```
        system.out.println(");
    }
```

★ FUNCTIONS & METHODS

- Function is a block of code that ~~is~~ perform specific or particular work as many times.

* Function is reusable.

Syntax: returntype function-name () {
    // body
    return statement;
}

- public static void main (String args[]) {
    // body
}

  void → returntype (खाली)
  main() → main()

* public static - Access modifier
* void - return type, Output.
* main - function, reserved Word.
* String args[] - ~~Out~~ Input of main().

* If we're not using void then there should be a return type.
* Functions that are written inside the class is called Methods.

• Syntax with Parameters

* Parameters are nothing but inputs given inside function.

Syntax->
```
returntype function_name (data-type param1, data-type param2)
{
    //body
    return statement;
}
```

* Parameters are of two types :
  i) Formal parameters or parameters (definition of functions)
  ii) Actual parameters or arguments (calling of functions)

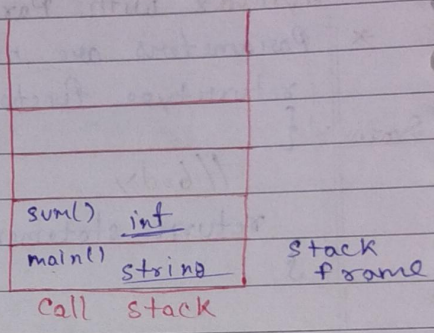
Ex->
```
public static int sumoftwo calculatesum (int num1, int num2) {
                                          parameters
int sum = num1 + num2;
return sum;
}

psvm {
  Scanner sc = new Scanner(System.in);
  syo ("Enter num1: ");
  int a = sc.nextInt();
  syo ("Enter num2: ");
  int b = sc.nextInt();
                          arguments
  int sum = calculatesum (a,b);
  syos ("Sum is: " + sum);
}
```

- What happens in Memory?

* Function also take memory.
* Function are stored inside call stack.

```
psvm sum (int a, int b)
{
  int sum = a+b;
  return;
}
psvm (String args[])
{
  sum();
}
```



* In stack frame, all the information i.e., function name, data types, variables etc are stored.
* When the return type is executed, functions are removed from call stack memory.

- Call by Value & Call by reference

* Java functions always uses Call by value.
* Java doesn't use Call by reference.

* If we call a method passing a value, it is known as call by value.
* The changes being done in the called method, is not affected in the calling method.

* In case of call by reference, original value is changed if we made changes in the called method.

Ex→ Factorial using function

⇒
```
public static int factorial (int num) {
    int fact = 1;
    for (int i = 1; i<=num; i++) {
        fact *= i;
    }
    return fact;
}
psvm (string args[]) {
    Scanner sc = new Scanner (System.in);
    int n = sc.nextInt();
    int f = factorial (n);
    Syso ("Factorial is: " + f);
}
```

Ex→ Binomial Coefficient

* Input n & r. * ${}^nC_r = \frac{n!}{r!(n-r)!}$

⇒
```
public static int factorial (int num) {
    int fact = 1;
    for (int i=1; i<=num; i++) {
        fact *= i;
    }
    return fact;
}
public static int bincoef (int n, int r) {
    int n_fact = factorial (n);
    int r_fact = factorial (r);
    int nr_fact = factorial (n-r);
    int bincoef = n_fact / (r_fact * nr_fact);
    return bincoef;
}
```

- **Types of functions or Methods**

There are two types of functions :

In-Built	User-Defined
⇓	⇓
~~It is available in Java.~~ ~~Made by user~~	Made by user.
⇓	⇓
~~factorial, sum~~, main(), math(), ~~product, etc.~~ sc.nextInt() etc.	factorial, sum, product, etc.
⇓	⇓
It can't be changed.	It can be changed.

- **Function Overloading**

* Multiple functions with the same name but different parameters in data type or no. of parameter.

Ex → multiply (int a, int b) , → add (int a, int b)
multiply (float a, float b) add (int a, int b, int c)
multiply (double a, double b)

* Function overloading doesn't react on return type.
* Function overloading only react on no. of parameters and type of parameters.

- **Prime or Not**

```
public static boolean prime (int n) {
    boolean isPrime = true;
    for (int i=2; i<=n-1; i++) {
        if (n%i==0) {
            isPrime = false;
        }
    }
    return isPrime;
}
```

```
psvm {
    Syso (prime(5));
}
```

- Prime or Not (Optimized way)

```
Public static boolean isPrime (int n) {
    for (int i=2; i<= Math.sqrt (n) ; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
psvm {
  syso (isPrime (199));
}
```

- Prime in Range

```
public static void inRange (int n) {
   for (int i=2; i<=n ; i++) {
      if (isPrime(i)) { //above program function
         System.out.print (i+" ");
      }
   }
   Syso();
}
psvm{
  inRange(20); //Print prime no. from 2 to 20.
}
```

- Convert from Binary to Decimal

* Decimal number system → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
ten

* Binary number system → 0, 1
two

* 0 to 99 binary conversion

$(0)_{10} = (0)_2$	$(5)_{10} = (101)_2$
$(1)_{10} = (01)_2$	$(6)_{10} = (110)_2$
$(2)_{10} = (10)_2$	$(7)_{10} = (111)_2$
$(3)_{10} = (11)_2$	$(8)_{10} = (1000)_2$
$(4)_{10} = (100)_2$	$(9)_{10} = (1001)_2$

* n = 1 1 0 1

$2^n \ldots 2^3\ 2^2\ 2^1\ 2^0$ $= 2^3(1) + 2^2(1) + 2^1(0) + 2^0(1)$

$= 8 + 4 + 0 + 1$

$= 13$

$\Rightarrow (1101)_2 = (13)_{10}$.

* Binary to Decimal (Logic)

n = 101

pow = 0

dec = 0

LD = 1 // LD = Lower digit of $(101)_2$ ← LD = n % 10;

dec = dec + [LD * 2^{pow}] = 0 + [1 * 2^0] = 0 + 1 = 1

dec = dec + [LD * 2^{pow}] = 1 + [0 * 2^1] = 1 + 0 = 1

dec = dec + [LD * 2^{pow}] = 1 + [1 * 2^2] = 1 + 4 = 5

∴ ~~$(101)_2 = (5)_{10}$~~

After loop;

pow ++;

binNum = binNum / 10;

* LD = n % 10
Modulus is to extract the last digit.

* binNum = binNum / 10;
Divide symbol is to remove last digit.

* Binary to Decimal

```
public static void binToDec (int binNum) {
    int myNum = binNum;
    int pow=0
    int decNum=0;
    while (binNum >0) {
        int lastDigit = binNum % 10;
        decNum = decNum + (lastDigit * (int) Math.pow(2,pow));
        pow ++;
        binNum /= 10;
    }
    Syso ("Decimal of " + myNum + " = " + decNum);
}

psvm {
   binToDec (101001);   //41.
}
```

* Convert from Decimal to Binary

- n = 7 $\Rightarrow (7)_{10} = (111)_2$

2 | 7			
2 | 3	1		10^0
2 | 1	1		10^1
0	1		10^2

- n = 13 $\Rightarrow (13)_{10} = (1101)_2$

2 | 13			
2 | 6	1		10^0
2 | 3	0		10^1
2 | 1	1		10^2
0	1		10^3

* Pseudo Code:

```
while (n>0) {
    divide by 2
        ↳ remainder
    bin = bin + rem. * 10^pow
    pow ++
    num = num / 2;
}
```

* Code for dec to Binary

```
public static void decToBin (int n) {
    int myNum = n;
    int pow = 0;
    int binNum = 0;
    while (n > 0) {
        int rem = n % 2;
        binNum = binNum + (rem * (int) Math.pow(10, pow));
        pow++;
        n = n / 2;
    }
    Syso ("Binary form of " + myNum + " = " + binNum);
}
```

* Block Scope

```
{
    int s = 5;
    Syso (s);
}
```

5

```
{
    int s = 5;
}
Syso (s);
```

Error
(Out of Block Scope).

QUESTIONS

1. Write a Java method to compute the average of three numbers.

```
public static void average (int a, int b, int c) {
    int avg = (a + b + c) / 3;
}
```

2. Write a method named isEven that accepts an int argument. The method should return true if the argument is even or false otherwise.

```
public static ~~void~~ boolean isEven (int n) {
    if (n % 2 == 0) {
        return true;
    }
    else {
        return false;
    }
}
psvm {
    System.out.print ("Enter Number: ");
    int num = sc.nextInt();
    if (isEven (num)) {
        Syso ("It's Even Number.");
    else {
        syso ("It's Odd Number.");
    }
}
```

3. Write a Java program to check if a number is palindrome in Java?

```
public static void isPalindrome (int n) {
    int palindrome = n;
    int reverse = 0;
    while (palindrome != 0) {
        int remainder = n % 10;
        reverse = reverse * 10 + remainder;
        n /= 10;
    }
```

```
if (palindrome == reverse) {
    Syso ("It is a Palindrome Number.");
else {
    Syso ("It is not a Palindrome Number.");
}
}
```

Ques 4. Write a java method to compute the sums of the digits in an integer.

```
public static int sumofdigits (int num) {
    int sum = 0;
    while (num > 0) {
        int lastDigit = num % 10;
        sum += lastdigit;
        num /= 10;
    }
    return sumof;
}
```

Ques 5. Use the math class in Java & understand it:-

a) Math.min()
b) Math.max()
c) Math.sqrt()
d) Math.pow()
e) Math.avg()
f) Math.abs()

S.No.	Method	Description
		It will return the :
1.	Math.abs()	absolute value of the given value.
2.	Math.max()	Largest value of two values.
3.	Math.min()	Smallest value of two values.
4.	Math.round()	round of the decimal numbers to the nearest value.
5.	Math.sqrt()	square root of a number.
6.	Math.cbrt()	cuble root of a number.
7.	Math.pow()	first argument raised to the power to second argument.
8.	Math.signum()	find the sign of a value.
9.	Math.exp()	E raised to the power of a value, E is euler's number & it is approximately equal to 2.871828.
10.	Math.log()	returns the natural logarithm of double value.
11.	Math.avg()	returns the average of the values.