12 July 2025 10:07

Functions

$f(x) = x^2 + 3$

$f(2) = 2^2 + 3 = 7$
$f(-1) = (-1)^2 + 3 = 4$

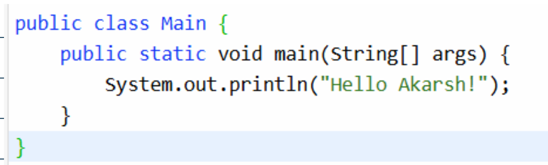
f^n is a rule that assigns exactly one o/p to each ip.

Java

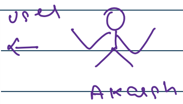
method
a f^n is a block of code that performs a specific task.
You define it once and can call multiple times.

prime number program

prime of number n.
prime of 20 numbers.



```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
    }
}
```



Requirements to execute a program

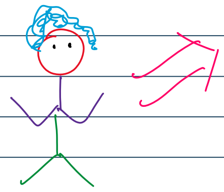
memory → who provides
↓
operating system → acts as an mediator b/w h/w and user

memory → Stack memory (running the program, and managing f^n calls)
→ Heap memory (dynamic memory allocation)

compiler → never executes the program
→ checks the syntax errors

JVM (Java virtual machine)

↳ provide environment for exection .

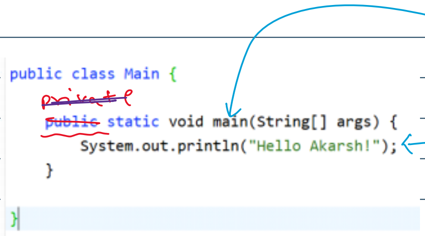


Unique

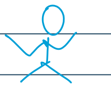
main method java

⇓

always unique in class



```
public class Main {
    ~~private~~
    ~~public~~ static void main(String[] args) {
        System.out.println("Hello Akarsh!"); ←
    }

}
```



JVM

Hello Akarsh!

Return types in function

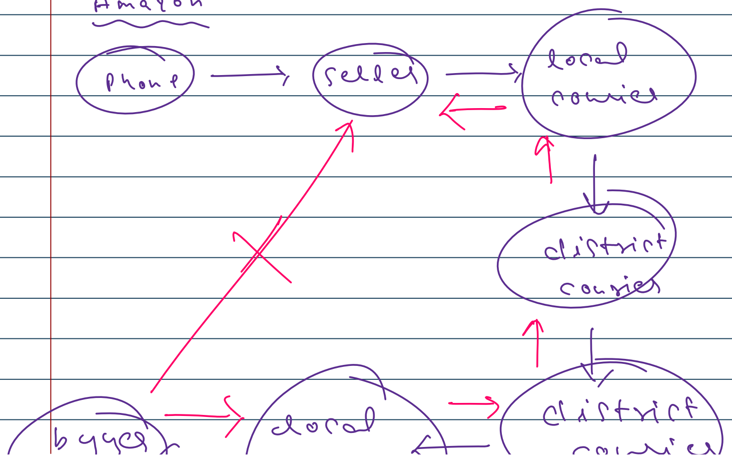
Task
- → I don't bring anything back
- → I bring something back .

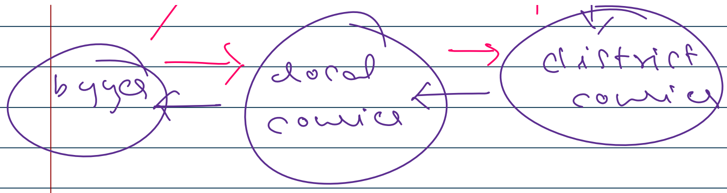
Return types
- → void
- → 2000 ₹ , 10.2 litre petrol
  {int} {float}

Methods
- → parameterized
- → Non-parameterized

} return type
- → void
- → primitive / non-primitive

Amazon



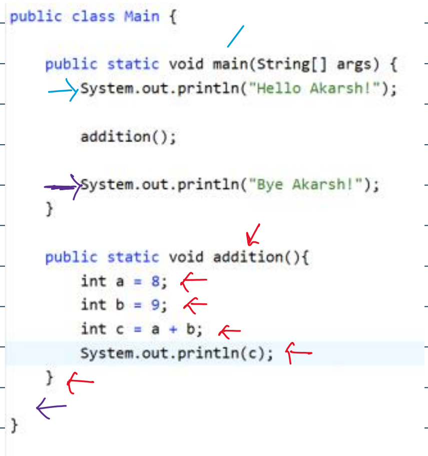


call stack

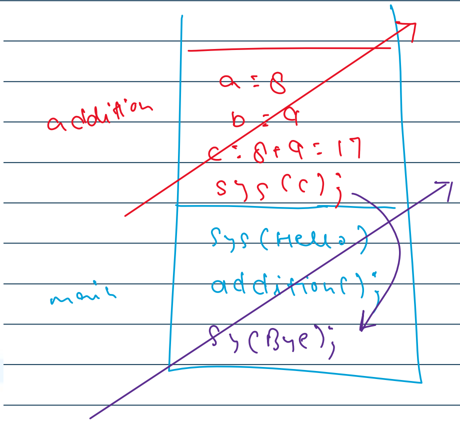
helps in managing the fn call.

↳ it keeps the sequence of fn call.

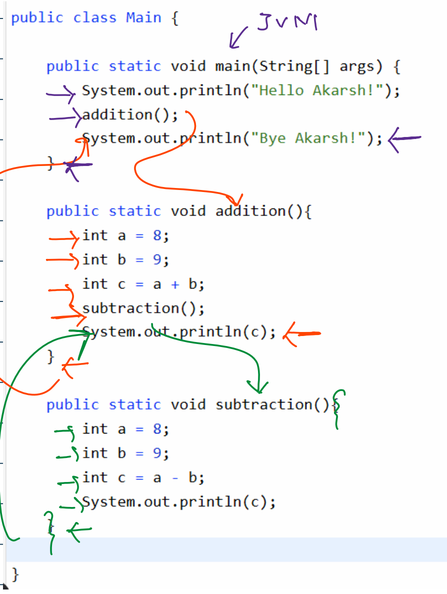
(1) currently executing fn
how arrived at that point.



```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");

        addition();

        System.out.println("Bye Akarsh!");
    }

    public static void addition(){
        int a = 8;
        int b = 9;
        int c = a + b;
        System.out.println(c);
    }
}
```

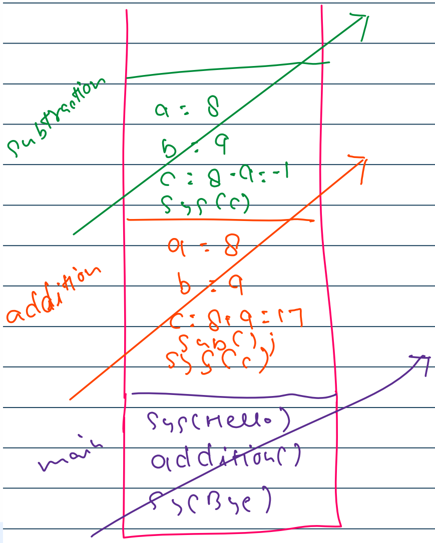


Hello Akarsh!
17
Bye Akarsh!

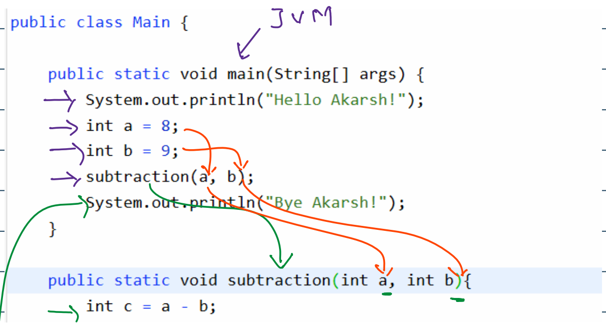


JVM

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        addition();
        System.out.println("Bye Akarsh!");
    }

    public static void addition(){
        int a = 8;
        int b = 9;
        int c = a + b;
        subtraction();
        System.out.println(c);
    }

    public static void subtraction(){
        int a = 8;
        int b = 9;
        int c = a - b;
        System.out.println(c);
    }

}
```

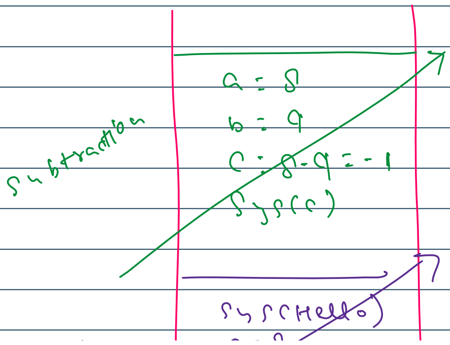


Hello Akarsh
-1
17
Bye Akarsh

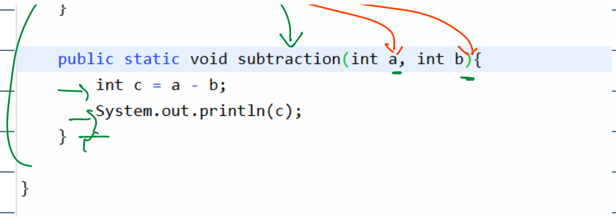


JVM

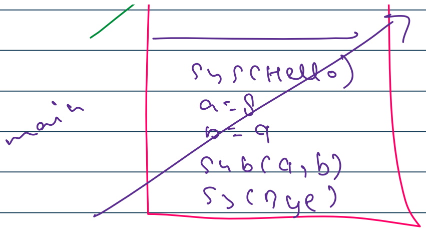
```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        subtraction(a, b);
        System.out.println("Bye Akarsh!");
    }

    public static void subtraction(int a, int b){
        int c = a - b;
```

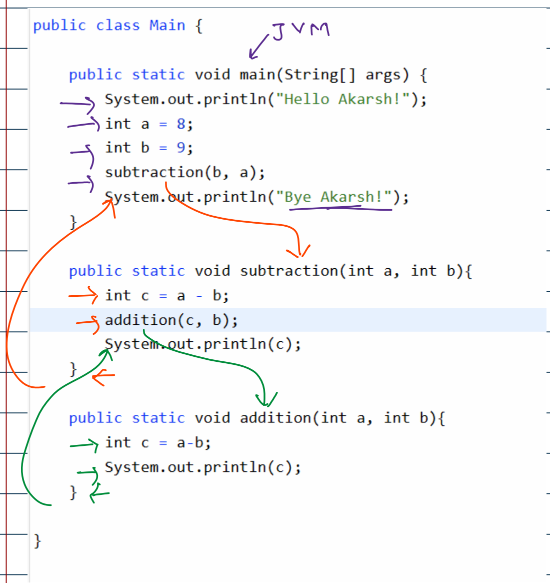


Hello Akarsh
-1
Bye Akarsh

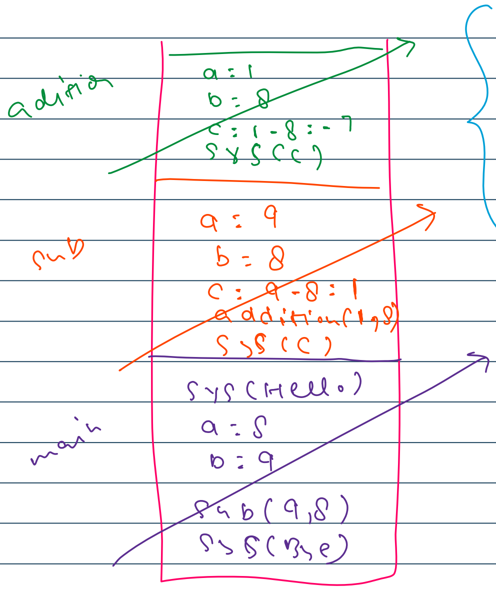


```
    }

    public static void subtraction(int a, int b){
        int c = a - b;
        System.out.println(c);
    }

}
```





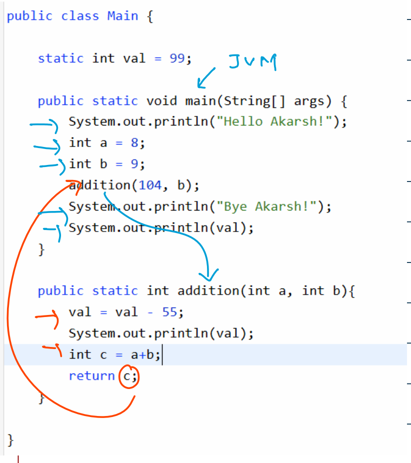
```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        subtraction(b, a);
        System.out.println("Bye Akarsh!");
    }

    public static void subtraction(int a, int b){
        int c = a - b;
        addition(c, b);
        System.out.println(c);
    }

    public static void addition(int a, int b){
        int c = a-b;
        System.out.println(c);
    }

}
```



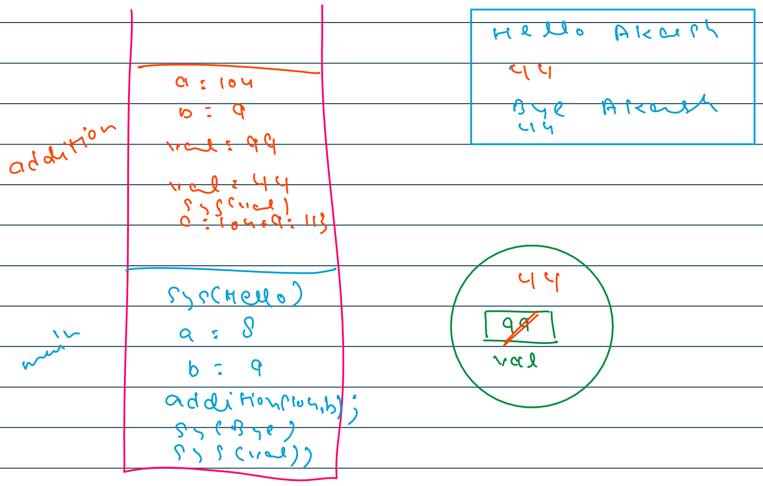
Hello Akarsh
-7
1
Bye Akarsh

A f^n can return only one value.
{single return type}

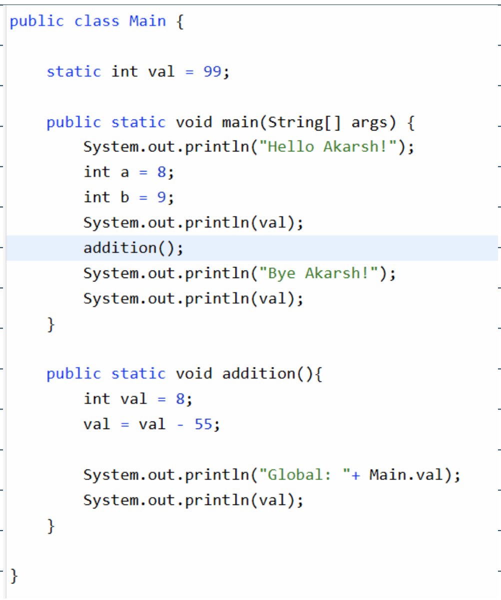
```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        // System.out.println(addition(a, b));
        int cc = addition(a, b);
        System.out.println("Printing sum of two numbers...");
        System.out.println(cc);
        System.out.println("Bye Akarsh!");
    }

    public static int addition(int a, int b){
        int c = a+b;
        return c;
    }

}


public class Main {

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        int cc = addition(104, b);
        System.out.println(cc);
        System.out.println("Bye Akarsh!");
    }

    public static int addition(int a, int b){
        int c = a+b;
        return c;
    }

}
```



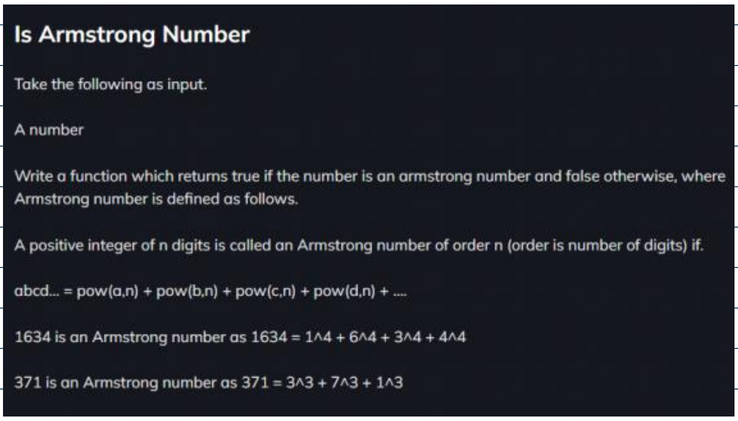
```
public class Main {

    static int val = 99;

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        addition(104, b);
        System.out.println("Bye Akarsh!");
        System.out.println(val);
    }

    public static int addition(int a, int b){
        val = val - 55;
        System.out.println(val);
        int c = a+b;
        return c;
    }

}
```



```
Hello Akarsh
44
Bye Akarsh
44
```



```java
public class Main {

    static int val = 99;

    public static void main(String[] args) {
        System.out.println("Hello Akarsh!");
        int a = 8;
        int b = 9;
        System.out.println(val);
        addition();
        System.out.println("Bye Akarsh!");
        System.out.println(val);
    }

    public static void addition(){
        int val = 8;
        val = val - 55;

        System.out.println("Global: "+ Main.val);
        System.out.println(val);
    }

}
```



Is Armstrong Number

Take the following as input.

A number

Write a function which returns true if the number is an armstrong number and false otherwise, where Armstrong number is defined as follows.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ...

1634 is an Armstrong number as 1634 = 1^4 + 6^4 + 3^4 + 4^4

371 is an Armstrong number as 371 = 3^3 + 7^3 + 1^3

$1\ 6\ 3\ 4 \rightarrow \boxed{4 \text{ digits}}$

$\Downarrow$

$1^4 + 6^4 + 3^4 + 4^4$

$1000 + \quad + \quad +$

$\Downarrow$

1634

✓

$3\ 7\ 1 \rightarrow 3$ digits

$3^3 + 7^3 + 1^3$

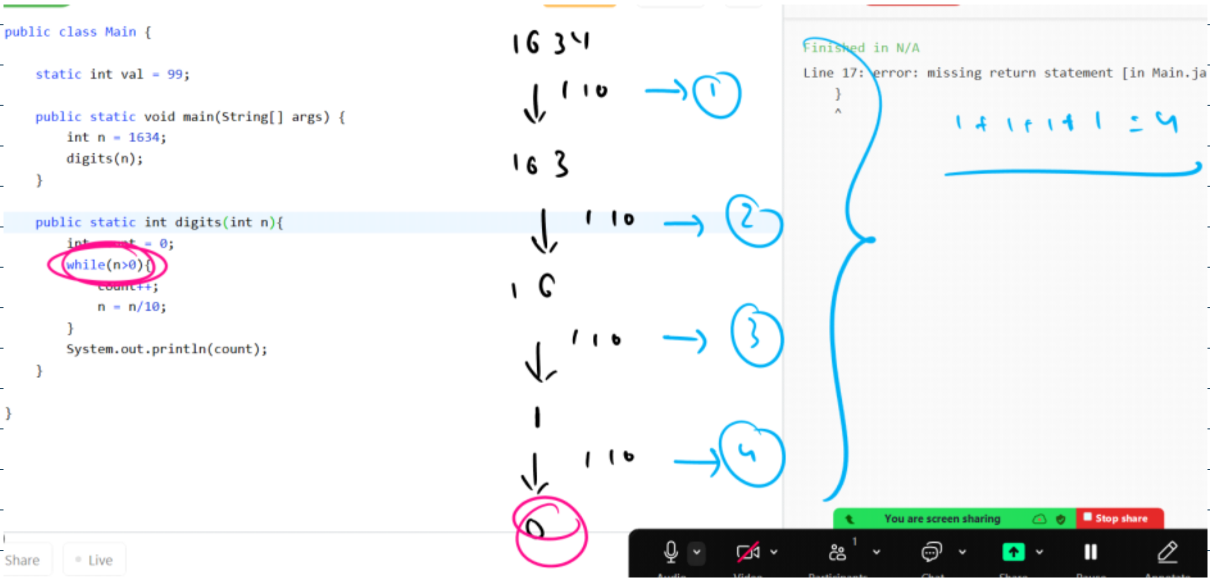
$27 + 343 + 1 = 371$

✓

$2\ 5 \rightarrow 2$ digits

$2^2 + 5^2 = 4 + 25 = 29$

✗



```
public class Main {

    static int val = 99;

    public static void main(String[] args) {
        int n = 1634;
        digits(n);
    }

    public static int digits(int n){
        int count = 0;
        while(n>0){
            count++;
            n = n/10;
        }
        System.out.println(count);
    }

}
```

```
Finished in N/A
Line 17: error: missing return statement [in Main.ja
    }
    ^
```