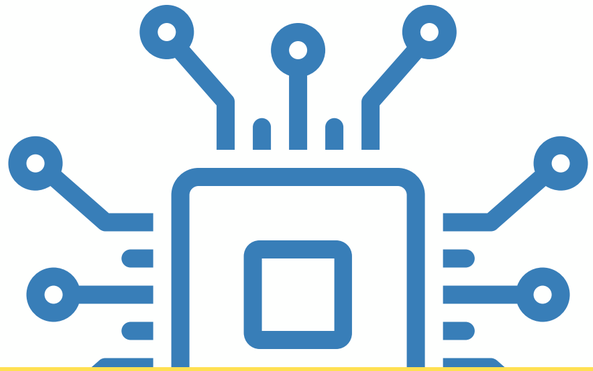
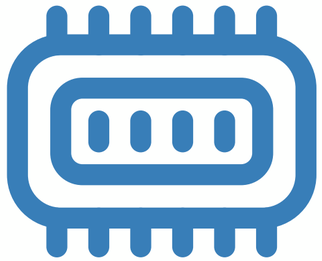
IS IT ME OR PYTHON MEMORY MANAGEMENT?

YULIIA BARABASH
LAYSA UCHOA

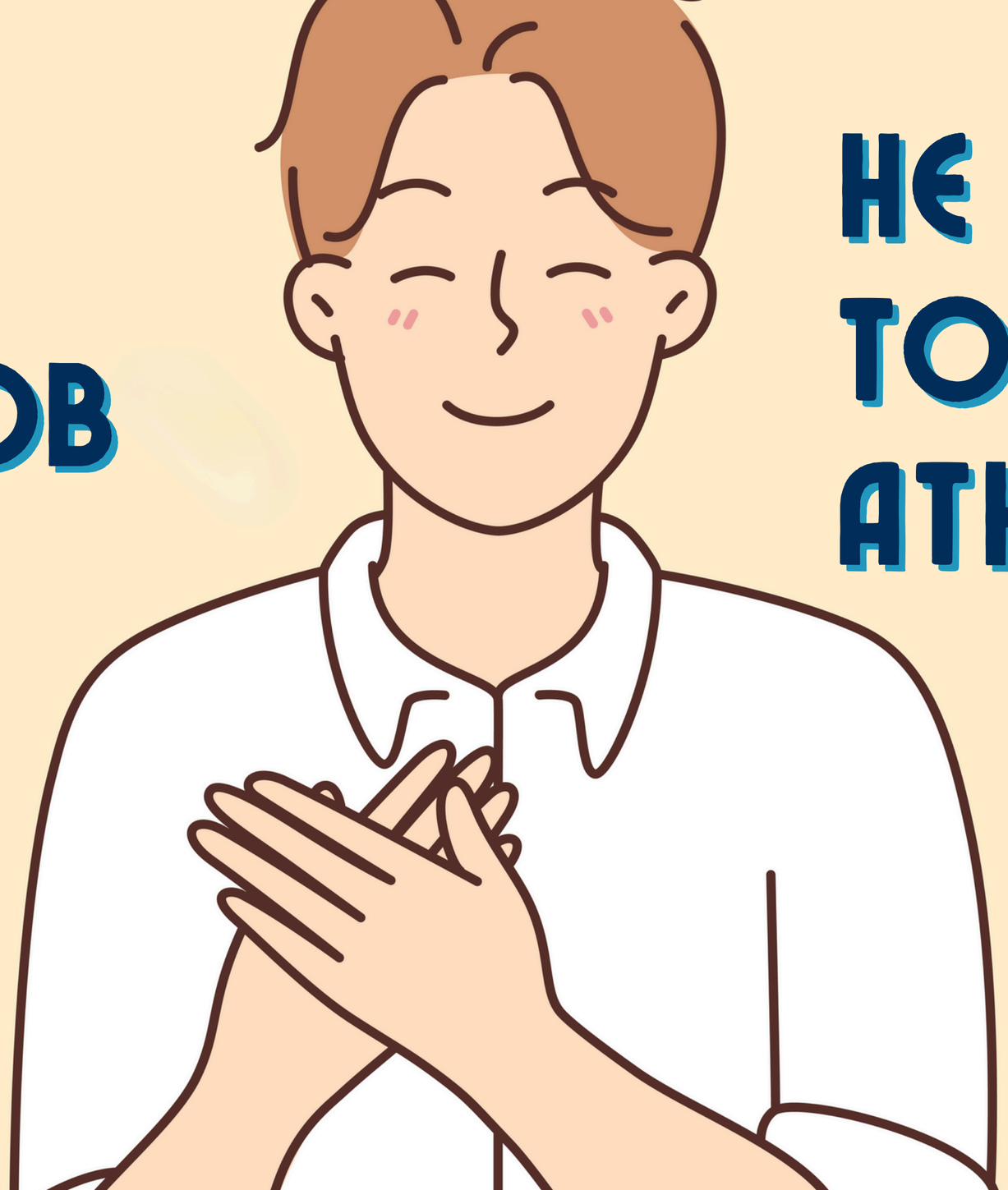
CLOUD ENGINEER AT NORDCLOUD



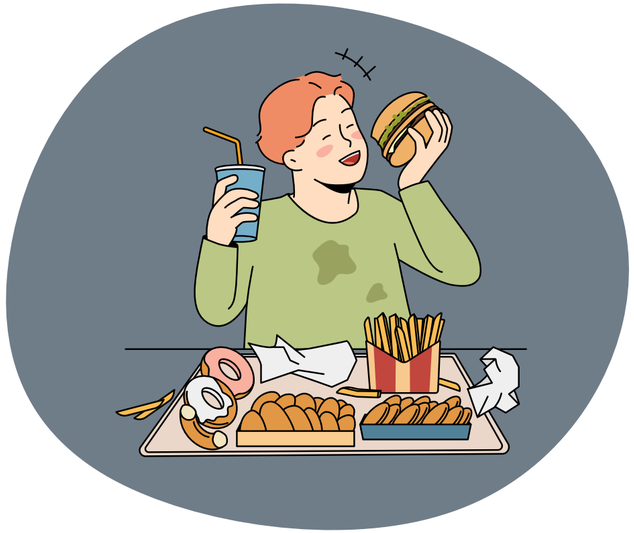
LET'S TALK ABOUT PERFORMANCE



IT IS BOB



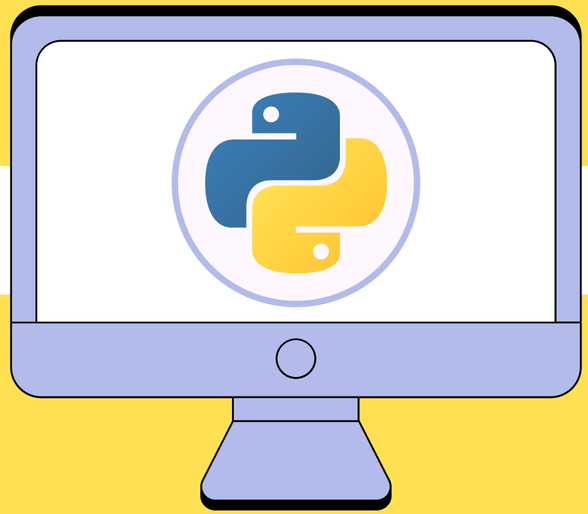
HE WANTS TO BE AN ATHLEAT



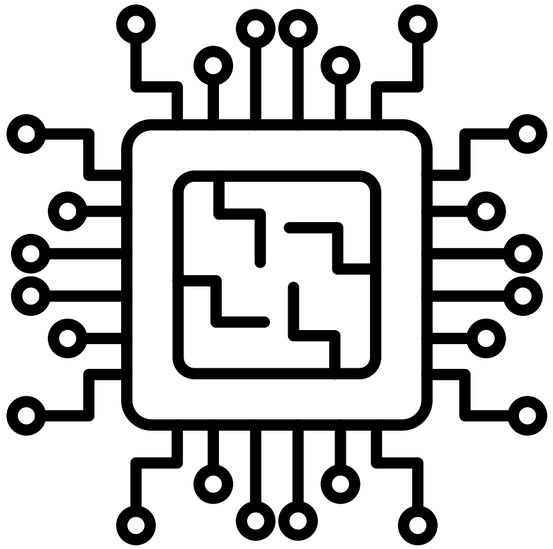
THERE IS NO SOFTWARE

THERE IS NO SOFTWARE

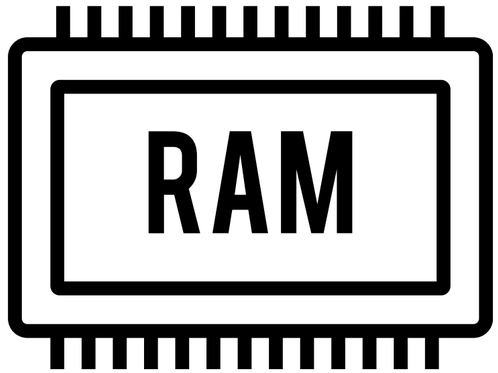
WITHOUT HARDWARE



COMPONENTS OF HARDWARE

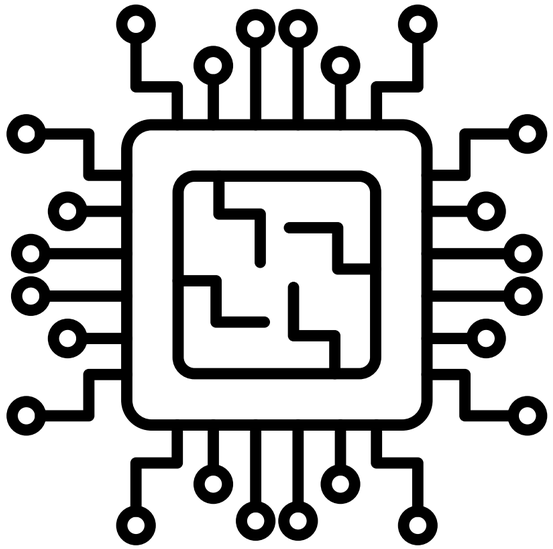


COMPUTING UNITS

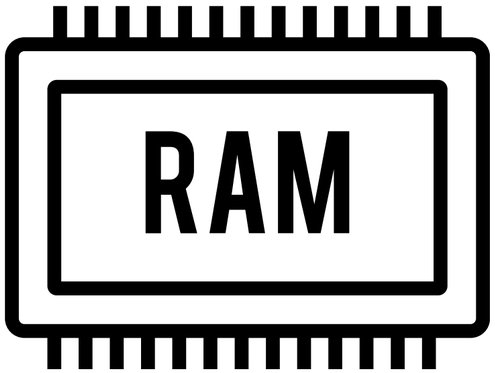


MEMORY UNITS

COMPONENTS OF HARDWARE

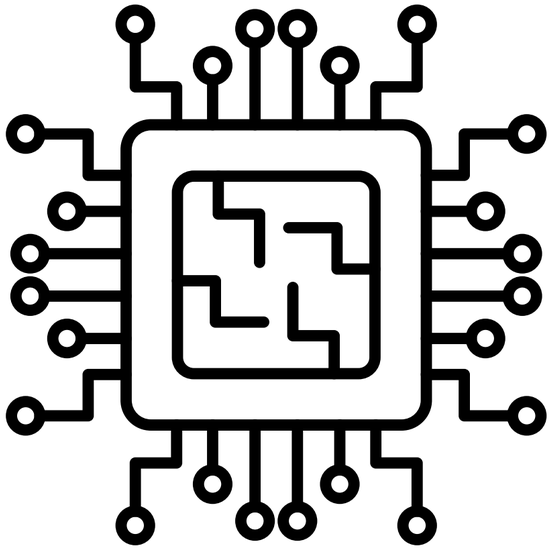


COMPUTING UNITS



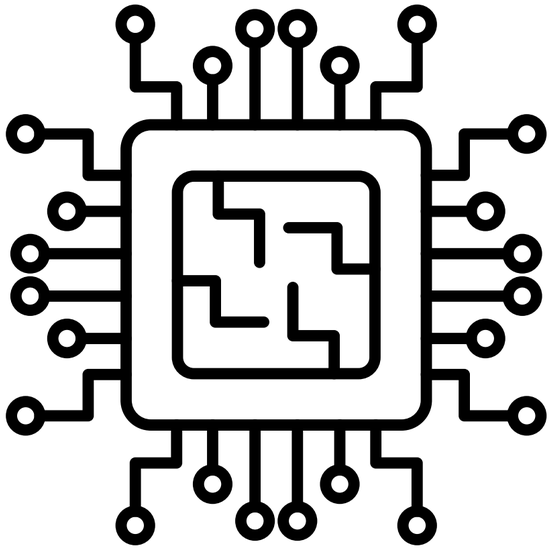
MEMORY UNITS

COMPONENTS OF HARDWARE



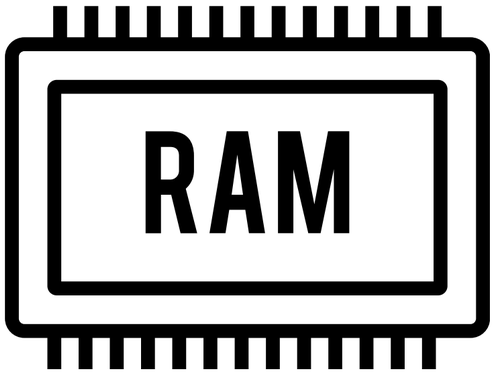
COMPUTING UNITS

COMPUTING UNITS



- Responsible for execution of operations
- Influence by:
  - n. of machine code instructions
  - execution speed of instructions
- Types:
  - CPU
  - GPU

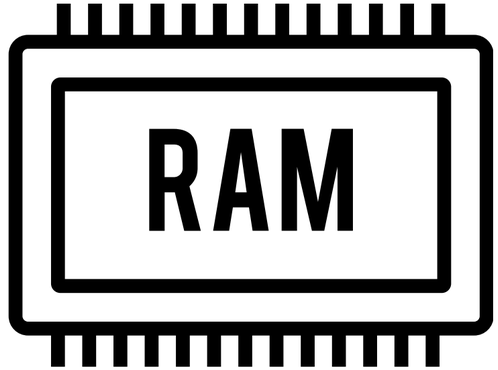
COMPONENTS OF HARDWARE



MEMORY UNITS

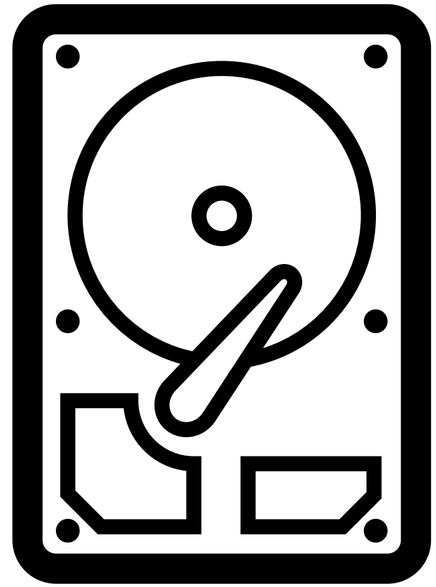
MEMORY UNITS

- Storing data and information
- Influence by:
  - Latency/architecture of memory units
- Types:
  - Hard Disk Drive(HDD)
  - Solid State Drive(SSD)
  - RAM
  - Cache L1/L2/L3

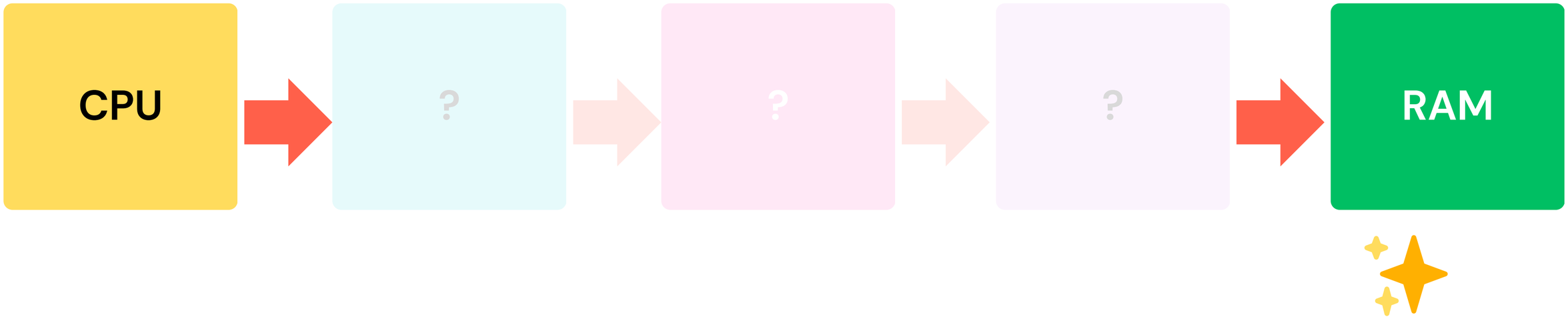


HARD DISK DRIVE(HDD)/SOLID-STATE DRIVE(SSD)

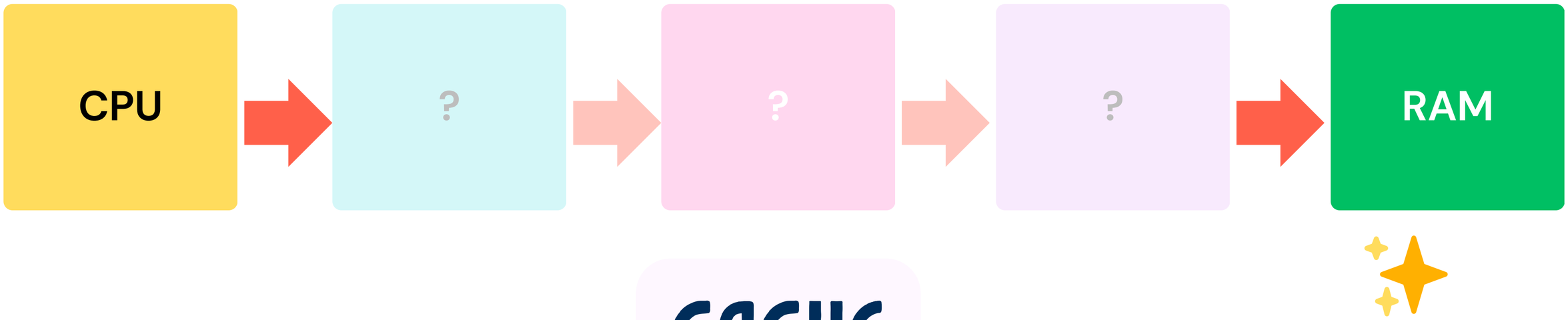
- Long-term storage
- Responsible for store the files and data needed for execution
- Lower speed write/read operations compare to RAM
- Reading and Writing Files



COMPONENTS OF HARDWARE

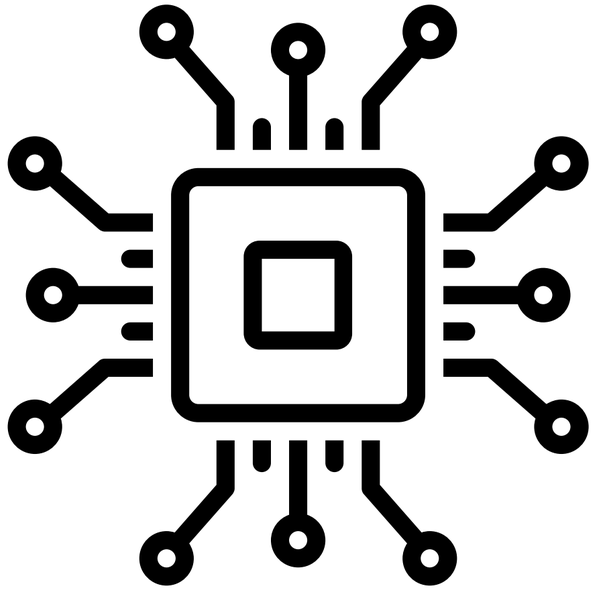


COMPONENTS OF HARDWARE



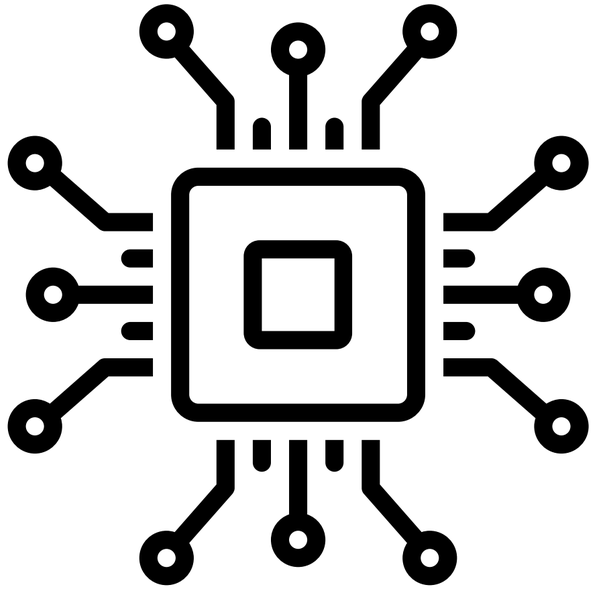
CACHE L1/L2/L3

- Store most frequently used data by CPU
- Fastest and smallest type of memory ✨
- Store most often executed instructions
- Cache Hit / Miss 🎯

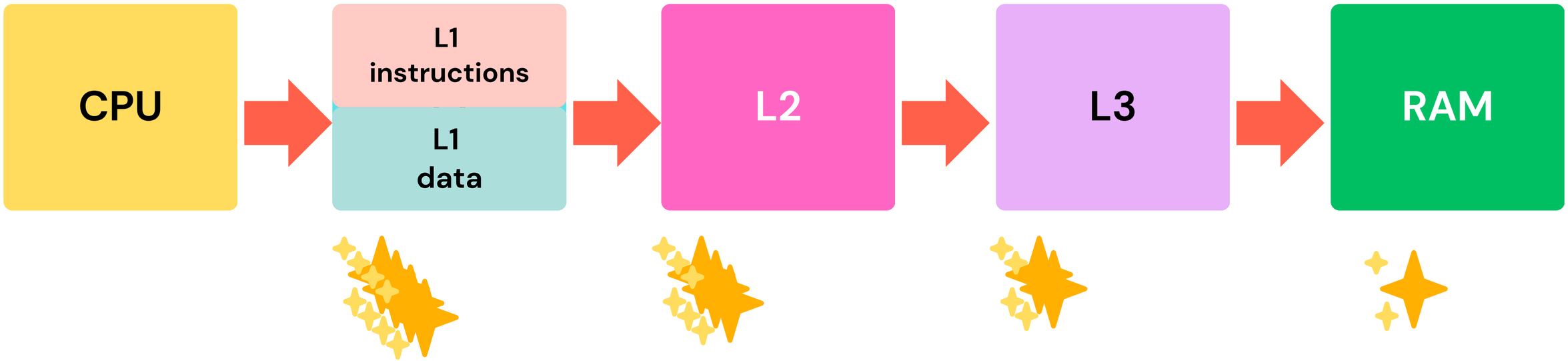


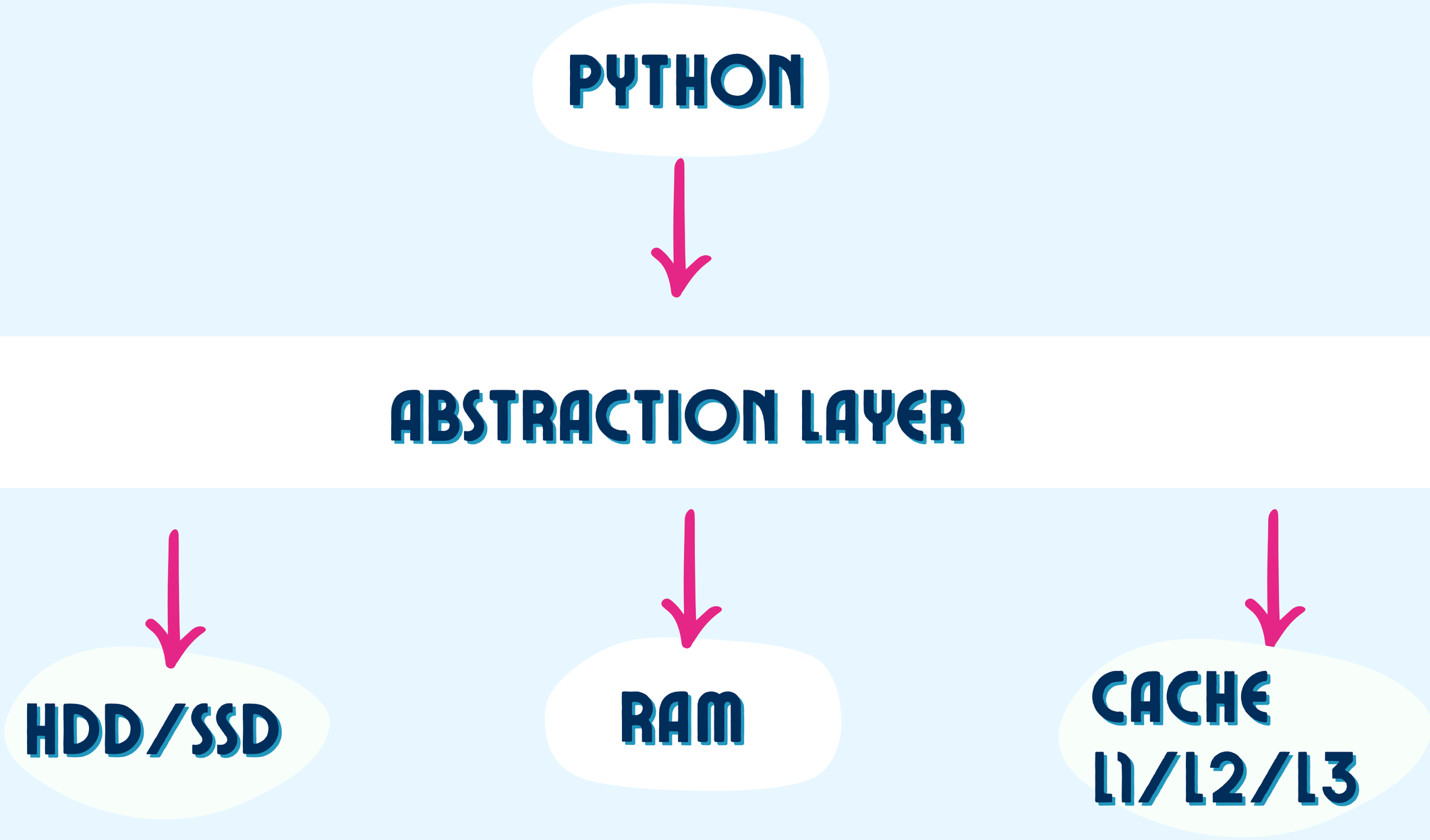
RANDOM ACCESS MEMORY (RAM)

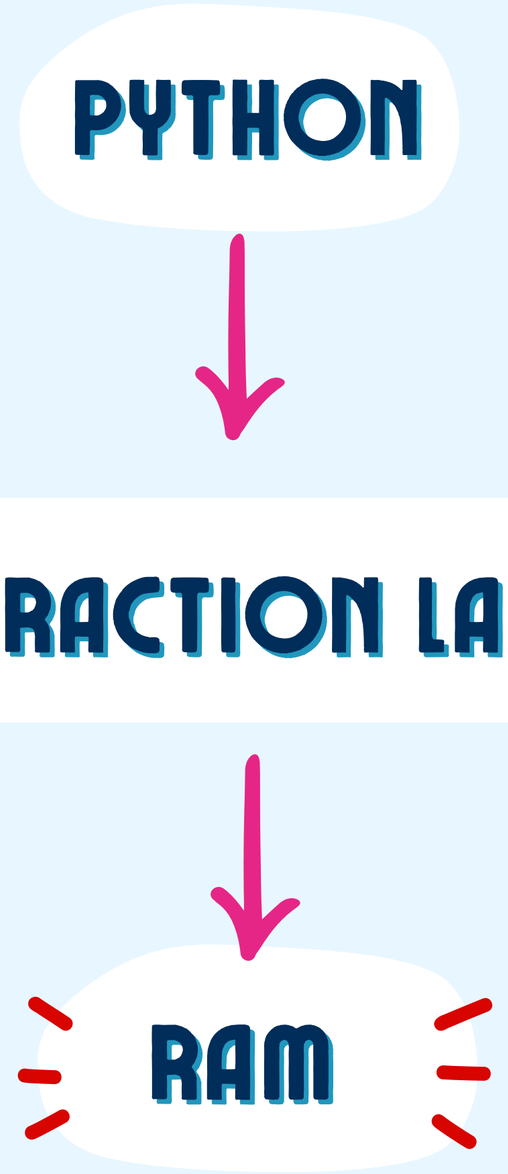
- Short-term memory
- Stores data and objects that are currently being used by program
- It stores Python private heap space



COMPONENTS OF HARDWARE

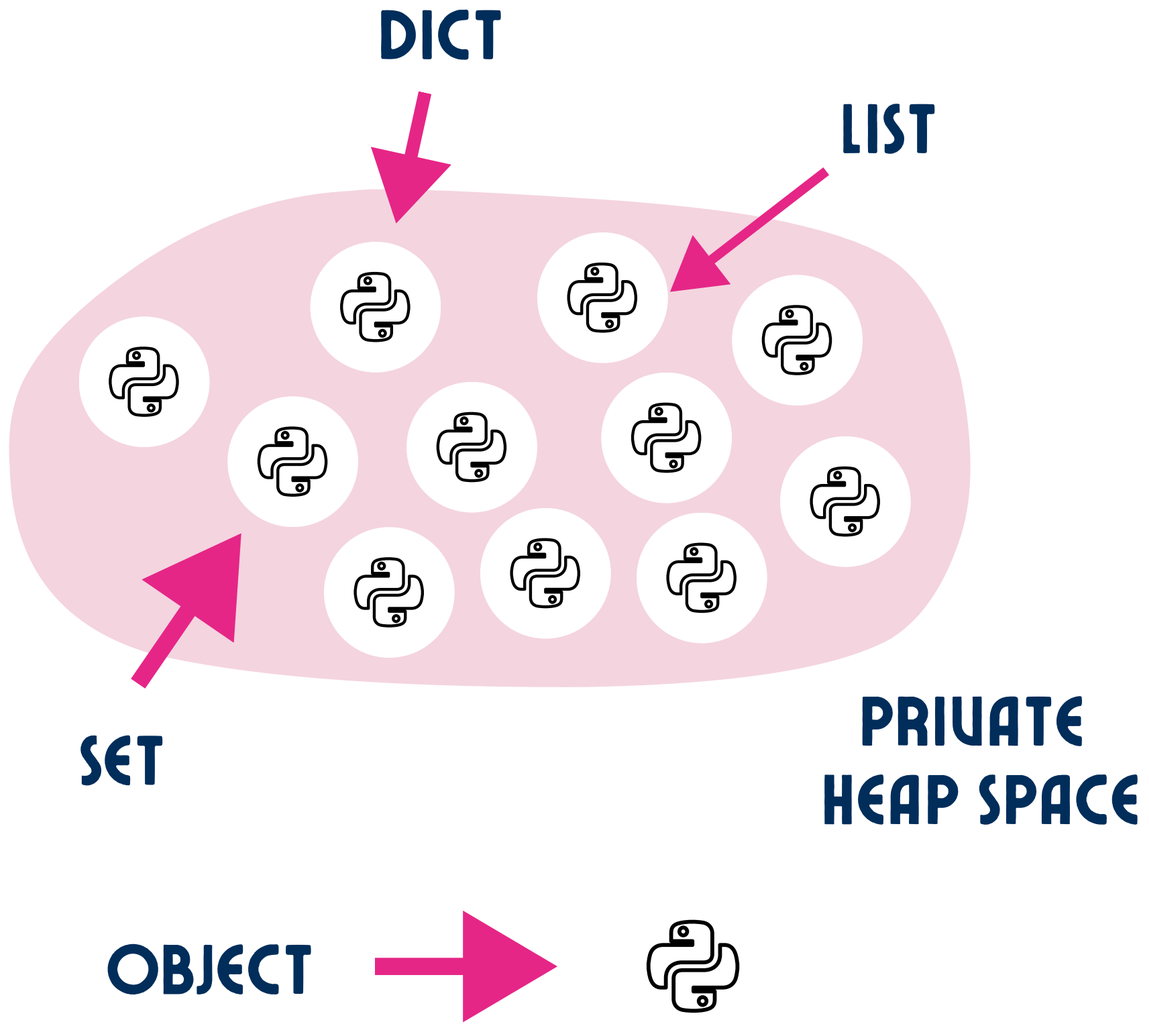


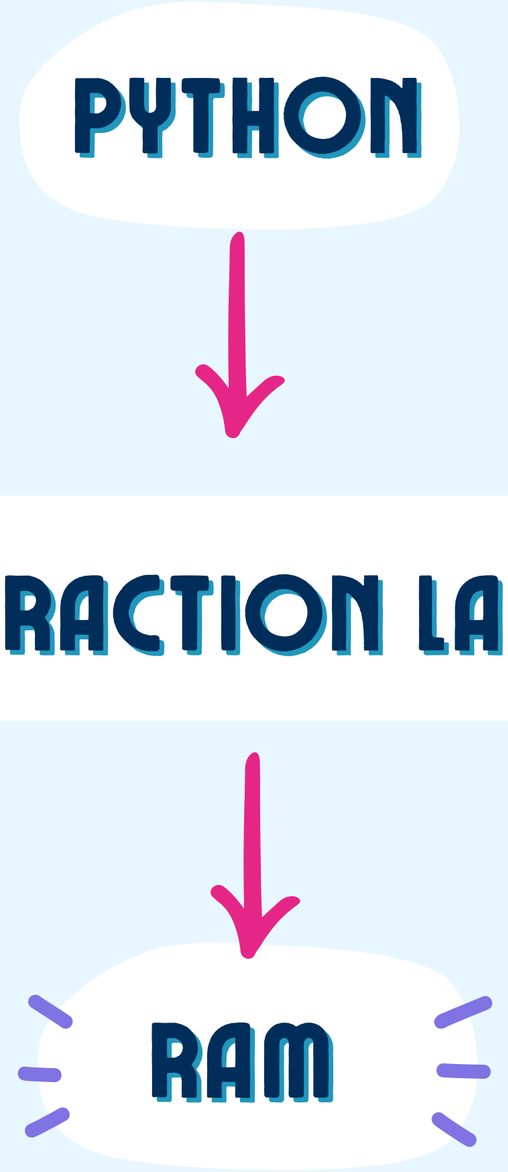

PYTHON
ABSTRACTION LAYER
HDD/SSD
RAM
CACHE
L1/L2/L3

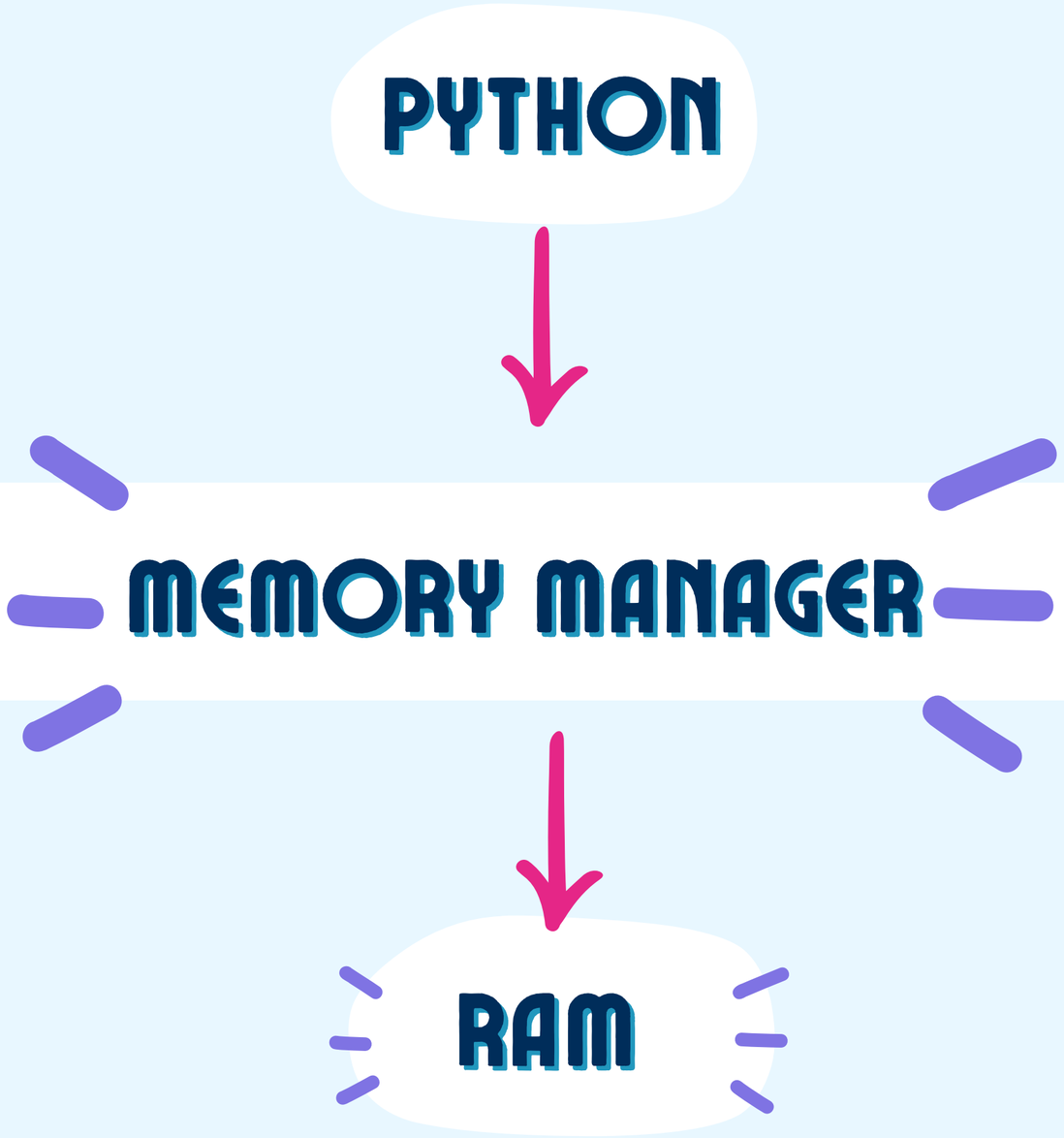

PYTHON
ABSTRACTION LAYER
RAM

PYTHON PRIVATE HEAP SPACE

- Part of memory that is managed by Python
- Everything is object in Python
- Python's memory manager and garbage collector

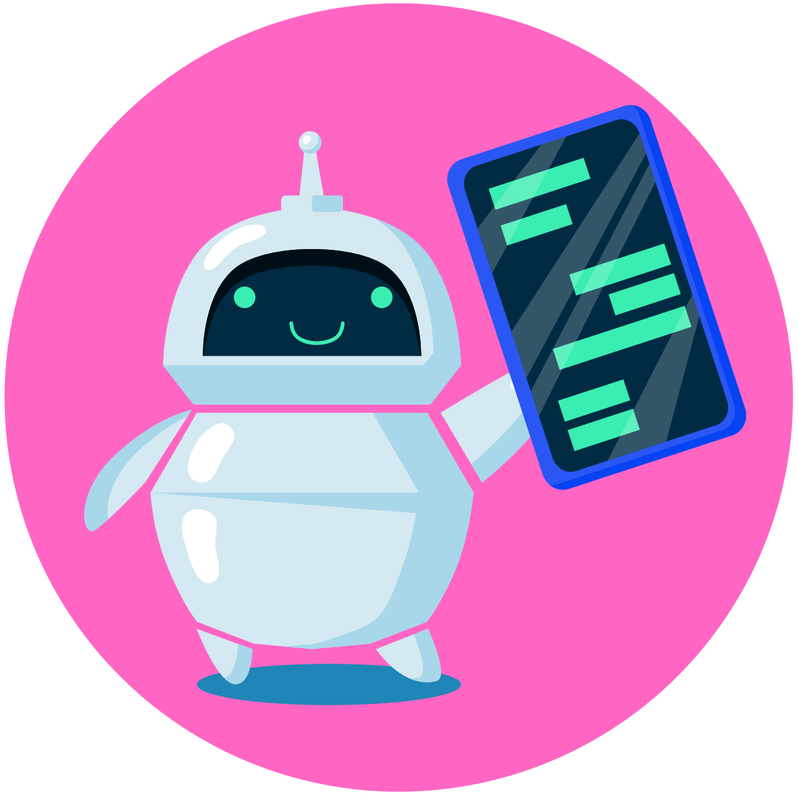



PYTHON
ABSTRACTION LAYER
RAM


PYTHON
MEMORY MANAGER
RAM

MEMORY MANAGER

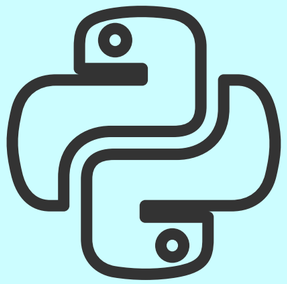
- Manages memory inside private heap space
- Allocate and free space depends on state of application
- Garbage collector works alongside the memory manager



RECAP

- Performance depends on execution environment
- Main components:
  - computing units
  - memory units
  - communication between them
- Types of memory
  - Hard Disk Drive (HDD)/Solid-State Drive (SSD)
  - Random Access Memory (RAM)
  - Cache L1/L2/L3
- All objects are stored in private heap space
- Private heap space is managed by memory manager

BUT HOW MEMORY MANAGER KNOWS WHAT TO DO



INSIDE PYTHON

MEMORY ALLOCATION STRATEGY

MEMORY ALLOCATION STRATEGY

Reference counting

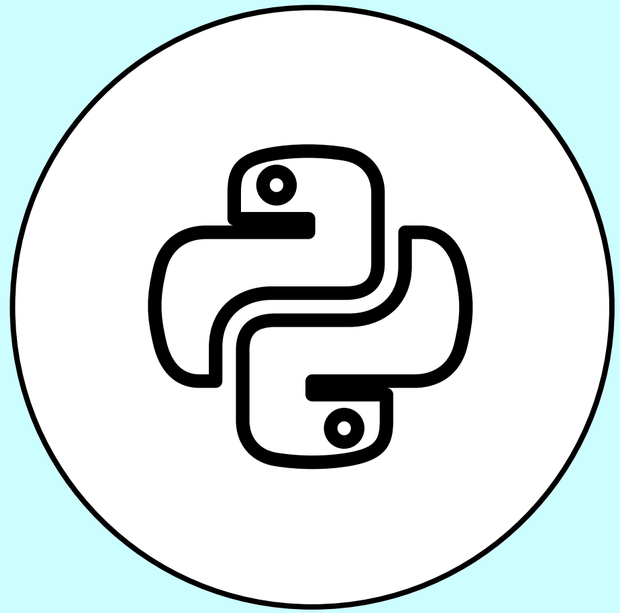
MEMORY ALLOCATION STRATEGY

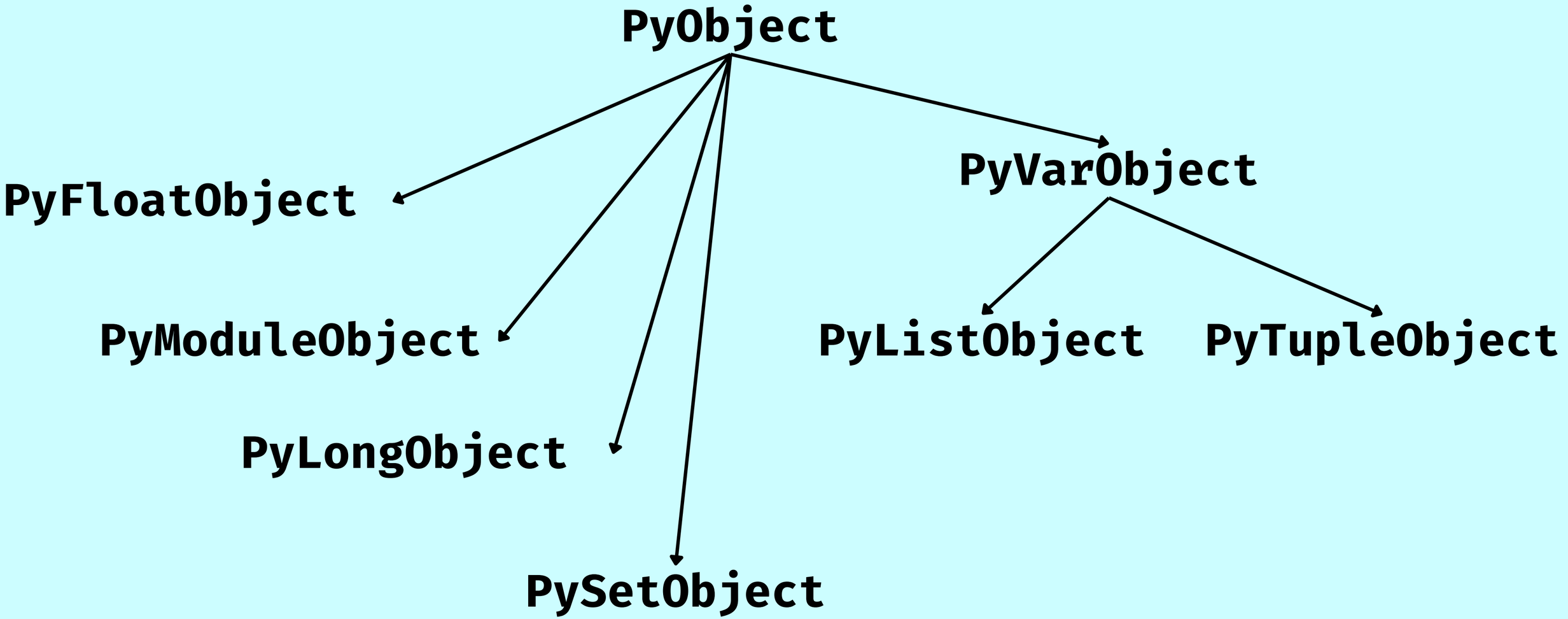
Reference counting

Generational garbage collection

EVERYTHING IS OBJECT

- Representations for:
  - code
  - function
  - none
  - integers
  - containers data type
  - so on




PyObject
PyFloatObject
PyModuleObject
PyLongObject
PySetObject
PyVarObject
PyListObject
PyTupleObject

FLOAT OBJECT

```
A = 3.5
```

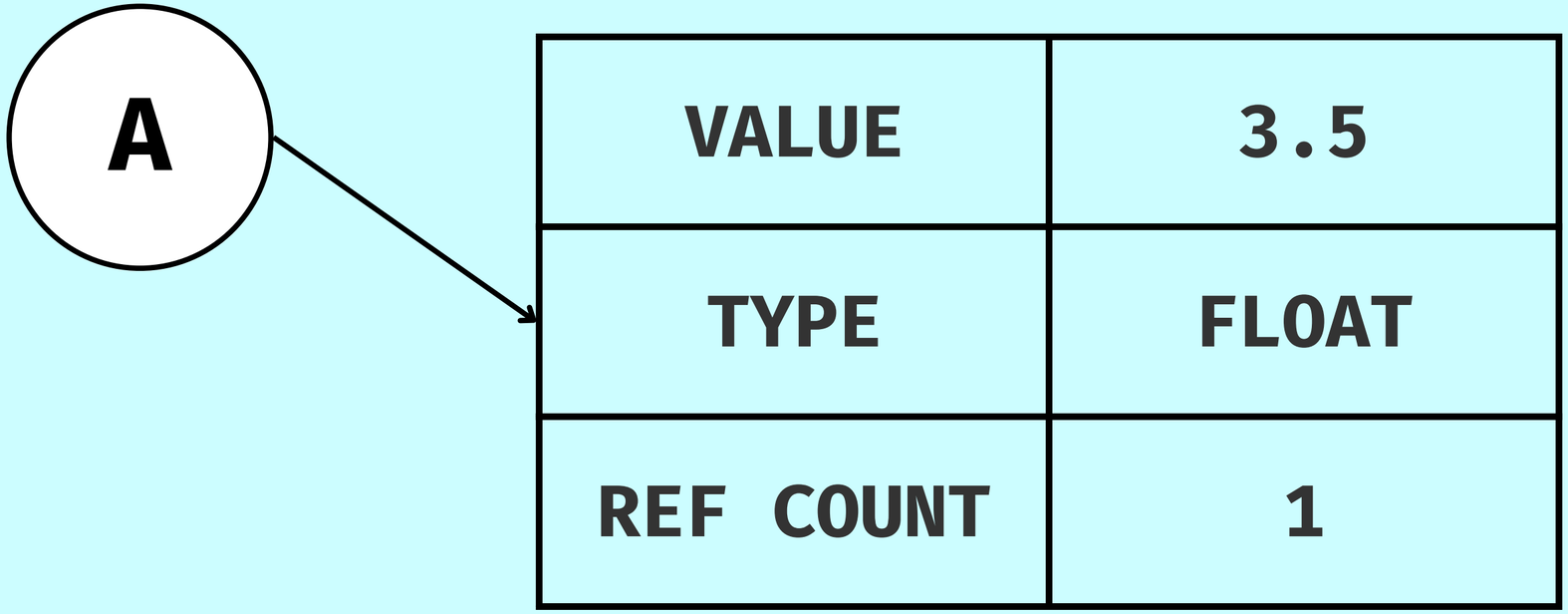
- Floats are represented as PyFloatObject
- Type and Ref Count are inherited from PyObject

VALUE	3.5
TYPE	FLOAT
REF COUNT	1

VARIABLES IN PYTHON

- References to the actual object in memory
- Think of them as names or labels that point to the actual object in memory

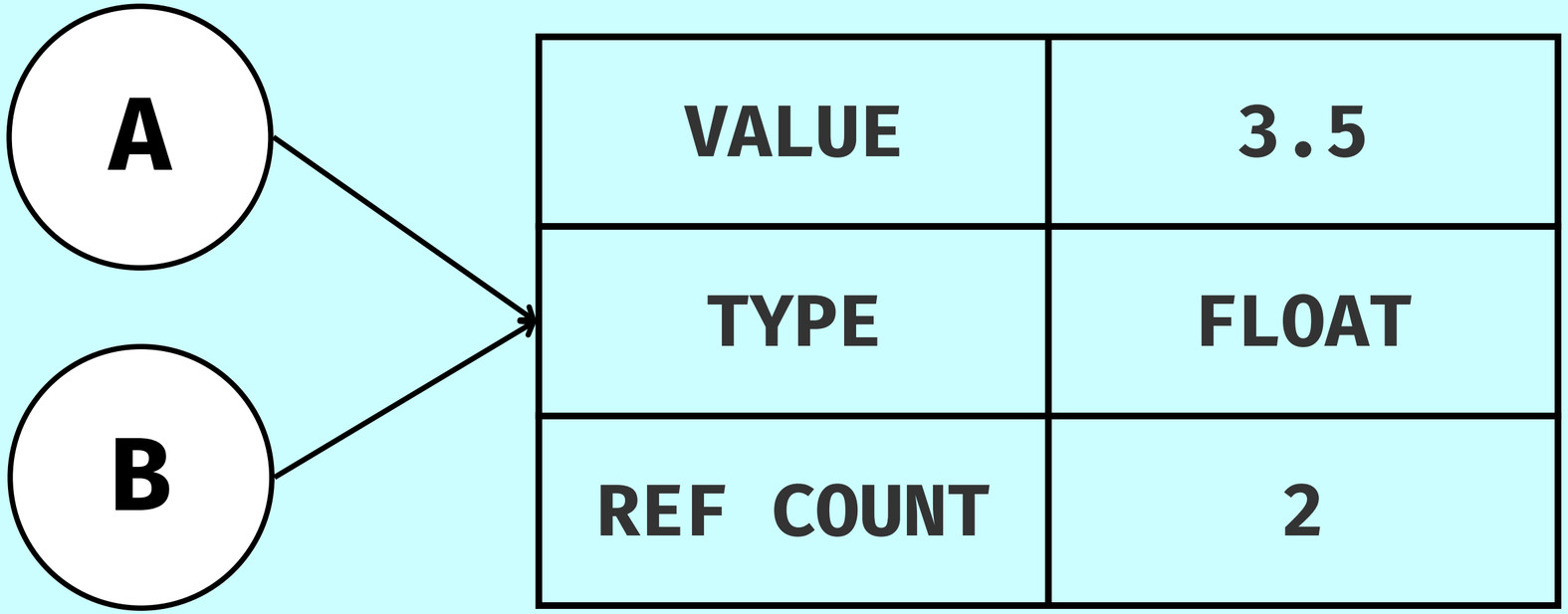
```
A = 3.5
```



VARIABLES IN PYTHON

- References to the actual object in memory
- Think of them as names or labels that point to the actual object in memory

```
B = 3.5
```



EXAMPLE

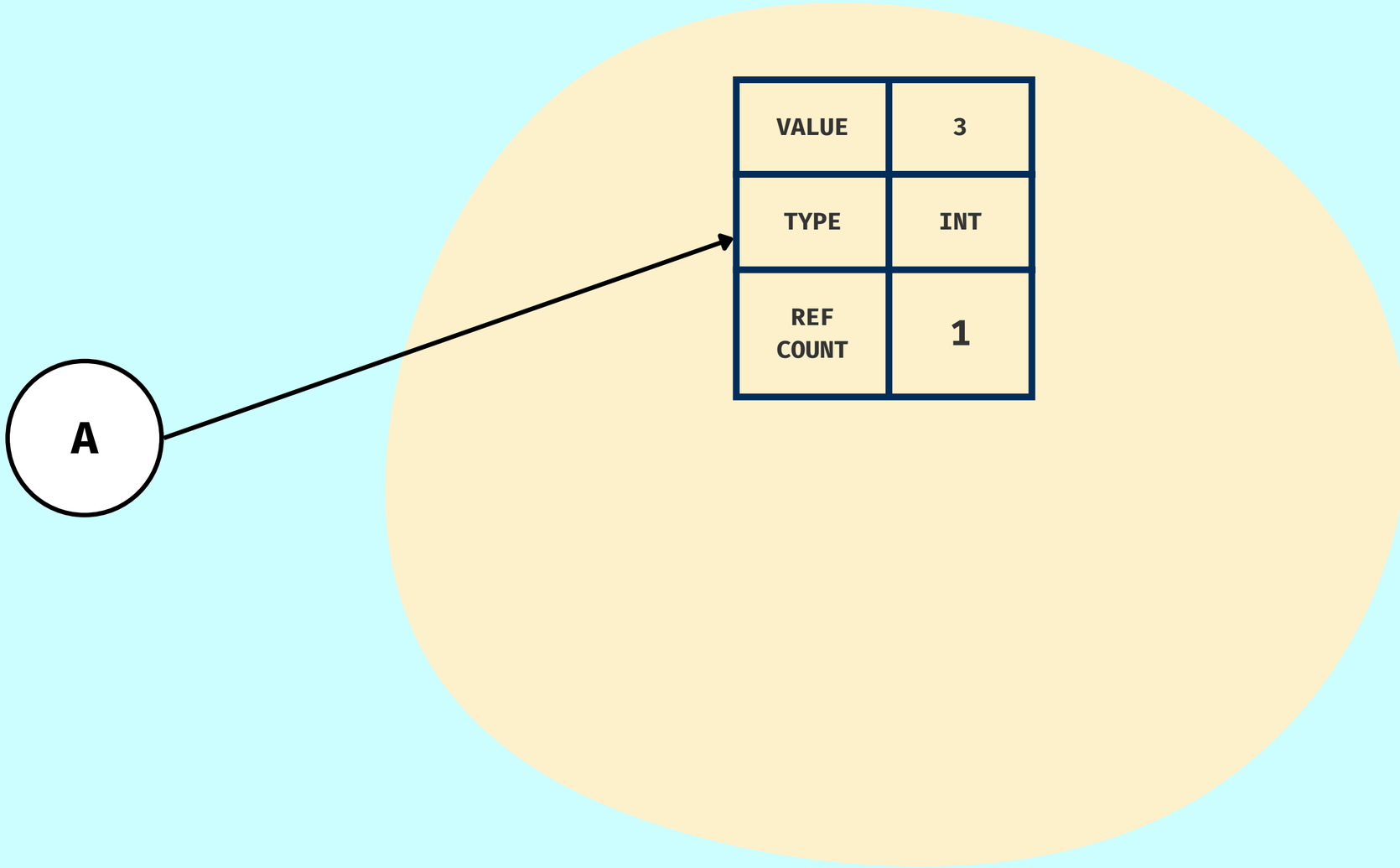
a = 3

a = 6

EXAMPLE

a = 3

a = 6

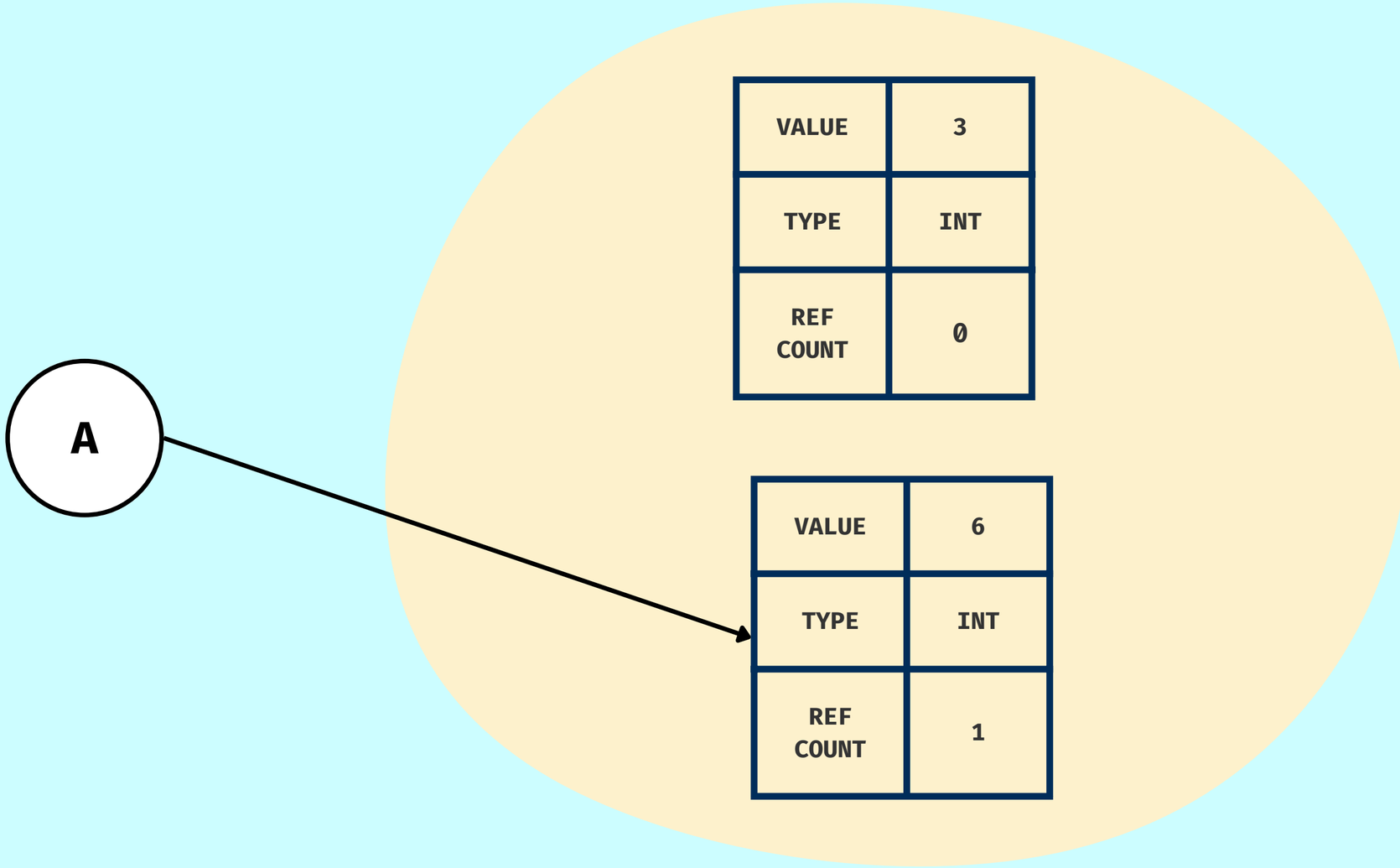


private heap

EXAMPLE

a = 3

a = 6



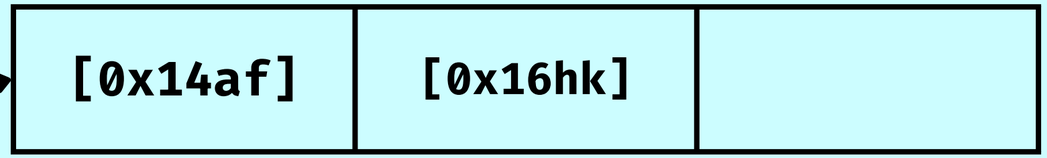
private heap

LIST OBJECT

- List are represented as PyListObject

```
B = [1,2]
```

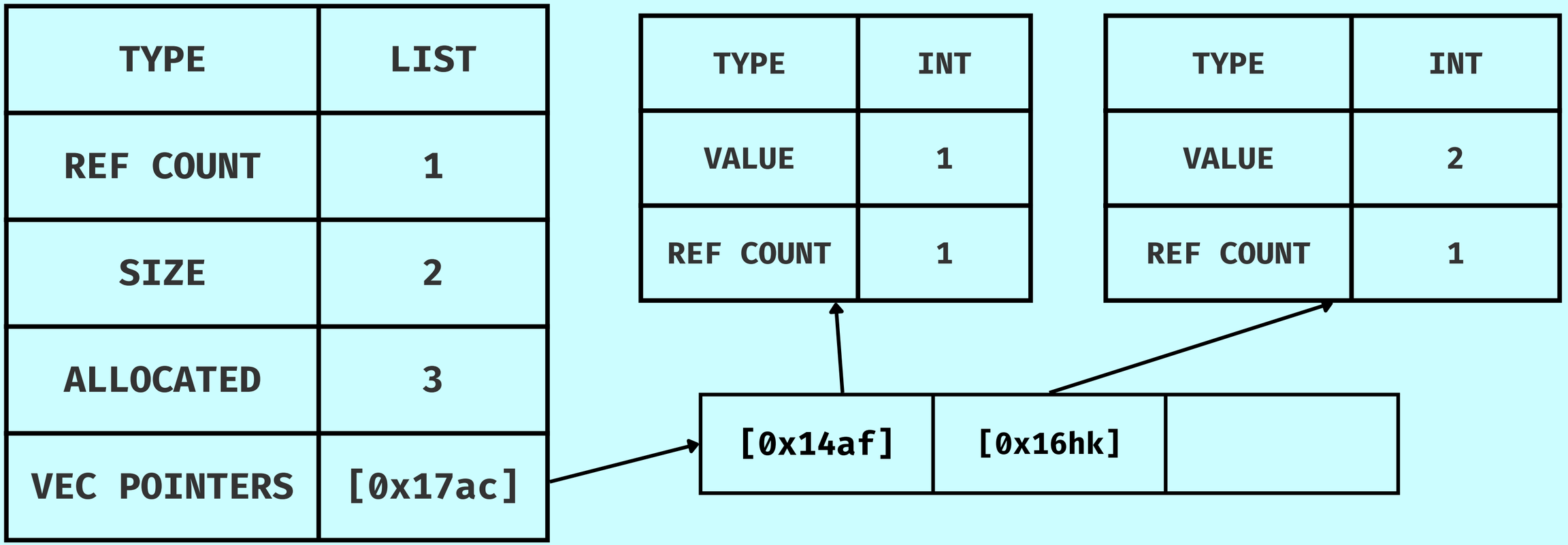
TYPE	LIST
REF COUNT	1
SIZE	2
ALLOCATED	3
VEC POINTERS	[0x17ac]



LIST OBJECT

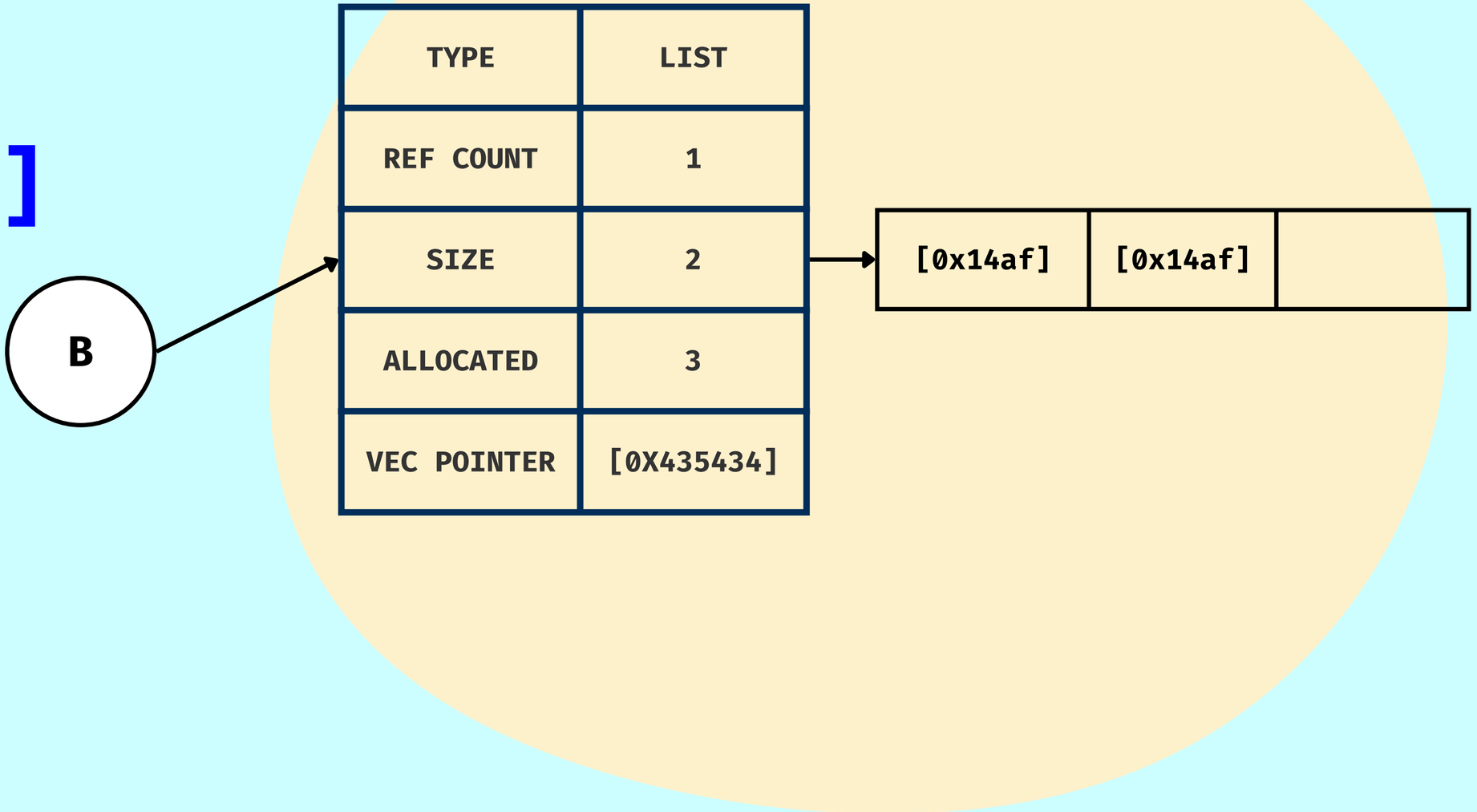
- List are represented as PyListObject

```
B = [1,2]
```



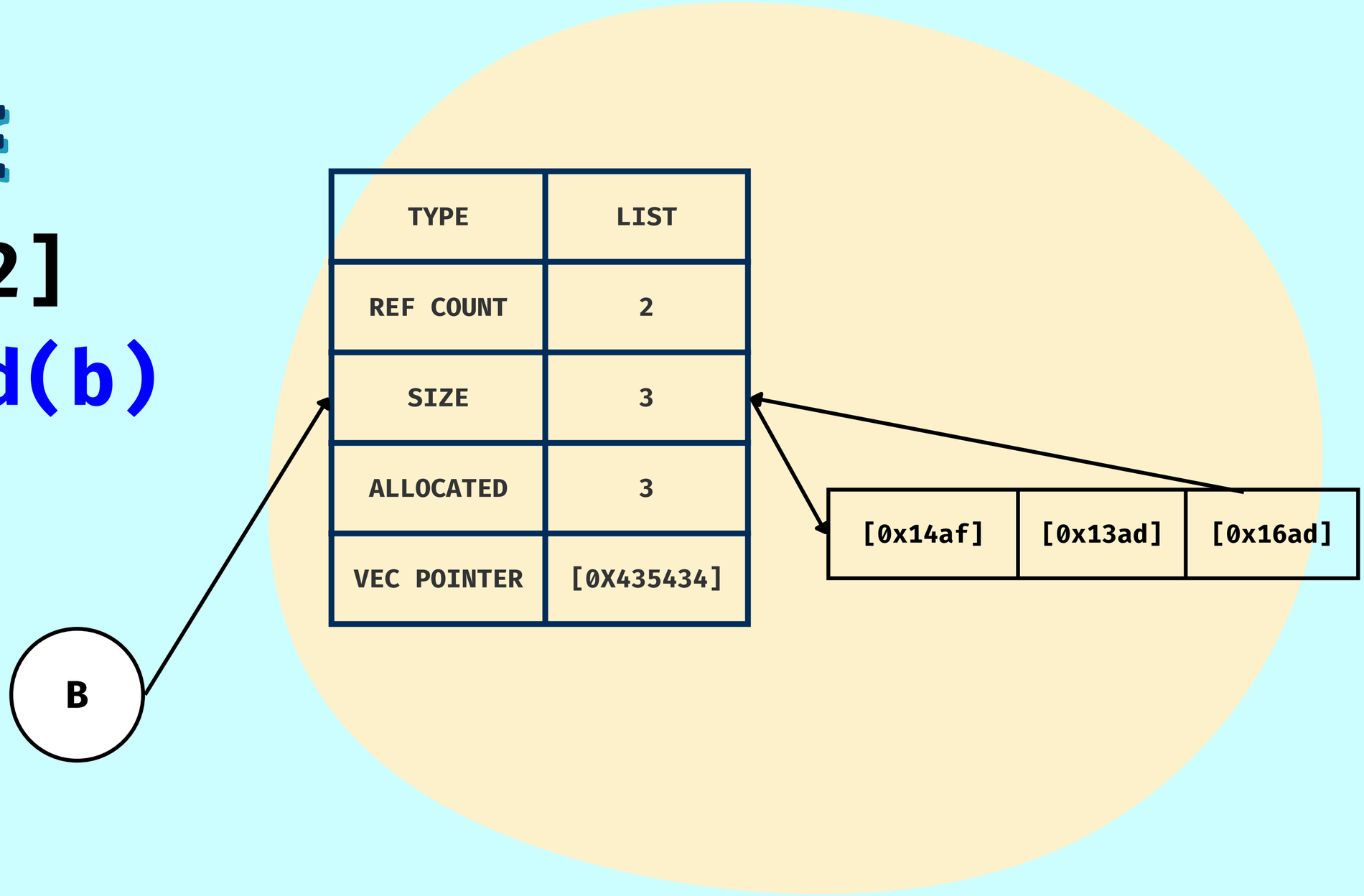
EXAMPLE

```
b = [1,2]
```


B
TYPE
LIST
REF COUNT
1
SIZE
2
ALLOCATED
3
VEC POINTER
[0X435434]
[0x14af]
[0x14af]
private heap

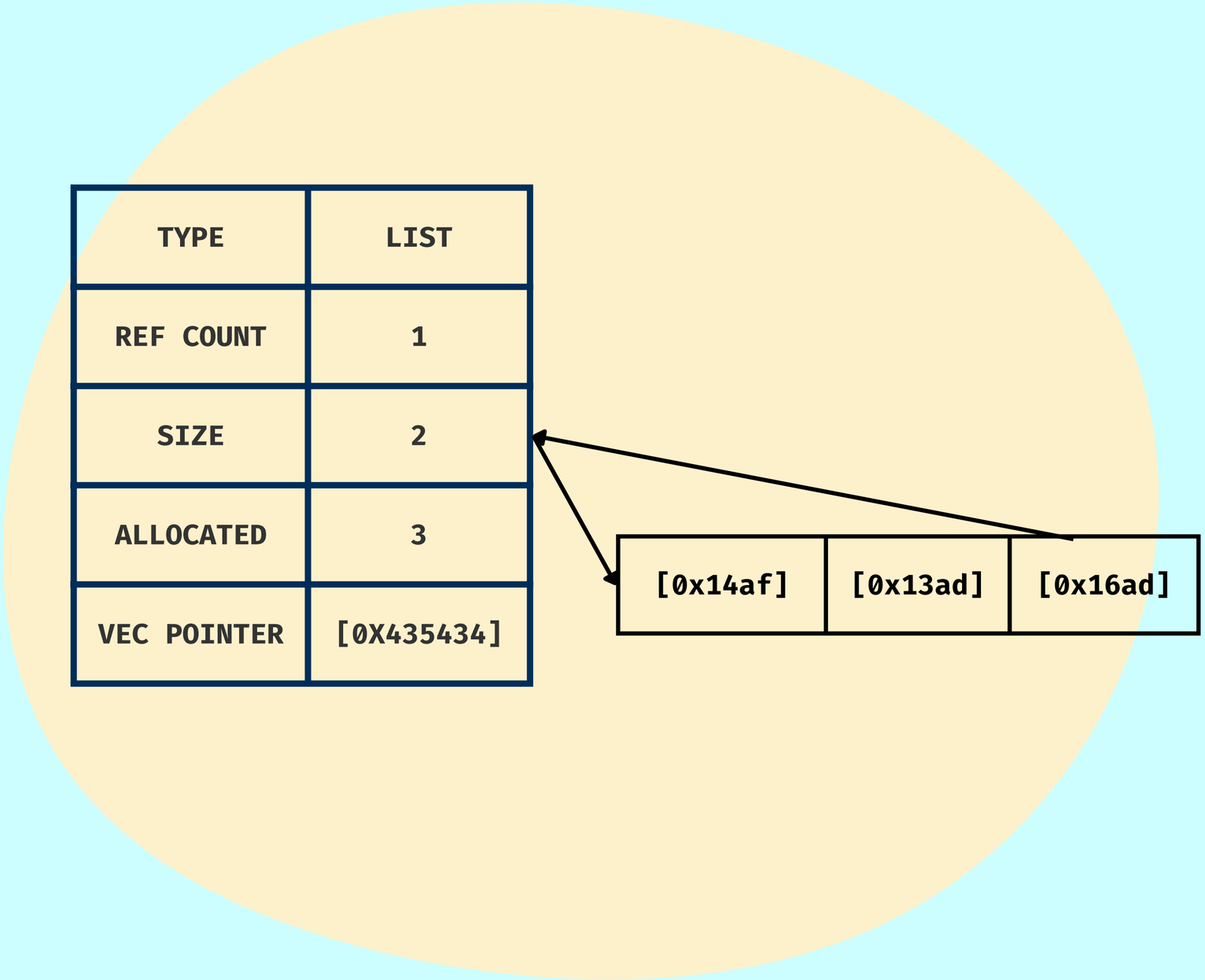
EXAMPLE

```
b = [1,2]
b.append(b)
```



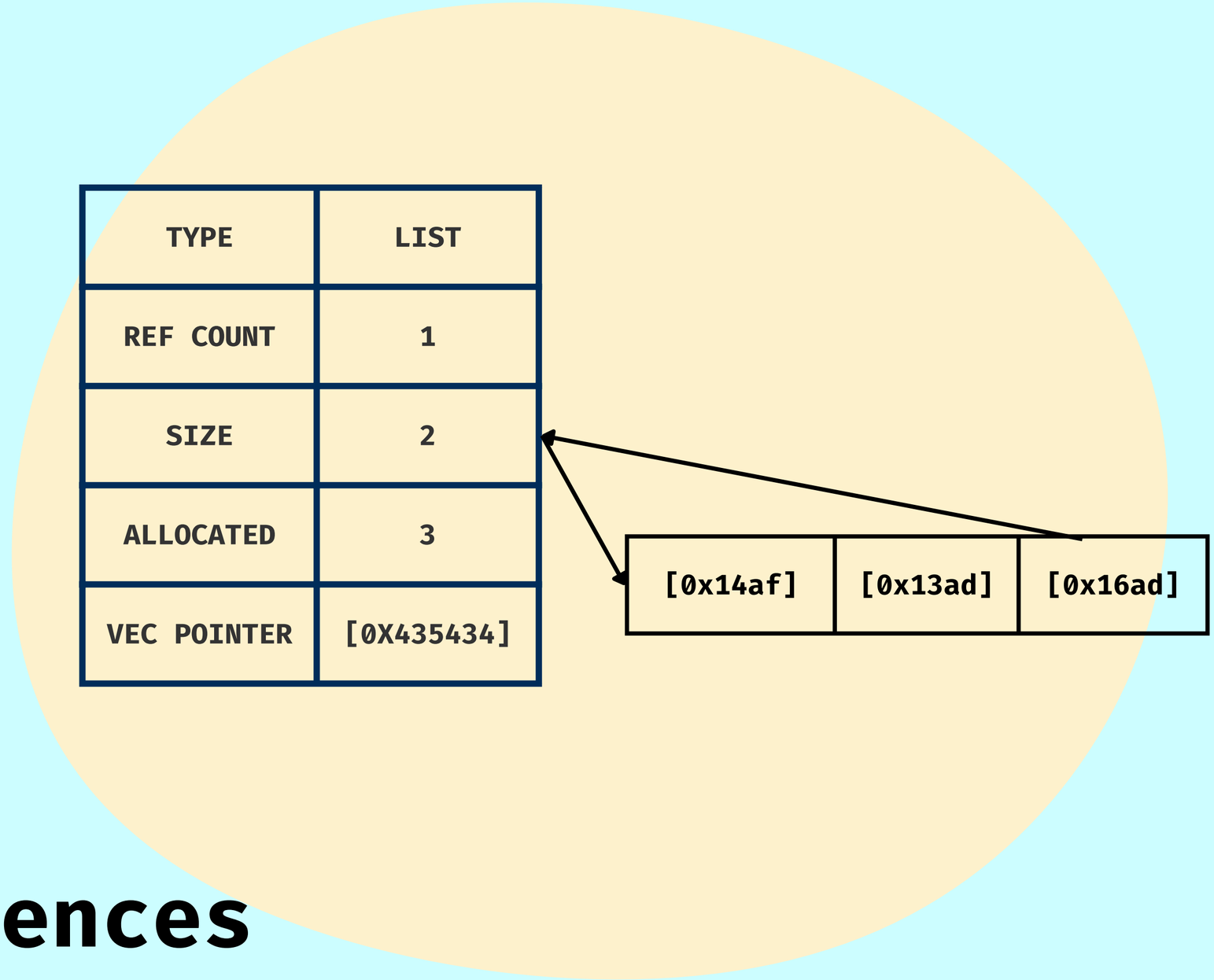
EXAMPLE

```
b = [1,2]
b.append(b)
del b
```



EXAMPLE

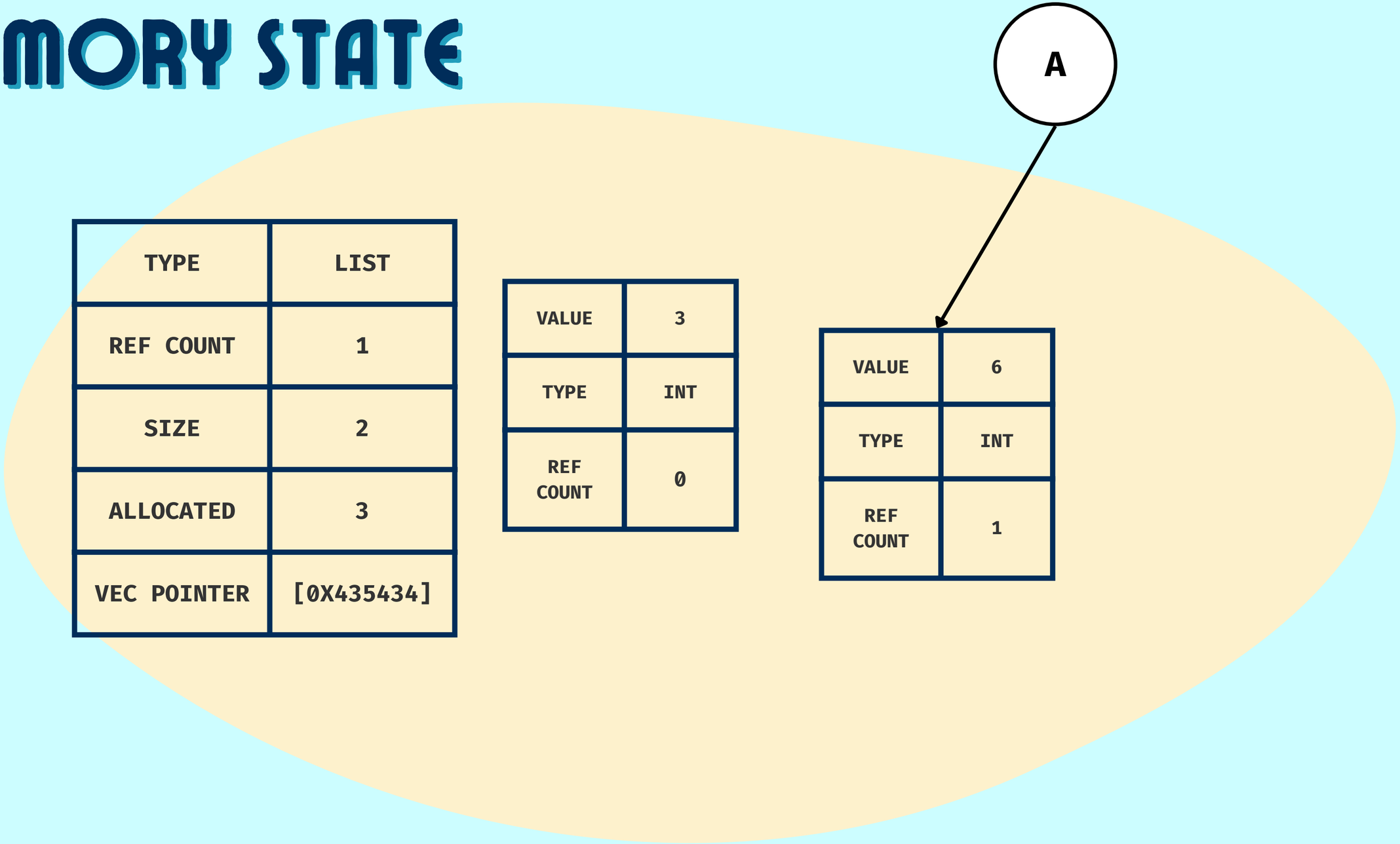
```
b = [1,2]
b.append(b)
del b
```



circular references

private heap

MEMORY STATE



STOP

BUT WHAT ACCTUALLY HAPPENDS TO THE OLD OBJECT

GARBAGE COLLECTOR

- Responsible for collection memory that no longer in used
- It can be trigger manually and by threshold
- Has module gc

GENERATION IN GC

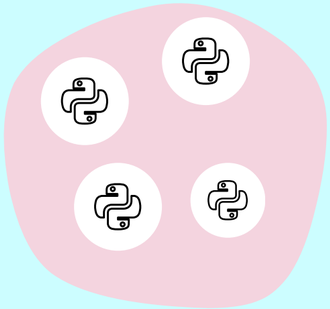
Generation 0

Generation 1

Generation 2

GENERATION IN GC

Generation 0



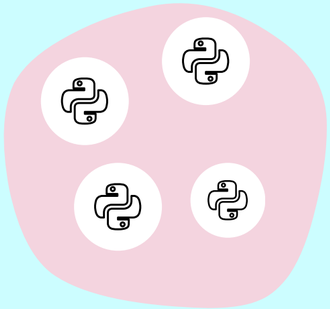
All newly created objects

Generation 1

Generation 2

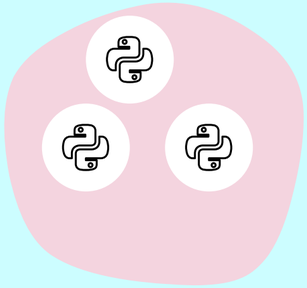
GENERATION IN GC

Generation 0



All newly created objects

Generation 1

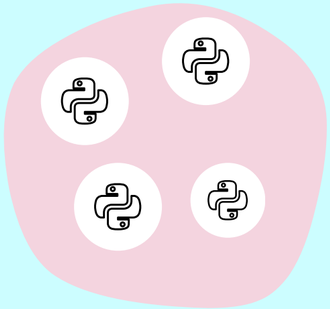


Object survived generation 0

Generation 2

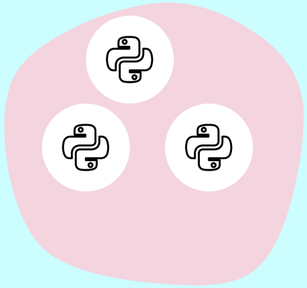
GENERATION IN GC

Generation 0



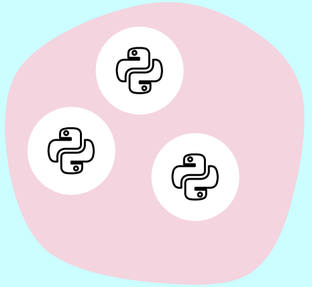
All newly created objects

Generation 1



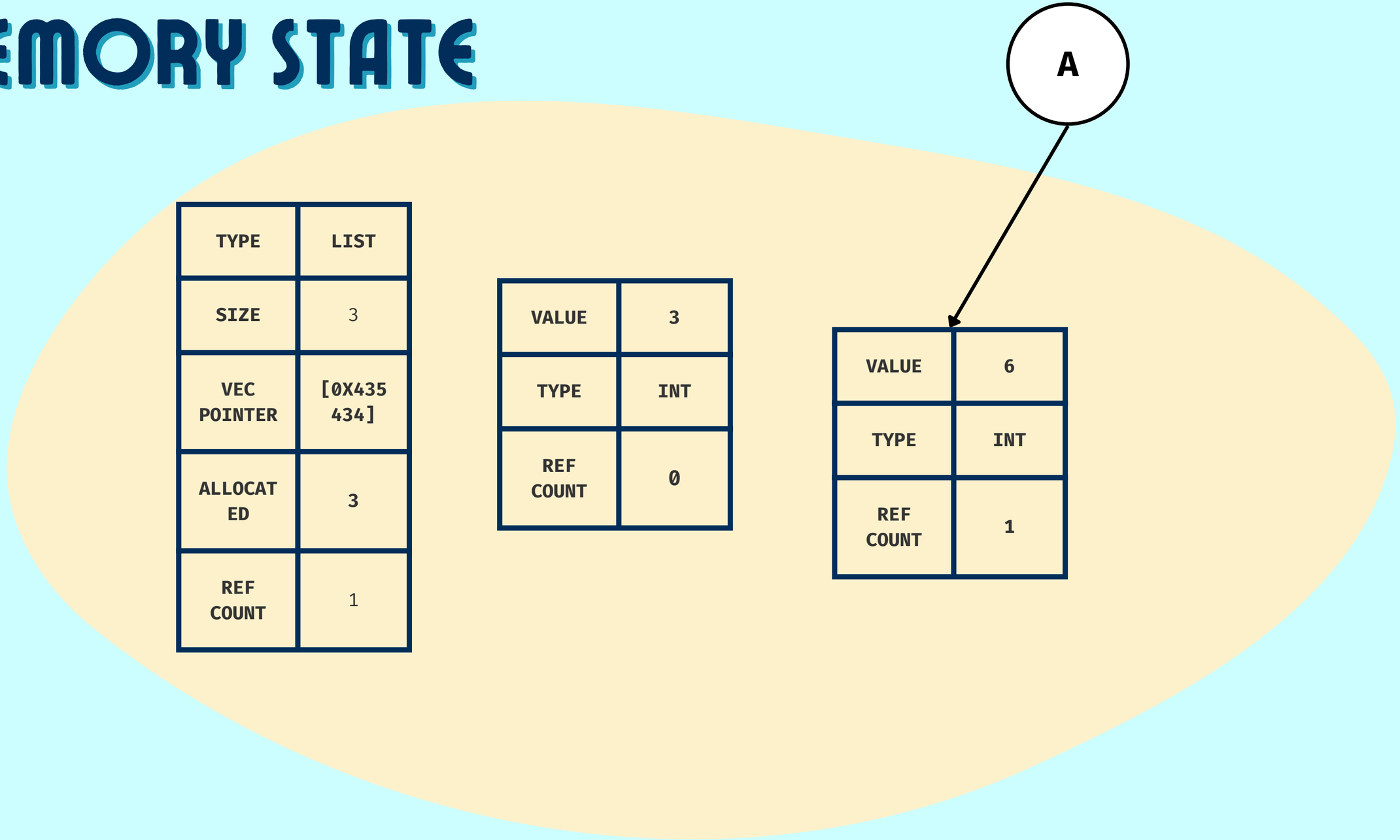
Object survived generation 0

Generation 2

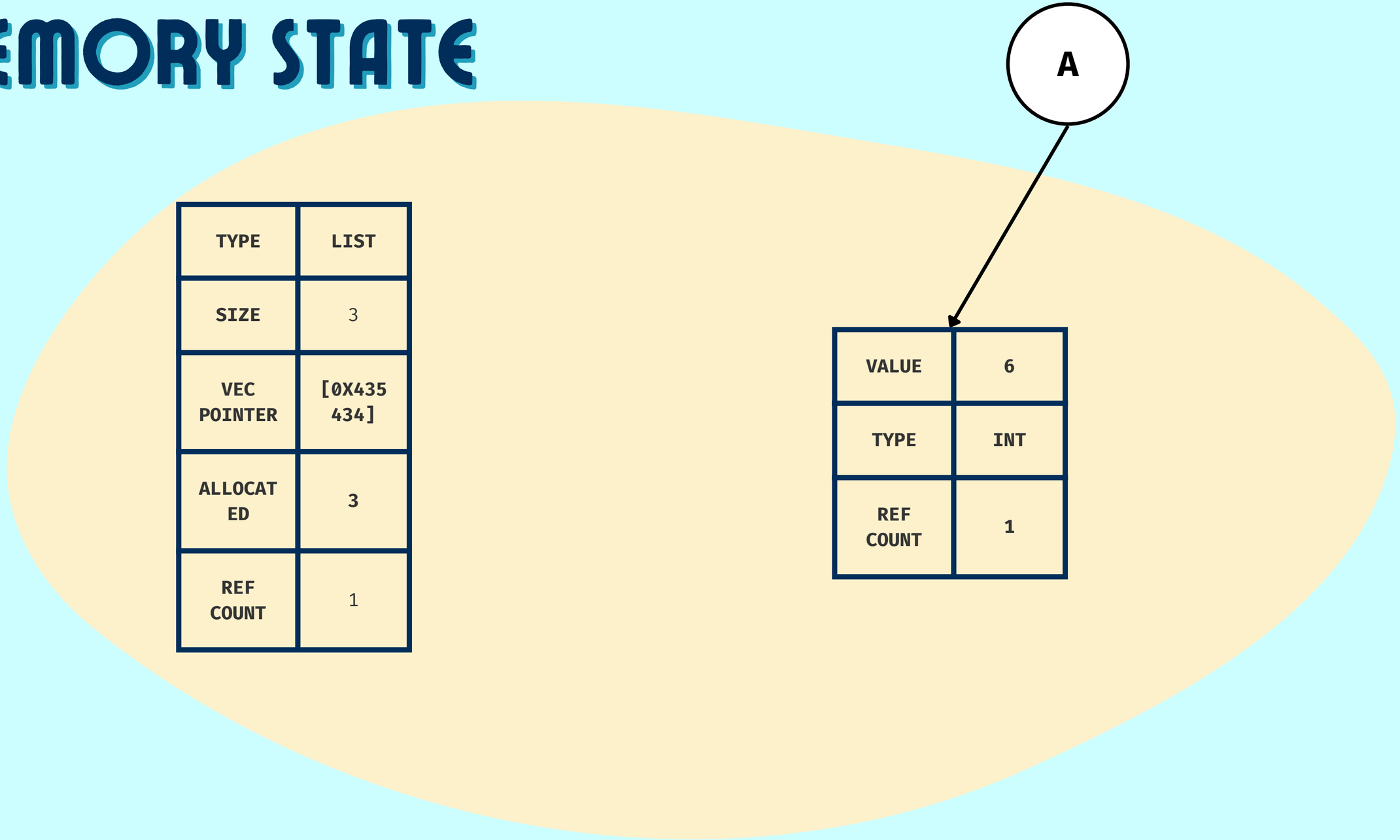


Object survived generation 1

MEMORY STATE

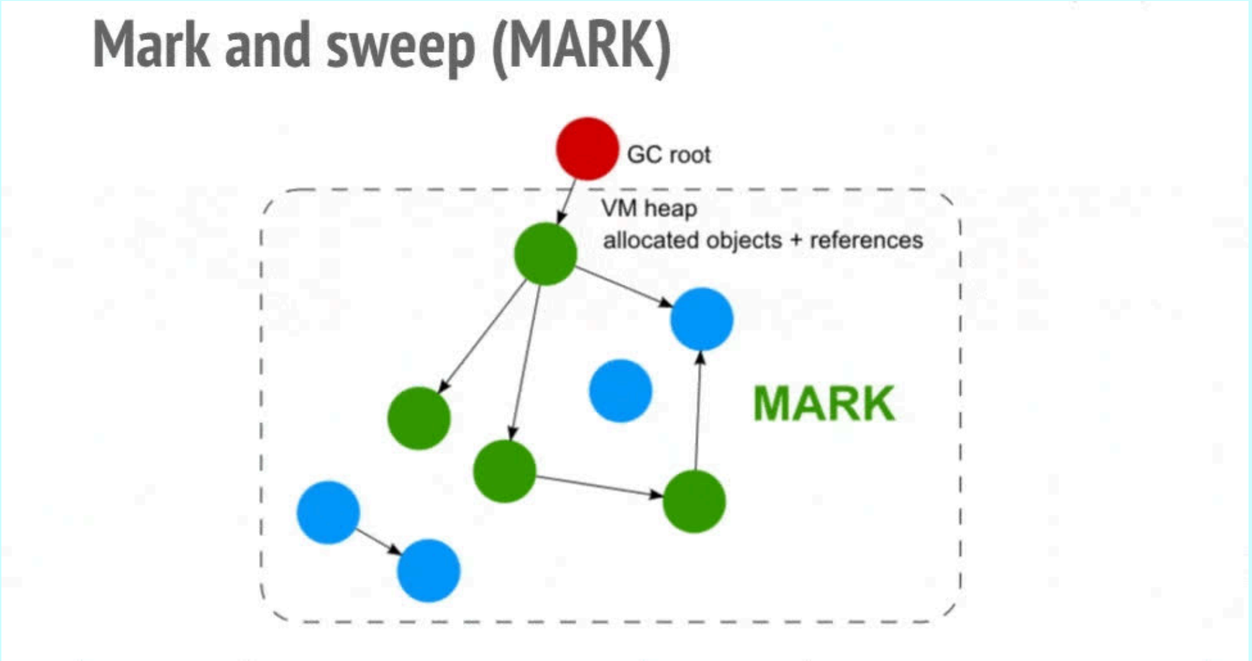


MEMORY STATE

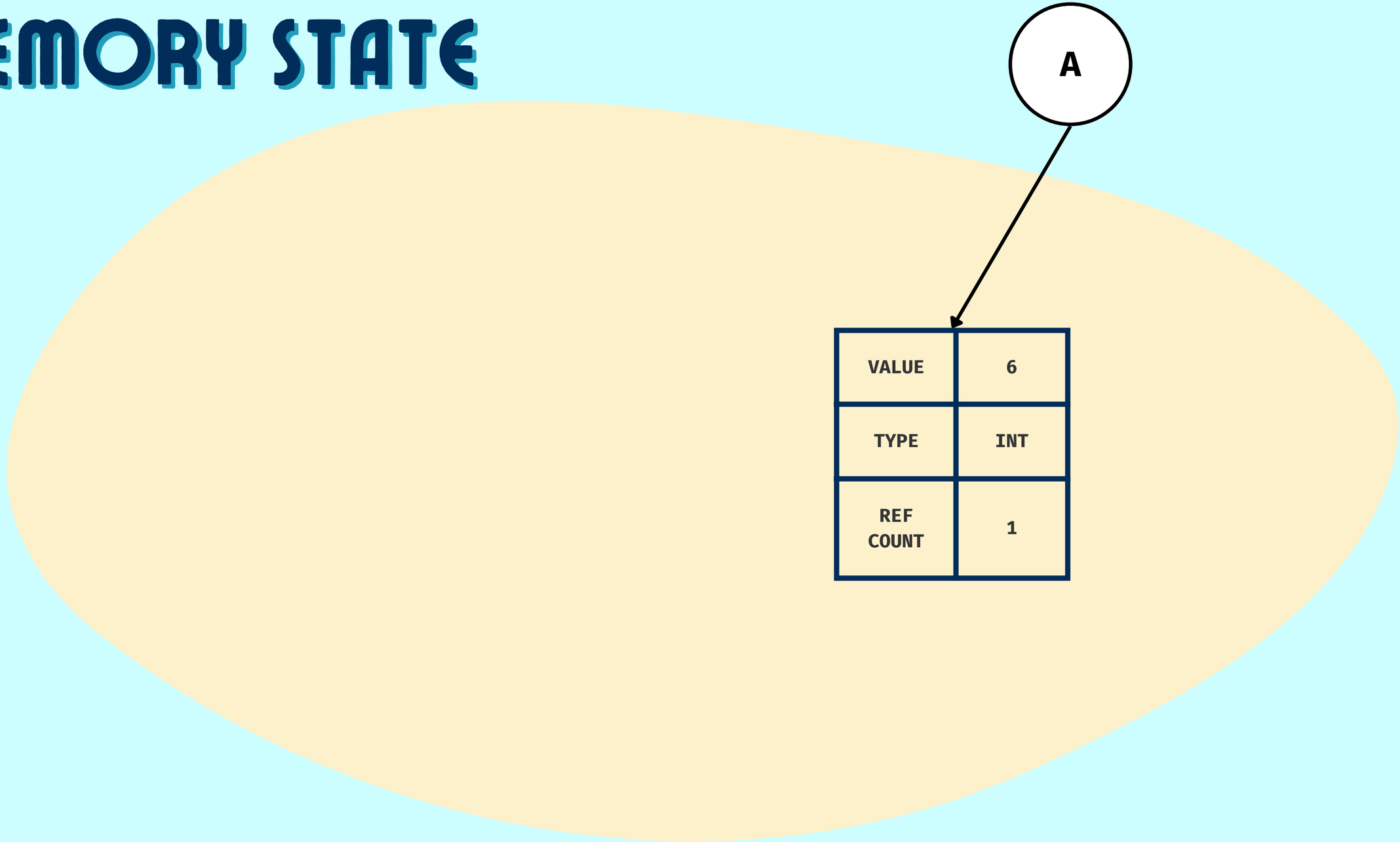


MARK AND SWEET

1. take global variables and local variables as GC root
2. trace all objects that can be reached from GC root
3. sweep unreachable objects



MEMORY STATE



RECAP

- Think about your execution environment
- Everything is object inside Python
- Objects are stored in private heap space
- Memory manager is responsible for private heap space
- Memory manager relies on:
  - reference counting
  - garbage collection

REFERENCES

- Memory Management in Python
- CPython Internals: Your Guide to the Python 3
- High Performance Python
- Talks:
  - PEP 683: Immortal Objects - A new approach for memory managing - Vinícius Gubiani Ferreira
  - Pointers? In My Python? It's More Likely Than You Think! - Eli Holderness
- pythonspeed.com/articles/faster-python/ by Itamar Turner-Trauring