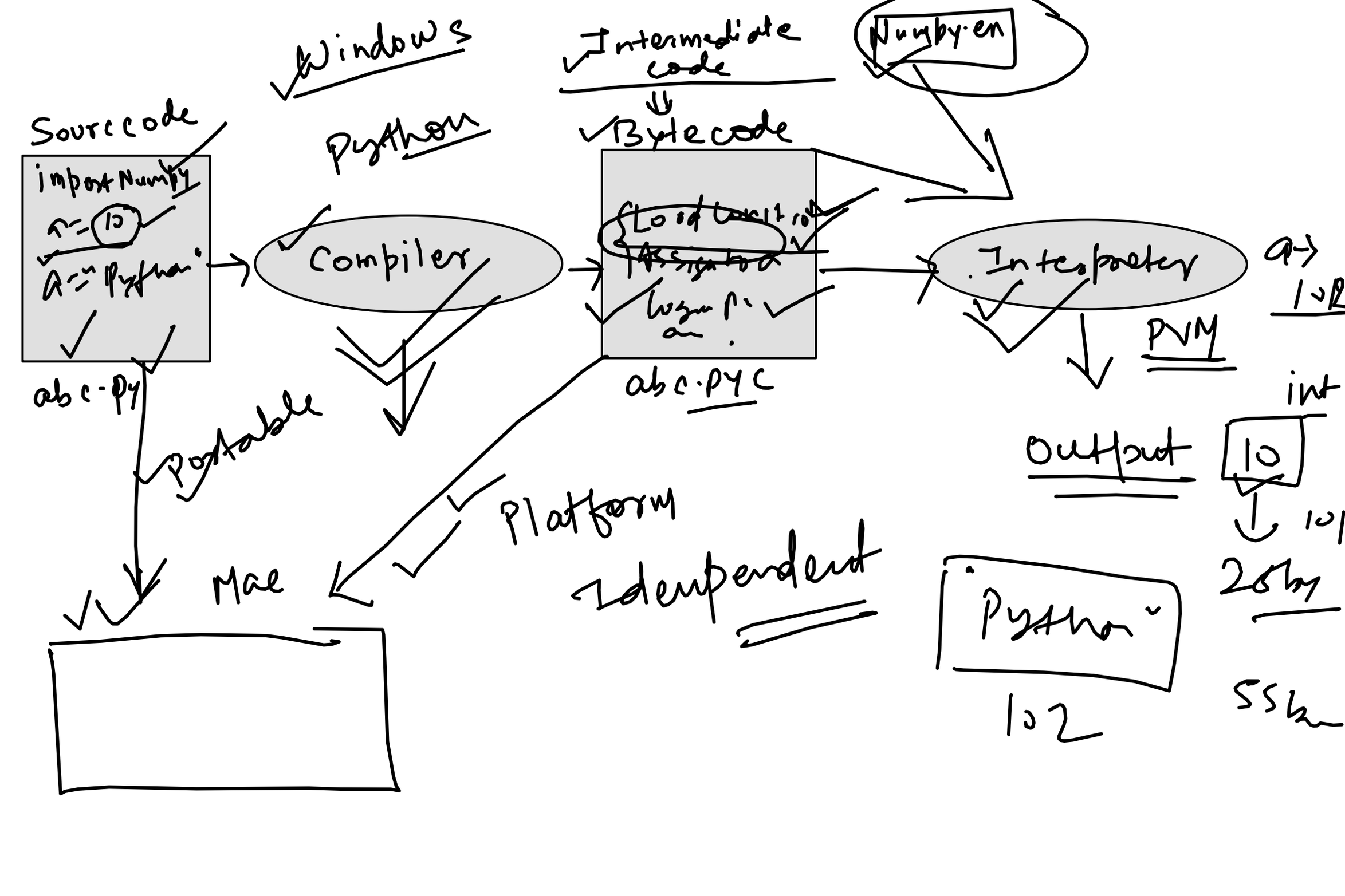
Portable

↳ All H.L.

P.L are

Portable:

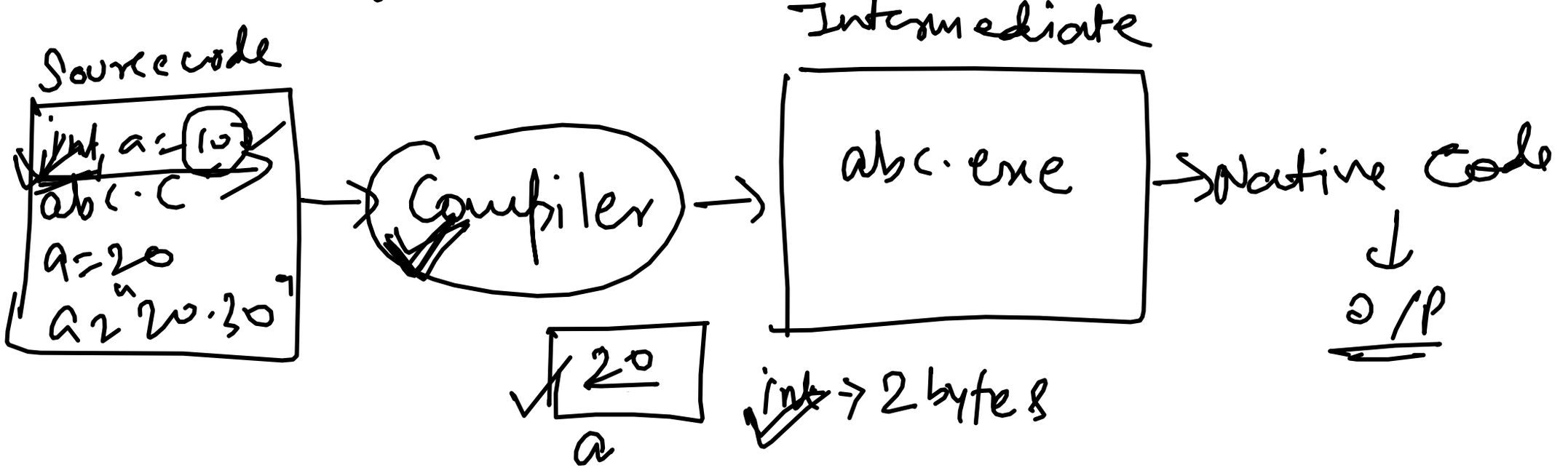


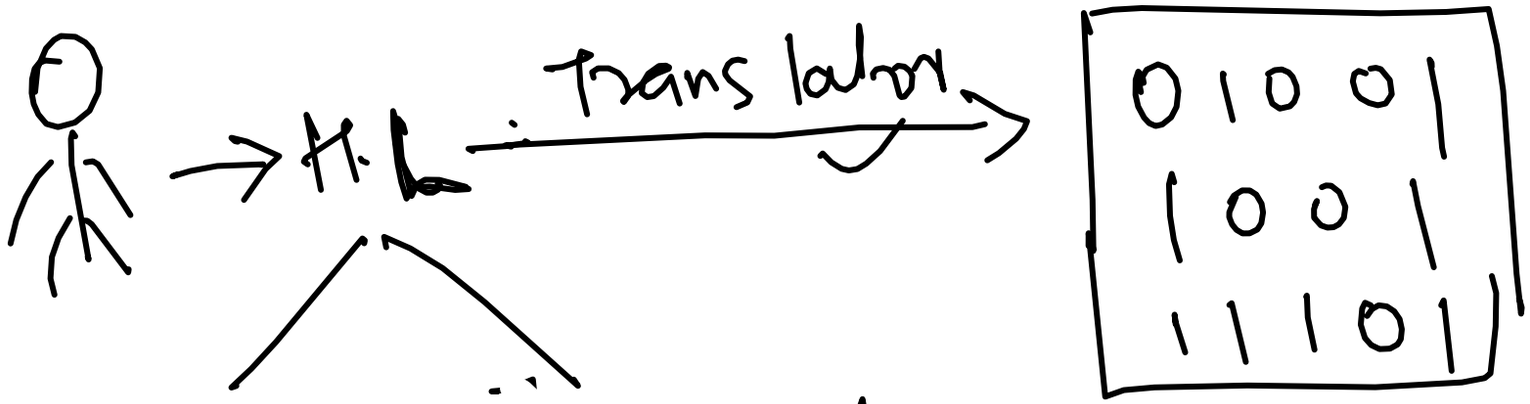

Windows
Intermediate code
Numpy.en
Source code
Python
Bytecode
import Numpy
a = 10
A = Python
Compiler
Interpreter
PVM
abc.py
abc.pyc
Portable
Platform
Mac
Independent
Output
10
Python
102

Statically Typed Programming Language

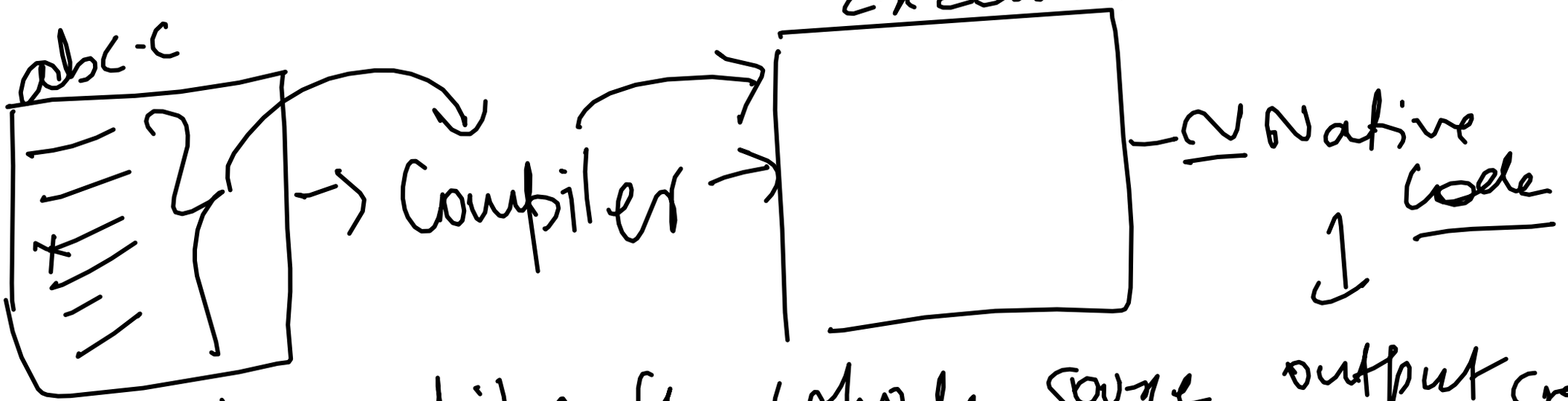
C, C++, Java ↑

How C Programs are executed





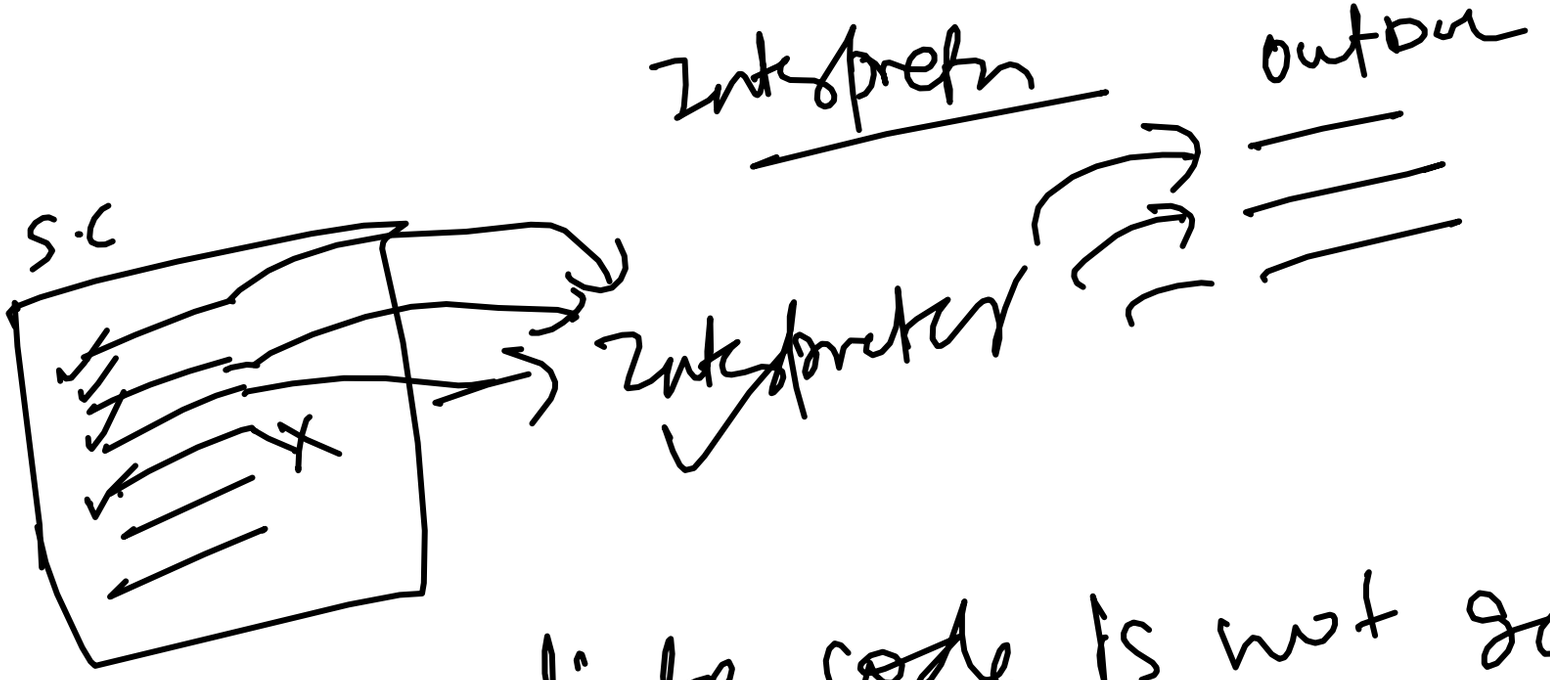
Compiler Interpreter Intermediate code



(1) It compiles the whole source code at once

(2) It generates the intermediate code

(3) If any error in s.c the I.c will not output code



(1) Intermediate code is not generated

(2) It executes the code instruction by instruction

(3) Stops the execution immediately when it finds the error

Compiled vs. Interpreted language

Memory allocation & Typechecking

Compilation Stage

↓

Compiled language

C, C++, Java

Interpretation Stage

↓

Interpreted language

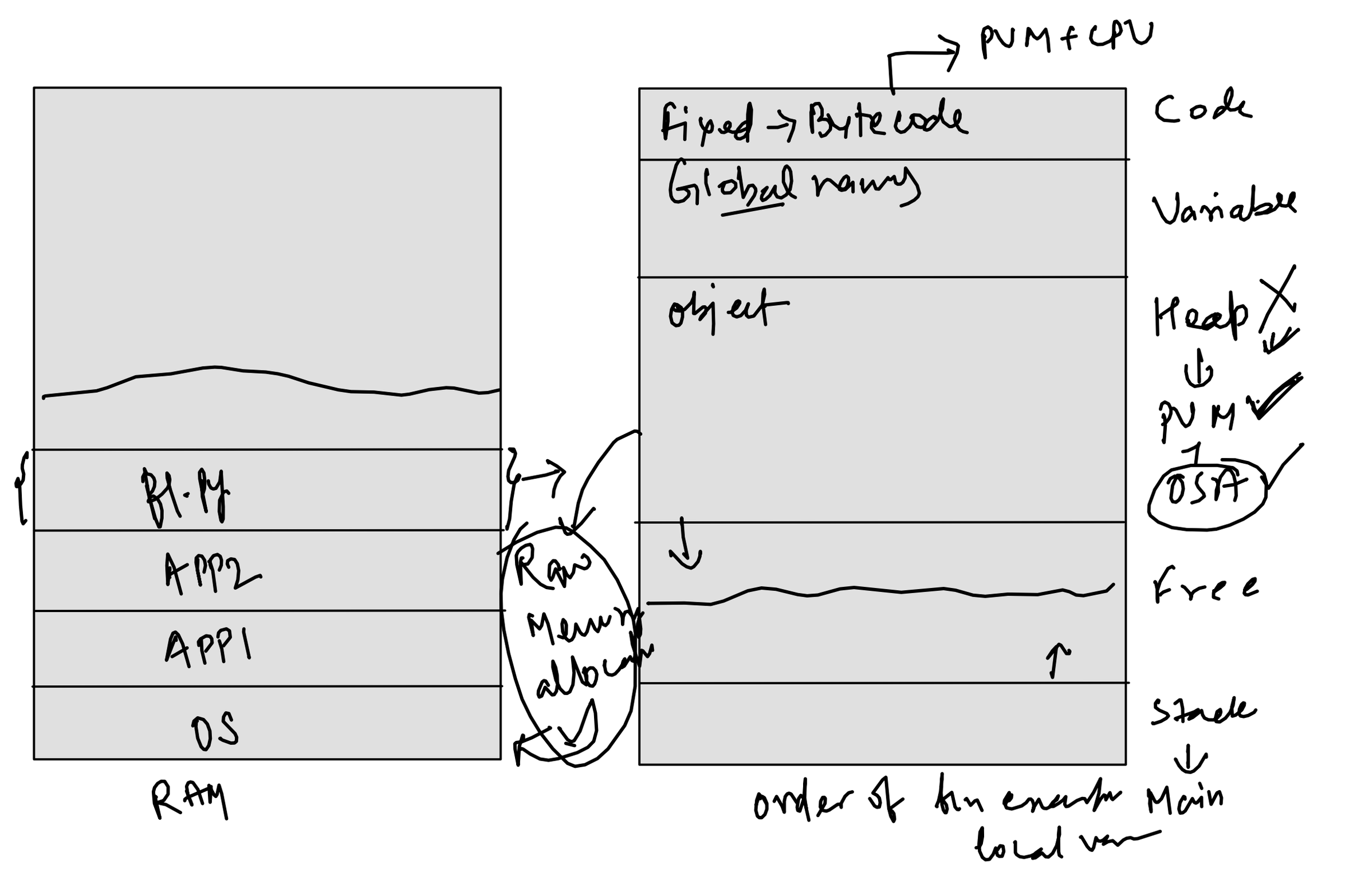
Task of Compiler & Interpreter in Python

Types of Error in Python

① Compile time Error → Compiler
- Syntax
- Indentation

→ S-C

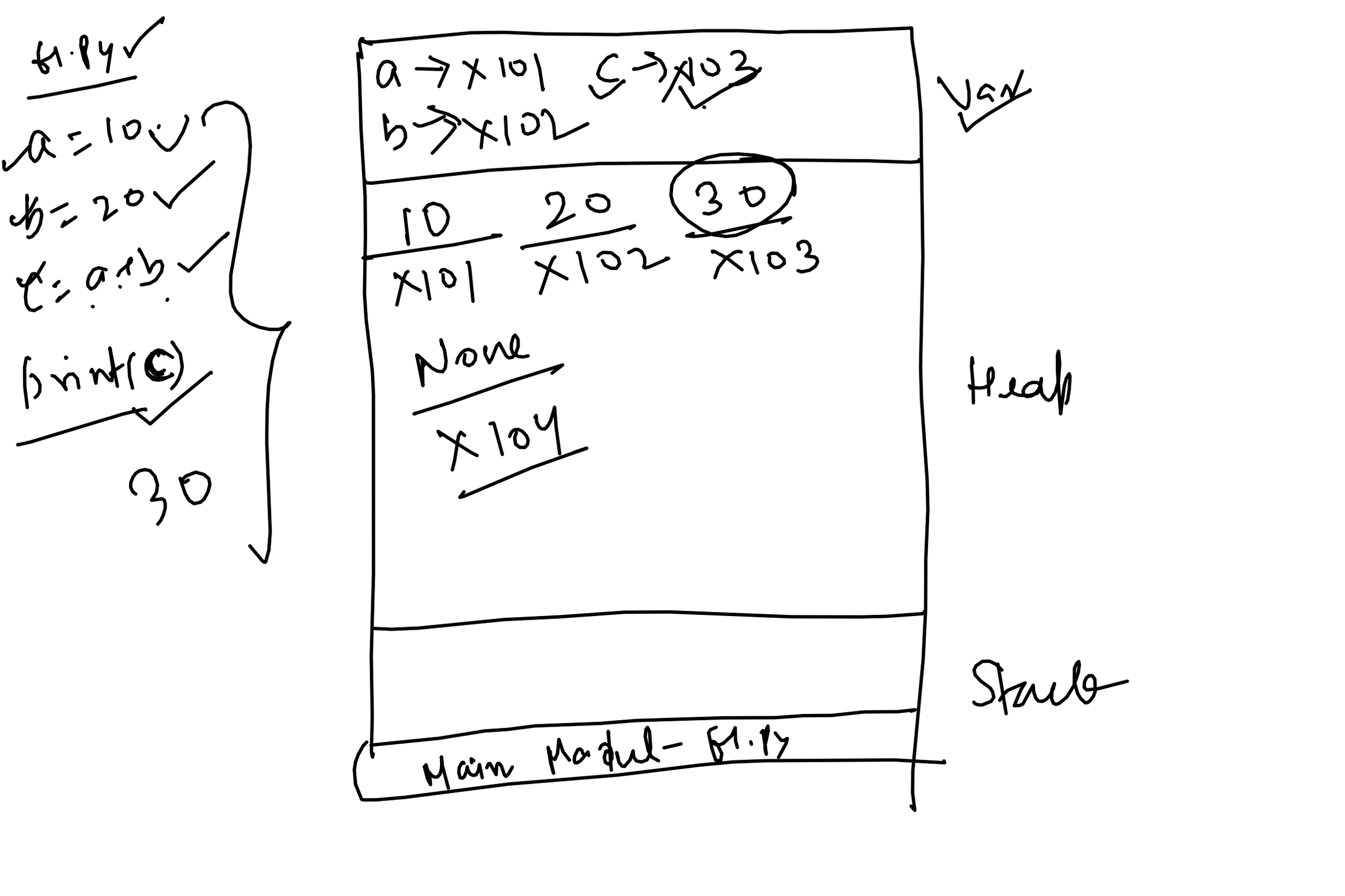
② Runtime Error → Interpreter → B-C
- Name
- Type
- Value etc


PVM + CPU
Fixed → Bytecode
Code
Global many
Variable
object
Heap
PVM
OSA
Free
Stack
Main
order of fun execute
local var
RAM
BI.My
APP2
APP1
OS
Raw
Memory
allocate

Objects

1. Integer
2. Set
3. Complex
4. Floats
5. String
6. Boolean
7. None
8. Tuple
9. List
10. Bytes
11. Dictionary
12. Userdefined objects

etc


fi.py
a = 10
b = 20
c = a+b
print(c)
30
a → x101 c → x103
b → x102
Var
10 20 30
x101 x102 x103
None
x104
Heap
Stack
Main Module - fi.py