* Loops

```
while (conditions) {
    —
    —
    —
}
```

Print number from 1 to 10.

```
int counter =1
while (counter <=10) {
   sysout print(counter);
   counter++;
}
```

```
1
2
3
4
5
6
7
8
9
10
```

Print ni

Print no.s 1 to n

```
~~Scanner sc range~~
~~Sysout ("Enter the range;");~~
~~scanner sc = new Scanner (System.in);~~
~~int n = sc.nextInt ();~~
~~while (~~

    int count= 1;                  value
    Sysout ("Enter the n:");
    while ( count <=n ) {
        sysout ( count);
        count++
      }
    }
}
```

DRY RUN

sum=0
i= 1 → i<=n → sum=1+0

sum=1 → i++=2

sum= 2+1
=3
↓
i++ → 3
↓
sum = 3+3 = 6 → 6+4 = 10 → 10+5 → 5

for loop

```
for ( Initialisation; condition ; updation) {
}
```

Initialisation → counter / iterator

condition → i<=10, <=100

updation → i = i+2, i = i+1

DRYRUN

1st i=1
↓
i<=10 ✓
↓
"Hello world"
↓
i++
↓
i=2

square pattern

```
*   *   *   *
*   *   *   *
*   *   *   *
*   *   *   *
```

```
for ( int i =1 ; i<=4; i++) {
    for ( int j= 1; j<= j ; j++){
      print ("*");
      }
    }
  }
}
```

Print Reverse of a number.

n=10899.

① Last digit pohije n % 10
② Last digit remove karaychay n/10

```
public class JavaBasics {
    public static void main (String args[]){
        int n = 10899;
        while (n>0) {
            int lastDigit = n %10;
            System.out.print (lastDigit + " ");
            n=n/10; //n/=10
        }
    }
}
```

Only the print reverse

Last digit →

print karnar

last digit kadnar

Reverse of the given number
(Here we need to store a number)

$\boxed{rev = (rev * 10) + last\ digit}$

storing the reverse number)

rev = 0
rev = (0*0)+9 = 9
rev = (9*10)+9 = 90+9 = 99
rev = (99*10) +8 = 990+8 = 998
rev = (998*10)+ 0 = 9980
rev = (9980*10)+1 = 99800+1 = 99801

```
int rev=0;
while (n>0) {
    Lastdigit
    rev = rev * 10 + LD
    n update
}
print (rev)
```

```
public class JavaBasics {
    public static void main (String args []) {
        int n= 10899;
        int rev=0;
        while (n>0) {
            int last Digit =n %10
            rev = (rev *10) + lastDigit;
            n=n/10;
        }
        System.out.println(rev);
    }
}
```

n madhe no. stored

the reversed print karaycha

Last digit

operation perform

Last digit removed to get next no. at last place do-while Loop

```
do{
   // do something
} while (condition);

public class JavaBasics {
   public static voidmain (String args[] ){
      int counter =1;

   do{
      System.out.println ("Hello World");
      counter++;
   }
   while(counter <=10);{  ——  only
                             condition
   ~~counter++~~

   }
}
```

Break statement.

```
Public class JavaBasics{
   Public static void main (String []args){
      for (int i=1; i<=5; i++){
         if (i==3){
            break;
         }
         sysout (i);
      }
   }
}
```

Q Keep entering numbers till user enters a multiple of 10.

Means →		print
1	⟶	1
5	⟶	5
7	⟶	7
3	⟶	3
20	⟶	×

```
~~int n~~;
do {
int n = sc.nextInt;
if (n % 10 == 0) {  ⟵ checks multiple of 10
  break;
}
sysout (n);
} while (true);
}
}
```

Continue Statement. —— skips the iteration.

```
public class JavaBasics{
   public static void main (String []args){
      for (int i=0 ; i<=5; i++){
         if (i==3){
            continue;
         }
         System.out.println(i);
```

Display all numbers entered by user except multiples of 10

		print
1	⟶	1
2	⟶	2
7	⟶	7
10	⟶	skip
20	⟶	skip.

```
public class JavaBasics{
    public ~~class~~ static void main
                  (String[] args){
        Scanner sc = new Scanner (System.in);

        do{
            ~~int n = sc.nextInt~~
            Sysout ("Enter your number:");
            int n = sc.nextInt
            if (n % 10 == 0){
                continue;
            }
            Sysout ("The number is:" ~~"is multiple"~~, n);
        } while (true);
    }
}
```

Check if a number is prime or not

Multiples ke half tak dhoondo.

```
① 1×12 }  ascending       n = 1×n
② 2×6  }  order              = n×1
③ 3×4  }
  4×3
  6×2                    n = √n × √n
  12×1
```

$$\uparrow \begin{matrix} 4 \times 4 \\ 2 \times 2 \\ 4 \times 1 \end{matrix} \downarrow$$

prime no. print Kartana

2 to n-1 —— to check whether no is prime or not

Suppose n = 12

```
 ┌──────┐
 │ 1×12 │      12 % 4 == 0
 │ 2×6  │      12 % 3
 │ 3×4  │
 └──────┘
   4×3
   6×2
   12×1
```

```
public class JavaBasics.
    public void main (String args[]){
        Scanner sc = new Scanner (System.in);
        int n = sc.nextInt ();
if(n==2){ →  boolean isPrime = true;                changes to (for optimising)
Sysout("prime");    for (int i=2; i<= n-1; i++){      → Math.sqrt(n)
}else{              if (n % i == 0) { //n is a multiple
                    isPrime = false;   of i (i is not
                                         equal to 1 or n)
                }
            }
            if (isPrime == true){
                System.out.println ("n is prime");
            }else{
                System.out.println ("n is not prime");
            }
        }
    }
```

10) star pattern

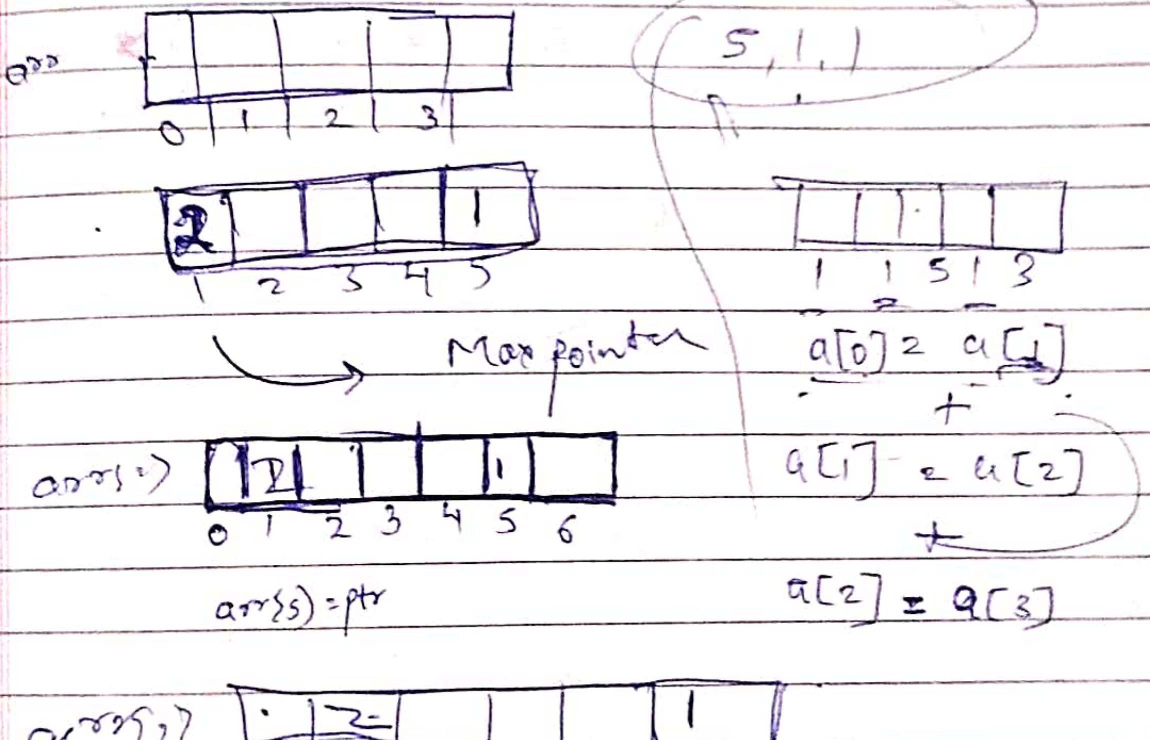
```
*            (1)
* *          (2)
* * *        (3)
* * * *      (4)
```

① ~~4~~ times
outer loop → 4 times
② number of times
inner loop

```
n = 4;
for (i=1 ; i<=n ; i++) {
    for ( j=1 ; j<=i ; j++) {
        Sysout ("*");
    }
  }
  println ();
 }
}
}
```

(i = 4 ; i ~~<~~=n ; i++

11)



Nested Loops

```
line = 1      star = ~~1~~ 2 → out of loop     for (line = ~~1~~; line<= 4;
line = 2      star = ~~1~~ ~~2~~ 3                    line++) {
                out of loop                   for(star = 1; star<=4;
                                                  star++) {
                                              ~~for (star = 1, star~~
line = 3      star = 1, 2, 3, ~~4~~ Out of loop
line = 4      star = 1, 2, 3, 4 ~~5~~ Out of loop
```

```
*   *   *   *  —  i = 1 , j ~~stars~~ = 4
*   *   *         i = 2 , ~~star~~ j = 3
*   *             i = 3 , j = 2            Star (n-i+1)
*                 i = 4 , j = 1

  for (int i = 1; i <= 4; i++){
     for(int j = 4 ; j ~~*~~ = i ; j--){
        sysout ("*");
     }
  }
```

```
1
1 2
1 2 3
1 2 3 4           n = 4
                  for ( i = 1; i <= n ; i++) {
                     for( j = 1; j <= i; j++){
                       sysout print (j);
                     }
                  ~~{~~  println();
                  }
```

DRY RUN — o/p

```
int n = 4
for( i = ~~1~~ 2; i <= n ; i++) {
    for ( j = 1; j <= i ; j++) {
        print (j);
    }
    println();
}
```

```
i = 1               , j = 1
line = 1            num = 1 2 → condn false
  ↓                   ↓
                     + 1
i++                 number = 1 ——— again initialize
                    num = ~~1~~ 2 ——→ +1
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

65

PAGE No.
DATE / /

Character Pattern.

A	line = 1, char = 1
B C	line = 2, char = 2
D E F	line = 3, char = 3
G H I J	—11— = 4, —1— = 4

① Outer loop ← line
② inner loop
↑
character

③

char ch = 'A' ~~'B'~~ ~~'C'~~ ~~'D'~~ ~~'E'~~ ~~'F'~~ ~~'G'~~ ~~'H'~~ ~~'I'~~ ~~'J'~~

data type

```
char       ch = 'A'
  int n = 4;
for (int i = 1; i <= n; i++) {
    ~~for ( int~~
      for( int j = 1; j <= i; j++){
      sysout print (ch);
           ch ++;
    }
     sysout print ln (ch);
  }
```

inner loop j>1 asta aani ch 'B'
asto so print 'B' hota aani
chtt hota parat condⁿ check hoti
aani 'C' ~~hota~~ print parat j++ hota
but i aanij ehi condⁿ discard hota
aani outer loop madhe jaun new line
print hoti.

Functions — block of code which follows a function

Syntax

Function is like a button takes some input & gives some output

```
return type name() {
   // body
   return statement;
}
```

return type → output ka type

→ means after performing certain operations what will be return type of the code.

```
public class JavaBasics {
    public static void main (String args[]) {
```

main → ~~output~~ Return type

(String args[]) → input

~~Acc modifier~~

Return type — reserved keyword

Ques : Why we need a functⁿ ?
Bec in main function we need to write the whole logic again & again which increase time & space complexity. To avoid this & increase reusability we create new function & call it through main function. This user define function is used to write the whole logic.

── reusable block of code.

```
}public class JavaBasics
public static void PrintHelloWorld {
    Sysout println ("Hello World");
}
public static void main (String args[]) {
}
```

execution always starts with main function in java (mostly in all languages)

```
public static void main (String args[]) {
    print HelloWorld ();
}
}
```

Syntax with Parameters.

```
return Type name (type param1, type param2) {
    ~~return~~ // body
    return statement;
}
```

(type param1, type param2) → Input of function.

```
import java.util.*;
public class JavaBasics {
    public static void print HelloWorld () {
        System.out.println ("Hello World");
```

~~public static void calculatesum () {~~
~~int sum = a+b;~~
~~system.out.println ("~~

```
import java.util.*;
public class JavaBasics {
    public static void print HelloWorld () {
     print ("HelloWorld");
  }

    public static void calSum () {
     Scanner sc = new scanner (system.in);
      int a = sc. nextInt ();
      int b = sc. nextInt ();
      ~~int~~ sum = a+b ;
      sysout print ( "sum" + sum);
  }

     public static void main (String args[]) {
      CalSum ();
  }
```

Function mein jo input aate hai wo "parameters" bolte hai

formal parameters → function definition

Actual params → call (Mainfunction)

```
public static void calculate sum (int num1,
                                  int num2) {
    int sum = num1 + num2;
    Sysout print ("sum is " + sum);
}
  public static void main (String args [])
  {
    Scanner sc = new scanner (system in);
    int a = sc.nextInt();
    int b = sc.next Int();
    calculateSum (a,b);
  }
```

nos ko receive kiya as a parameters

functions mein nos pass kiye

Another way to calculate sum.

Pura yaycha calculatesum through the yenar.

```
public static void calculate sum (int num1, int num2) {
     int sum = num1+num2;
     ~~system.out~~ ("sum is:" Sum);
     return sum;
  public static void main (string args[]) {
    Scanner sc = new Scanner (System.in)
    int a = sc.nextInt();
    int b = sc.next Int ();
    int sum=calculate sum (a,b);
    Sysout print ("The sum is :" + sum);
  }
```

void → int type

Sum ko return kiya jisne isse call kiya means main function → ko return kiya.

numbers la pass kele

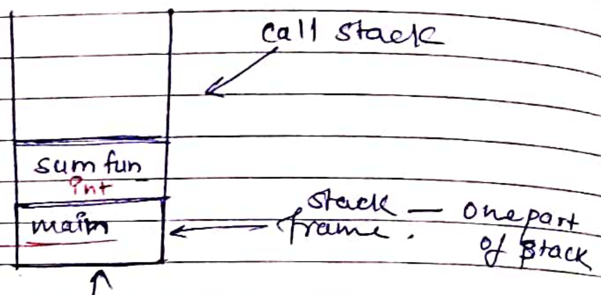
what happens in Memory.

```
Public static void sum (int a, int b)
{
    int sum = a+b;
}

P.S. void main (String arg[]) {
    sum ()
}
```

String of arguments

Functions la jya veli apan call karto tya veli memory occupy ~~hota~~ hot aste



call variables of "main functions" gets stored here.

① Main function madhe String type ~~chi value~~ store zaleli aste

② & when we call the sum sum func will be occupied in stack frame & type will be int & if this func calls another func^n new stack frame will be created in stack

** Inputs variables are stored in stack.

Whenever we call a function memory gets reserved for variables & inputs present in ~~respective~~ respective func^n

Return keyword func^n chya end la takla ki sum ha function ransh hota after performing its role & "main" function la jar return nahi kela tr ~~apo~~ automatically after completing it task call stack gets ~~roi~~ empty.

* call By value

```
Public static void swap (int a, int b)
    ~~int a =~~
    temp = ~~int~~ a;
    a = b;
    b = temp;
②
Public Static void main (String args[])
{
    int a = 5;
    int ~~b~~ b = 10;
    swap (a, b);
    ~~Sysout~~
    Sysout (a);
    Sysout (b);
```

① In 1st round o/p remains as it is

② o/p — a = 10
         b = 5

① o/p = a = 5
         b = 10

call by value - function's change will be ~~that~~ limited to itself

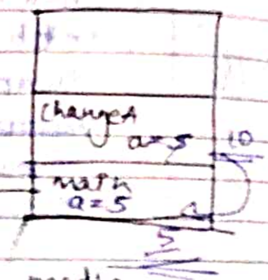
swap function madhe don ~~parameters~~ parameters swap la
dilet int a & int b
call kartana je arguments asta (a & b)
tyancha copy kartat.

Means swap function madhe je variable pass zalet aani jevdhe pn changes
zali te fakta swap function
paryant applicable rahnar
that's why number aren't swaped
they remain as it is ! Main
function madhe change nahi honar

Call by value

```
change A (int a) {   <- formal parameter
    a = 10           copy.
}
ps void main ( ) {
    int a = 5.
    change A (a)  <- main argument; cha
    syso(a)
}
```

What happens in memory?
when you call the function



main function la call lavla
ki call stack madhe
variable store hota.

~~call by~~ * Java always works on call by value term *

Another meaning of call by value ;
" value di hai means the copy of value not the actual value "

* Call by reference (pending)

```
P.S. Int Prod (int a, int b) {
Always   int prod = a*b ;
as it is return prod
or
access
modifier }

~~int prod~~ ~~product~~ (3,5) ;
  ~~print (prod)~~
  int product = prod (3,5);
  print (product)
```

hyat store honar.   kgacha ans

```
Public static int prod (int a, int b ) {
    int mult = a*b
    return mult
}
    P.S. void main(String args []) {
    int product = ~~prod~~ prod (3,5) ;
    print (product)
```

```
public static int factorial (int n) {
    int fact = 1;
    for (int i = 1; i<=n; i++) {
        fact = fact * i;
    }
    return fact;
}
```

Binomial coeff.

```
public static int binCoeff (int n, int r) {
    int fact_n = factorial (n);
    int fact_r = factorial (r);
    int fact_nmr = factorial (n-r);

    int bincoeff = fact_n/((fact_r) * fact_nmr);
    return binCoeff;
}

public static void main (string args[]){
    system.out.println (binCoeff (5, 2));
}
}
```