Today's agenda

- Intro to arrays
  - Syntax
  - Storing values
  - Reading input
- Return sum of arr[] elements
- Return max of arr[] elements.
- Array with functions
- Swap 2 indexes
- Reverse array
- 1 more problem

// Intro to array

```
int a = 1;
int b = 2;
int c = 3;
int d = 4;
.
.
.
int j = 10;
```

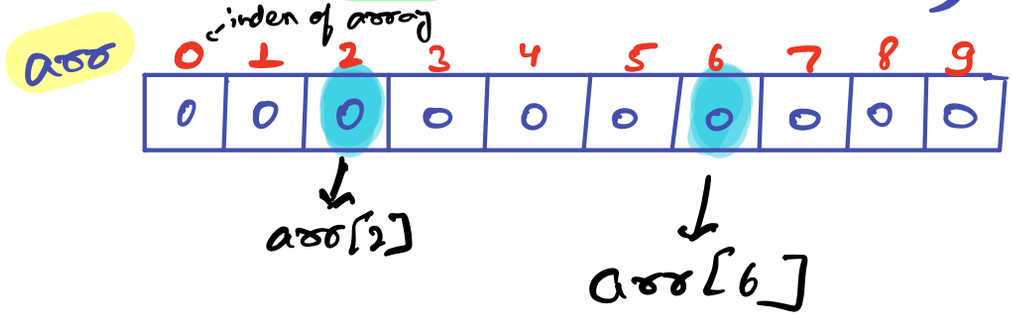
100 Students
!!
to store 100 variables
↓
Arrays

Arrays Syntax

```
type [] name = new type [size];
```

Q) Create an array of size 10 containing integers.

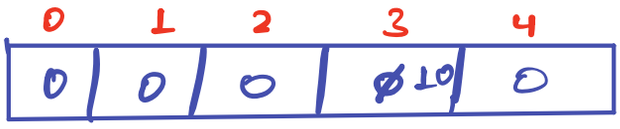
```
int [] arr = new int [10];
```



Indexing & Properties

↳ `int[] arr = new int[5];`

arr



→ `System.out.println(arr[3]);` → 0

`arr[3] = 10;`

→ `System.out.println(arr[3]);` → 10

→ `System.out.println(arr[5]);` → error (index out of bound)

→ If array is of length N.

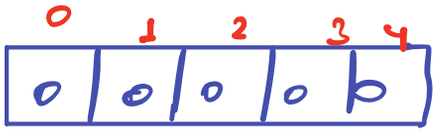
1st index = 0;
last index = N-1;

Q) Create an array of length 5 with values 10 20 30 40 50.

Way 1:

Step1: Create the array.

↳ int [] arr = new int [5];



Step2: assign the values.

```
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;
```

Way 2:

```
int [] arr = {10, 20, 30, 40, 50};
```

(0 1 2 3 4)

```
System.out.println(arr[3]);
```

→ 40

→ to get size of array: arr.length;

Q) Sum of array

↳ Read an array of N length and Print the Sum of all elements.

Eg: arr[4]: 10 -1 3 -7 ⇒ 5 (indices: 0 1 2 3)

//Pseudo Code

```
void main( ) {
    Scanner scn = new Scanner (System.in);
    int n = scn.nextInt();
    int [] arr = new int[n];
    for (int i=0; i<n; i++){
        arr[i] = scn.nextInt();
    }

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

        Sum = Sum + arr[i];
    }
    s.o.p(sum);
}
```

Tracing

	0	1	2	3
arr[4]:	10	-1	3	-7

```
int Sum = 0;
for (int i = 0; i<n; i++){
    Sum = Sum + arr[i];
}
S.O.P(Sum);
```

Sum = 0

i	i<n	Sum
0	t	10
1	t	9
2	t	12
3	t	5
4	f	

f → exit

45

Q) Max of array elements

↳ Read an array of N length and Print the max of all elements.

	0	1	2	3	
Ex: arr[4]:	10	-1	3	-7	→ 10
arr[4]:	-10	-20	-30	-40	→ -10

//Pseudo Code

```
void main( ) {
    Scanner scn = new Scanner (System.in);
    int n = scn.nextInt();
    int [] arr = new int[n];
    for (int i=0; i<n; i++) {
        arr[i] = scn.nextInt();
    }

    int max = 0;        → max = arr[0];
                        ↳ max = -∞ (Integer.MIN_VALUE)

    for (int i=0; i<n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
        else {
            // nothing
        }
    }

    S.O.P (max);
}
```

```
int max = 0;

for(int i=0; i<n; i++){
    if(arr[i] > max){
        max = arr[i];
    }
    else{
        // nothing
    }
}
```

arr[5]:

0	1	2	3	4
10	-1	3	-7	20

max: 0

i	i<n	arr[i]>max	max
0	t	t	10
1	t	f	10
2	t	f	10
3	t	f	10
4	t	t	20
5	f		

↳ exit

max: -∞

arr[4]:

0	1	2	3
-10	-20	-30	-40

↳ 0

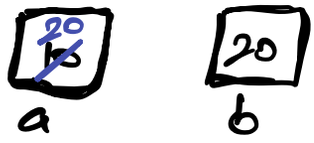
Break till 7:30 PM

Q) Swap the values of 2 variables

a = 10 b = 20 ⇒ a = 20 b = 10

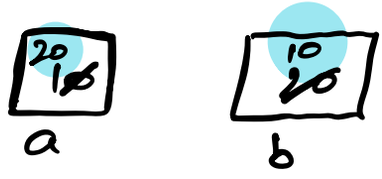
// incorrect way

```
void main() {
    int a = 10;
    int b = 20;

    a = b;
    b = a;
}
```



// Correct way

```
void main() {
    int a = 10;
    int b = 20;

    int temp = a;
    a = b;
    b = temp;
}
```



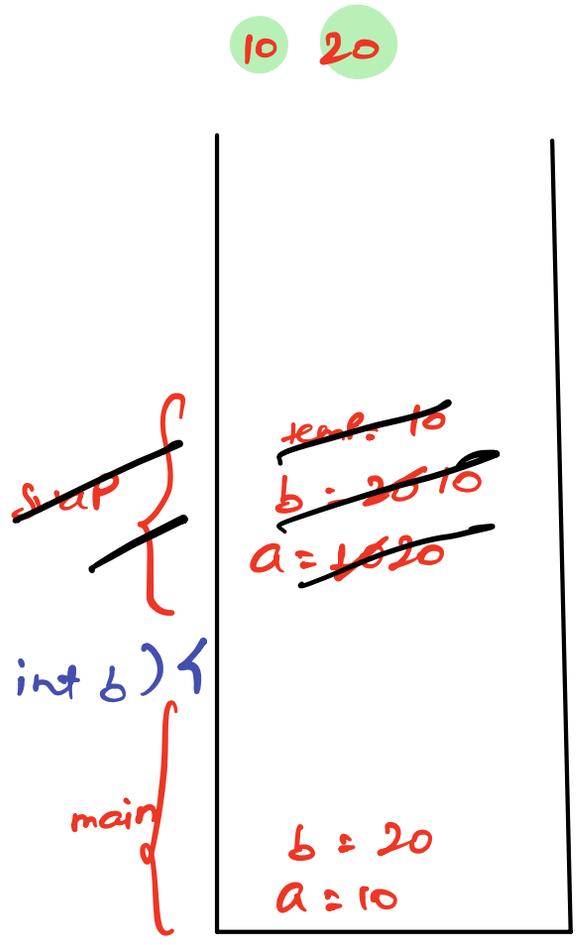
// Arrays with functions

```
main ( ) {
    int a = 10;
    int b = 20;
    swap (a, b);
    S.O.P (a);
    S.O.P (b);
}
```

→ 10

→ 20

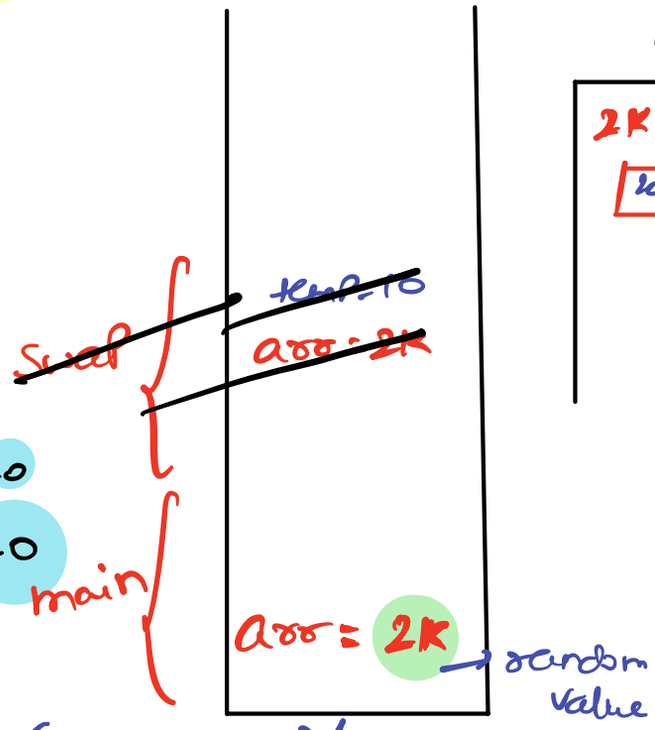
```
Public static void swap (int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```



→ variables of 2 functions are not connected.

Q) → 20 10

```
void main() {
    int[] arr = {10, 20};

    swap(arr);

    s.o.p(arr[0]);   → 20
    s.o.p(arr[1]);   → 10
}
```



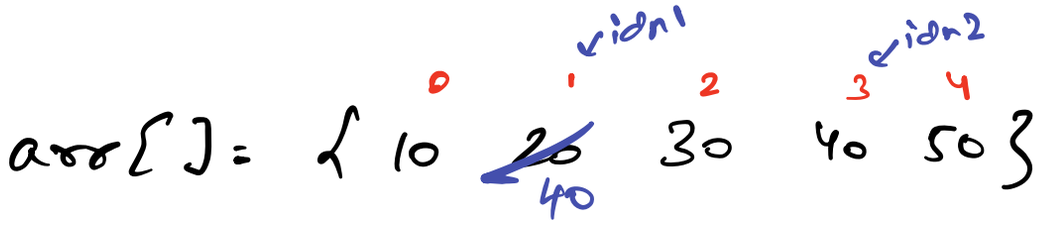

```
Public static void swap(int[] arr){

    int temp = arr[0];
    arr[0] = arr[1];
    arr[1] = temp;
}
```

→ arrays across functions are always connected.

Q) Swap indexes

↳ Given array of length N and two indexes idx1 and idx2, swap the element of those two indexes.

↳ done just before this page



idx1 = 1 idx2 = 3

```
int temp = arr[idx1];
arr[idx1] = arr[idx2];
arr[idx2] = temp;
```

Q) Reverse array

↳ Given array of length N, Reverse the whole array.

ex: arr[5]: { 10 20 30 40 50 }

0 1 2 3 4

↓

50 40 30 20 10

Combination of Swaps : reverse

arr[5]: { 10 20 30 40 50 }

0 1 2 3 4

4 3 1 0

↓

50 40 30 20 10

Swap (0, 4)

Swap (1, 3)

arr[10] :

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100
9	8	7	6	5	4	3	2	1	0
100	90	80	70	60	50	40	30	20	10

[Swap (0, 9)
Swap (1, 8)
Swap (2, 7)
Swap (3, 6)
Swap (4, 5)] : reverse the array

//Pseudo code

```
int main() {
    // arr input

    reverse(arr);
}
```

P S void reverse (int arr[ ]){

```
int sP: 0
int eP: arr.length-1;
```

Swap (0, 9)
Swap (1, 8)
Swap (2, 7)
Swap (3, 6)
Swap (4, 5)

```
while (sP<eP) {

    int tempP = arr[sP];
    arr[sP] = arr[eP];
    arr[eP] : tempP;
      sP++;
      eP--;

}
```

}

```
int sp = 0
int ep = arr.length-1;

while (sp<ep) {
    int temp = arr[sp];
    arr[sp] = arr[ep];
    arr[ep] = temp;
    sp++;
    ep--;
}
```

arr[9] = 90 80 70 60 50 40 30 20 10

sp	ep	sp < ep
0	8	t
1	7	t
2	6	t
3	5	t
4	4	f → exit

```
int sp = 0
int ep = arr.length-1;

while (sp<ep) {
    int temp = arr[sp];
    arr[sp] = arr[ep];
    arr[ep] = temp;
    sp++;
    ep--;
}
```

arr[6] = 60 50 40 30 20 10

sp	ep	sp < ep	sp != ep (won't work)
0	5	t	t
1	4	t	t
2	3	t	t
3	2	f → exit	t