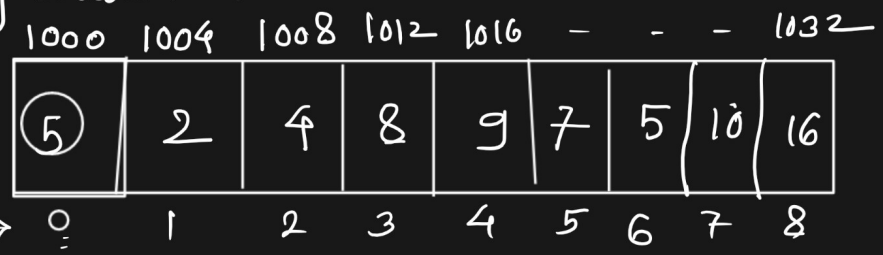
Day 4 :- Array :-

Def^n :- Collection of element of similar types.

- Contiguous Memory allocation



JAVA

```
int arr[] = new int[size];
```

int → 4 byte

Python :- arr = []

Type
- 1D Array
- 2D Array

Access the location

arr[0] ⇒ 1000 location → 5

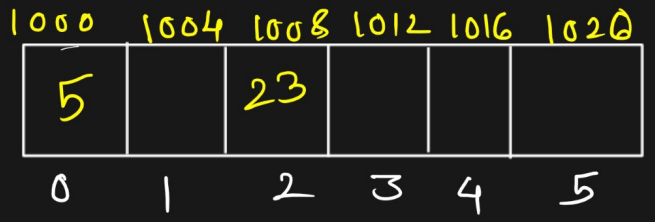
Print

↳ arr ⇒ address of array 1st element ⇒ 1000

&arr ⇒ base address of array = 1000

*arr → * astrick ⇒ * → value at

*arr → *(1000) → value at (1000) ⇒ 5



arr[0] ⟹ 1000 location?

arr[2] ⟹ 1008 ?

arr[0]

arr[i] = *(arr + i * size of data type)

= *(1000 + i * 4byte)

arr[0] = *(1000 + 0×4)

= *(1000) = 5

int

arr[2] = *(arr + (2×4))

= *(1000 + (8))

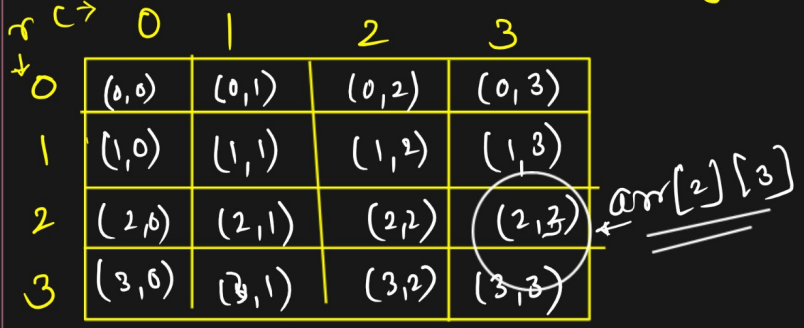
= *(1008)

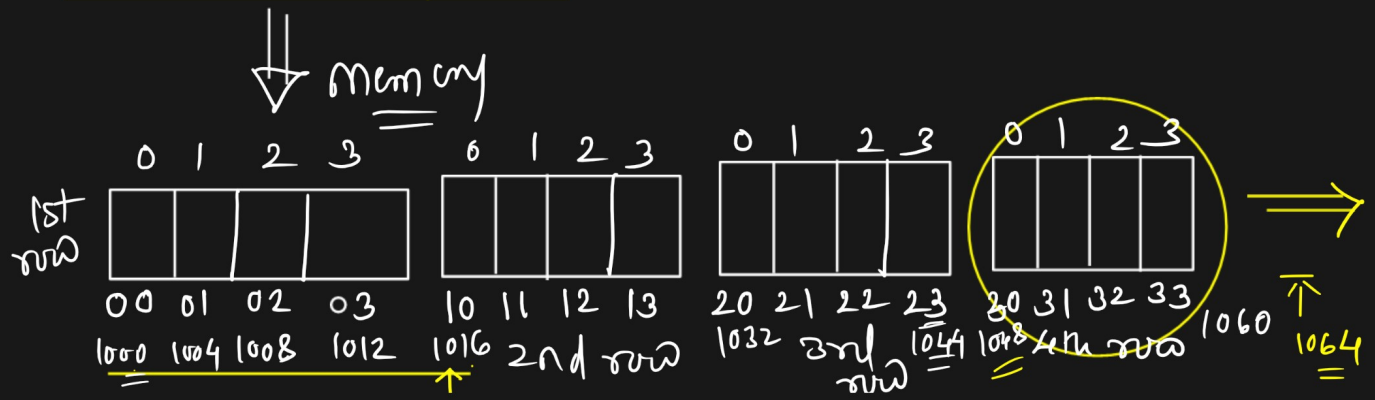
= 23

JAVA ⟹ 2D Array :— Syntax:-

data type array name [ ] [ ] = new data type [row] [column]

int arr [ ] [ ] = new int [4] [4]





$arr[2][3] = *(1044) =$

row size = 16 byte

$arr[i][j] = *(arr + (i * sizeofrow) + (j * sizeof datatype))$

$i * 16 + j *$ 4byte

i → row index, j → clmn index, arr → base address

$arr[2][3] = *(arr + (i*16) + (j*4))$

$= *(1000 + (2\times16) + (3\times4))$

$= *(1000 + 32 + 12)$

$= *(1000+44)$

$= *(1044) = 22$

JAVA

Python

1) Array declr ✓
2) Array elements initialize ✓
3) Display array ele ✓
4) Search

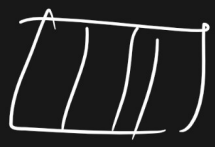
JAVA :- int n ;

```
int arr[] = new int[n]  ⟶ declr
```

```
for ( int i=0 ; i<n ; i++)
 {
        arr[i] = sc.nextInt();
 }
```

i

1000 → 0
1004 → 1
1008 → 2
1012 → 3



Operation on Array :-

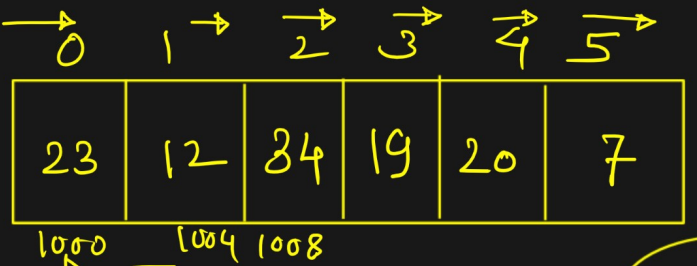
1) Declr
2) Initialization
3) Display
4) Search

Search in Array

Two Algorithms

- Linear Search
- Binary Search

Linear Search :-



→ Search_ele = 27

→ int flag = 0;

for(int i=0; i<n; i++)

{ if (arr[i] == Search_ele) (True)

{ flag = 1; print("Elem is present", i);

return;

}

}

if (flag == 1)

S.o.p ("Element Found");

else

S.o.p ("Element is not Found");

Traversing → arr[i] (1000, 1004, 1008) → i value change? → for loop

i = 0; arr[0] == 27 → 23 == 27 X

i = 1; arr[1] == 27 → 12 == 27 X

i = 2; arr(2) == 27 → 34 X

i = 3; arr(3) == 27 → 19 X

i = 4 → 19 == 27 → 20 X

i = 5 → 7 == 27 X

Time Complexity = O(n)

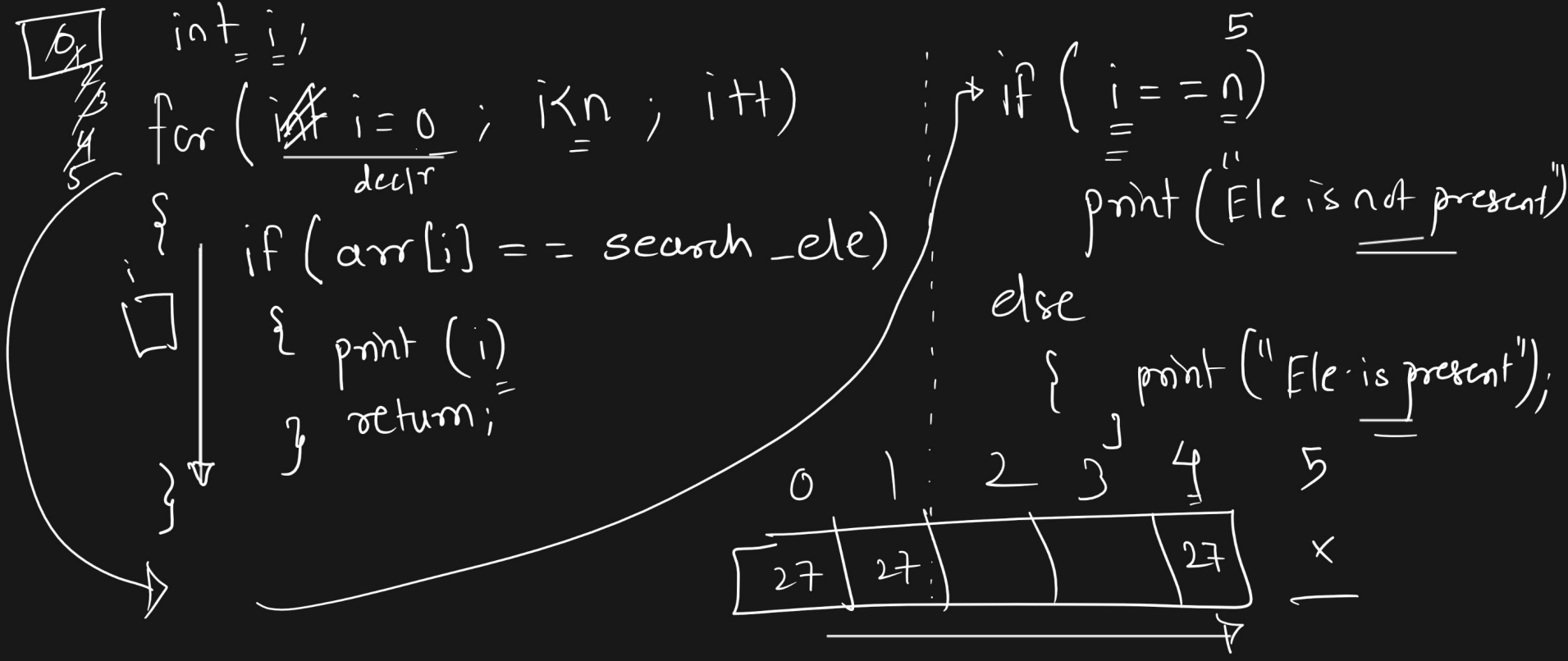
i=6 For X

for (int i=0; i<n; i++)

Can We Avoid Flag variable in Linear Search? ⇒ 4 byte

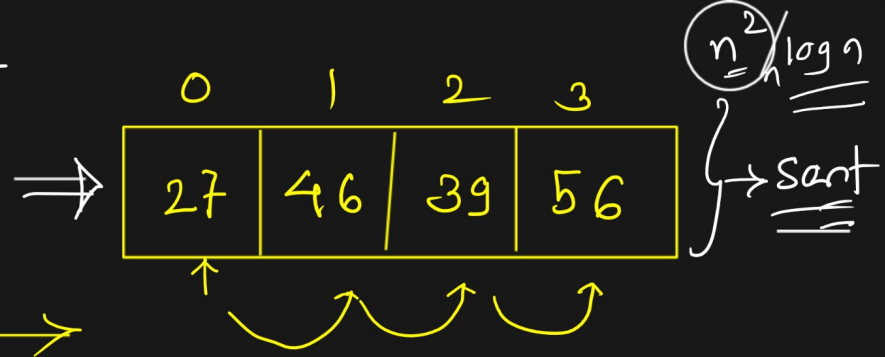
int search_ele = 27

n=5 i = 0, 1, 2, 3, 4, 5 ← stop for

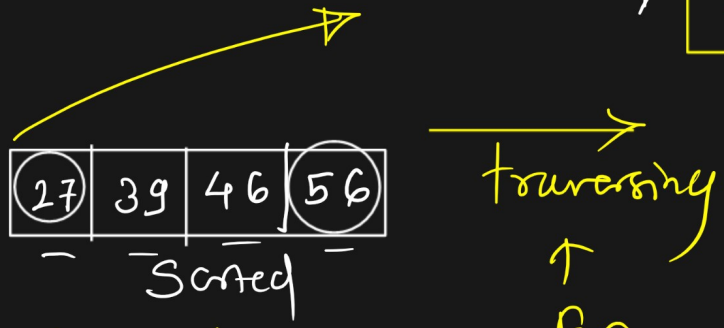


* Largest Element from Array :-

Ans :- 56



Brute Force :-



for ( )
{
for ( ) $O(n^2)$

Time complexity ↓

int largest_ele = arr[0]

for (int i=0; i<n ; i++)
{
if (largest_ele < arr[i])
{ largest_ele = arr[i];
}
}

Time complexity
$O(n)$

return largest_ele;

$O(n)$

	largest_ele = 27	largest_ele
i = 0	27 < 27	27
i = 1	27 < 46	46
i = 2	46 < 39 X	46
i = 3	46 < 56	56
i = 4	For X	

H.W. Find out second_largest element & second_smallest element from given array.