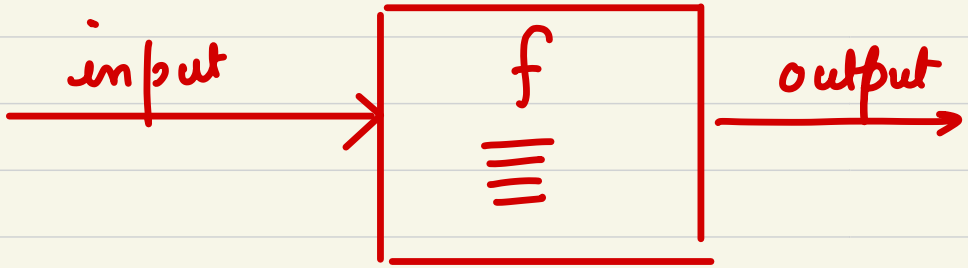
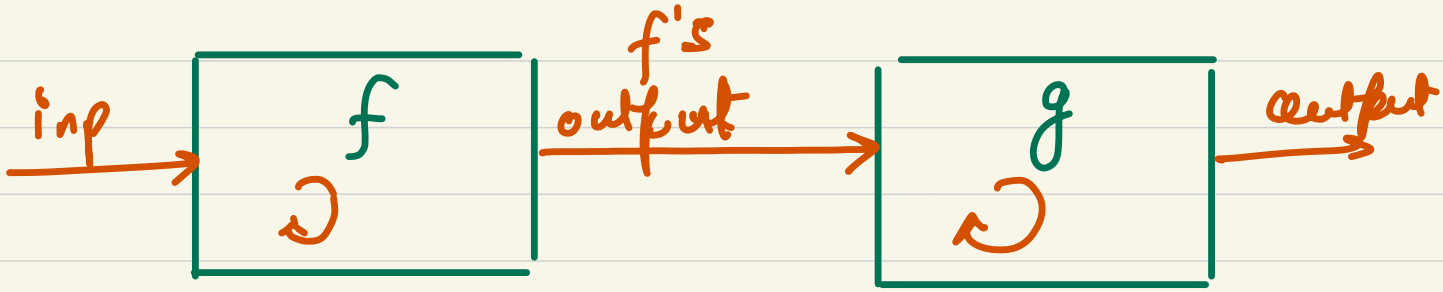


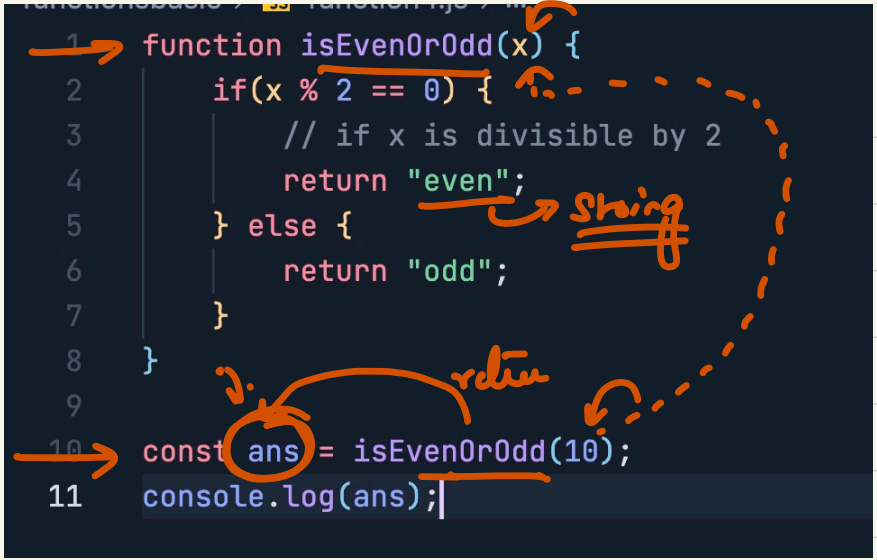
Q. how is printing a value diff than giving an output?

1) When you print a value you cannot use it for some operation



to generate output in JS functions we can use return keyword.

the moment you hit return the func^n stops the execution.

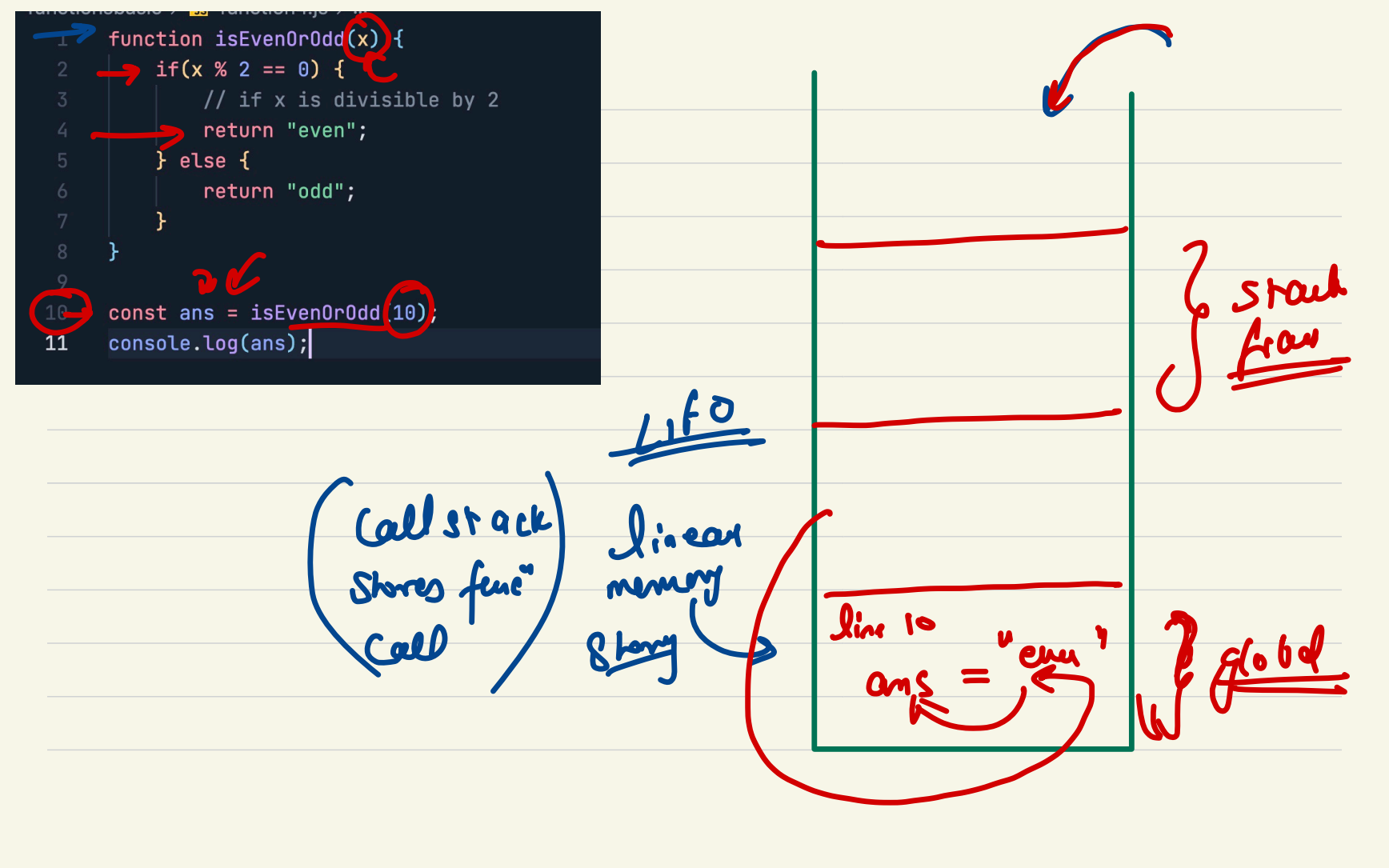


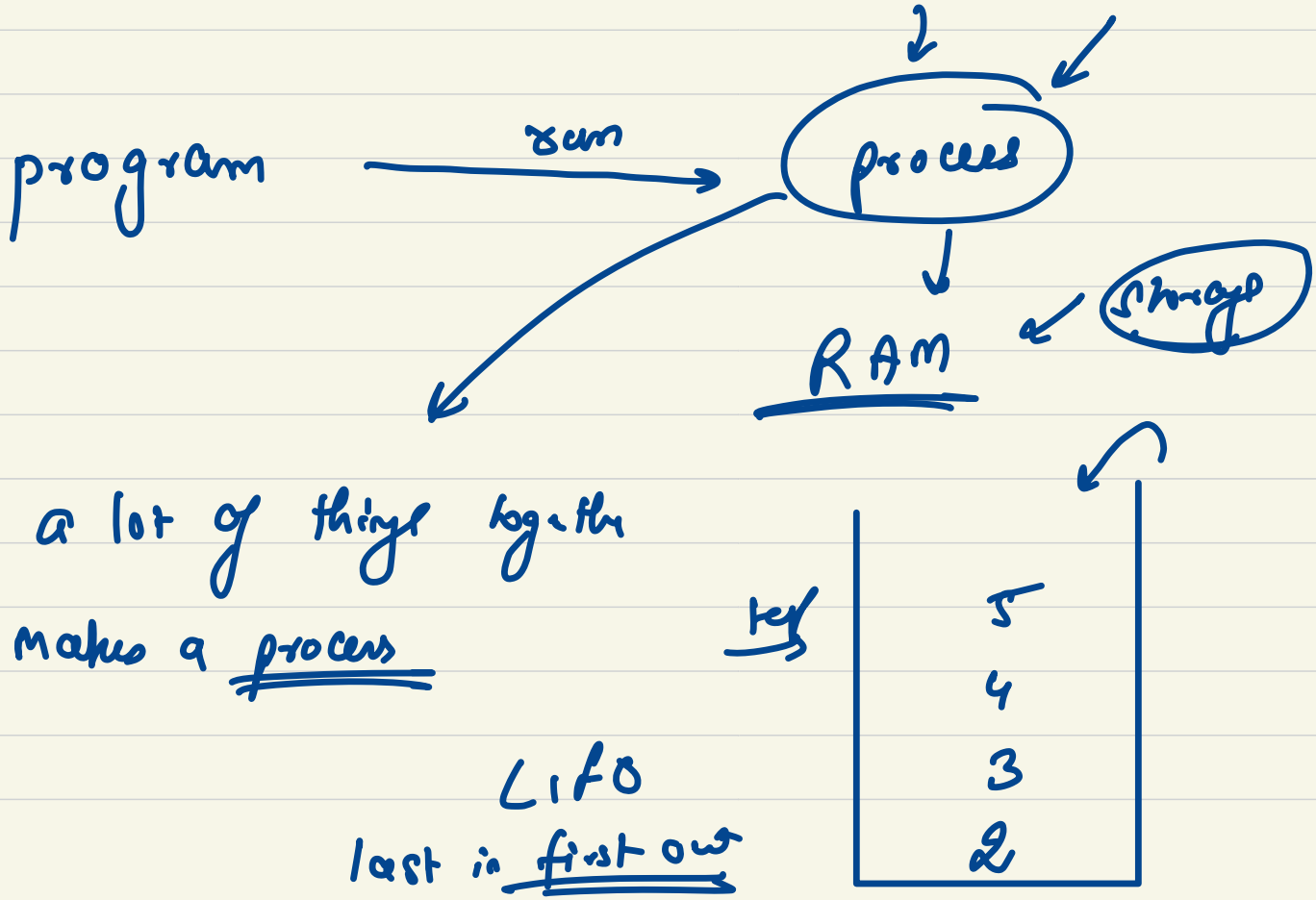
the moment JS reads function keyword, it understands that we want to define a function.

the moment we hit return statement func exec immediately stops.

```
function isEvenOrOdd(x) {
    if(x % 2 == 0) {
        // if x is divisible by 2
        return "even";
    } else {
        return "odd";
    }
}

const ans = isEvenOrOdd(10);
console.log(ans);
```




program
scan
process
RAM
Storage
a lot of things together
make a process
key
5
4
3
2
LIFO
last in first out

There are 3 main storage areas in a process →

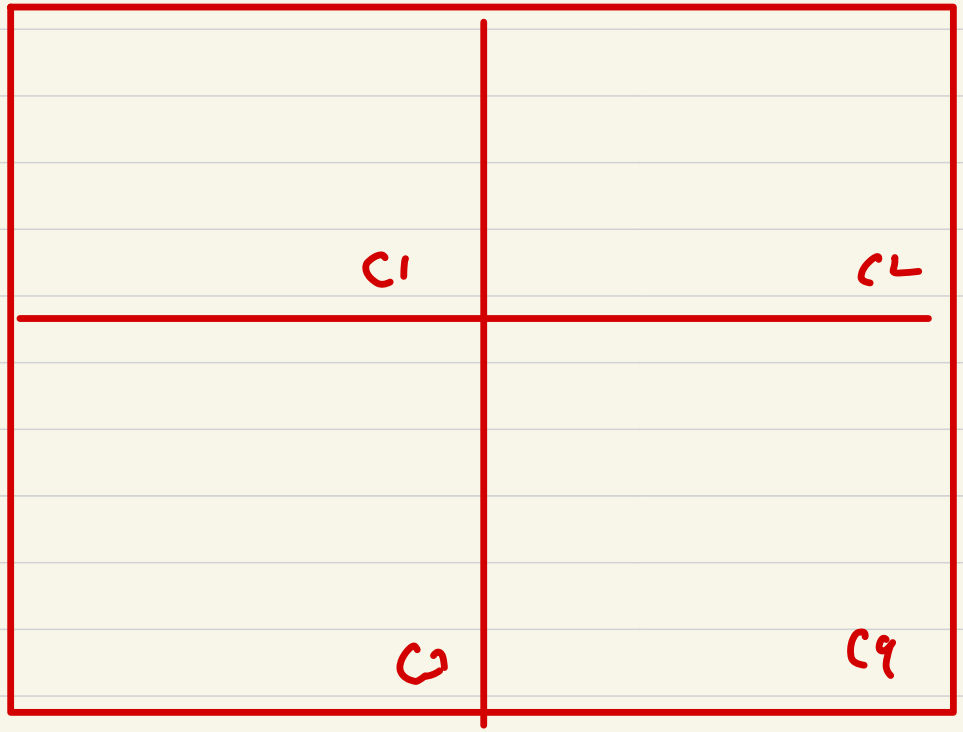
1) Call Stack → func"

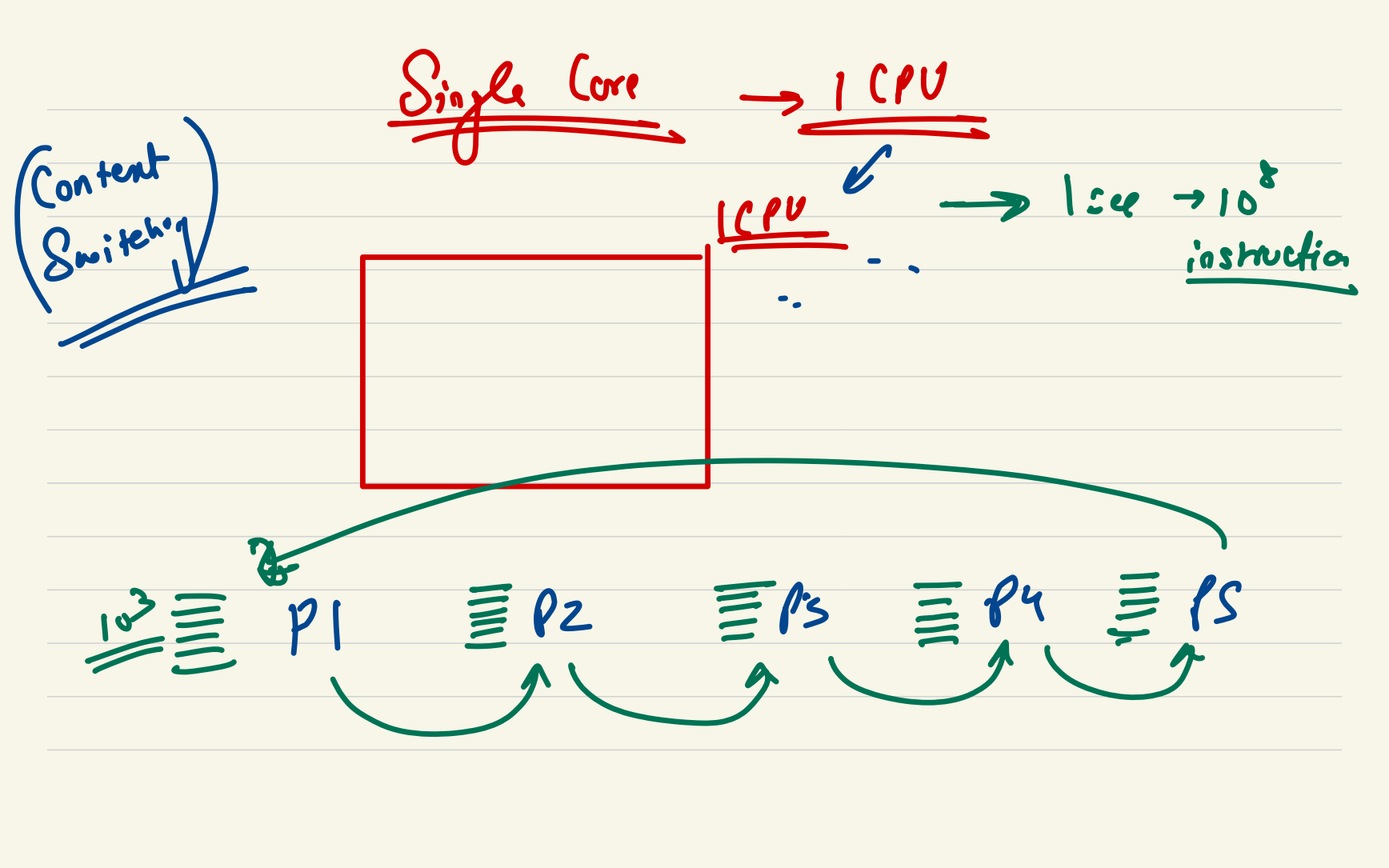
2) Kernel Stack ←

3) Heap Memory ← object

(Them on some storage area)

Quad tree




Single Core
1 CPU
Context Switching
1 CPU
1 sec → 10^8 instruction
10^3
P1
P2
P3
P4
P5

at any point of time in your code if you call a func^n a new entry will be added in the call stack.

The moment you hit return it removes the entry from call stack.

every entry added in the call stack by the func° call is called as Stack frame

(Note) if we donot return anything manually from the func° then JS will automatically return undefined.

in functions we can define default parameter value. if we do not manually pass anything to the parameter it will automatically pick the default value or if we passed something then it will take the passed value.

In JS there are a lot of internal in built func^n

Math → object

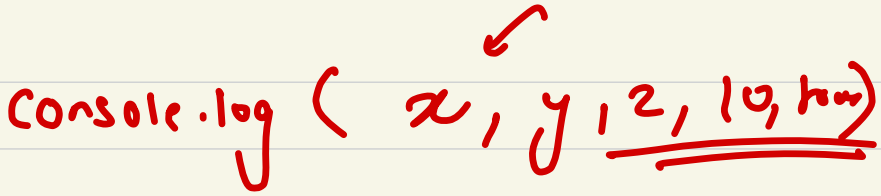
{

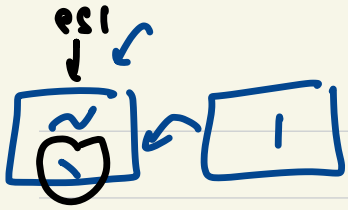
abs : f()

min : f()

⋮

}


Console.log (x, y, z, 10, hw)



text → shift

backtick

→ " abcd..... " → double quote

→ ' ~~~~~ ' → single quote

→ ` ~~~~~ ` → backtick

! Rajo said "where are you going?" !

' I don't know may '

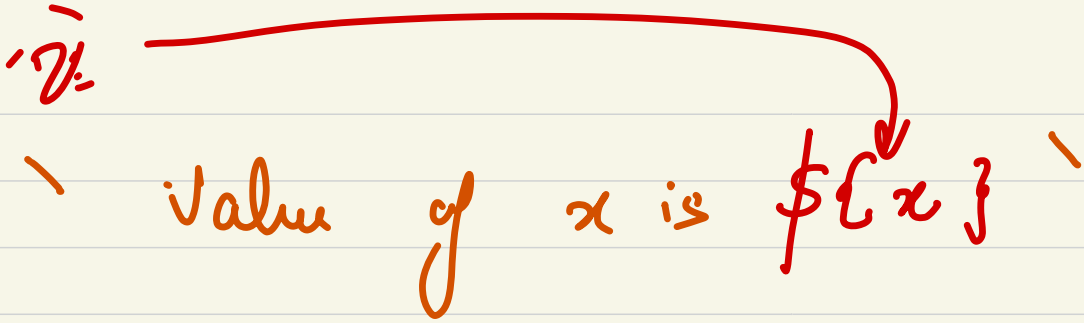
String interpolation / templated string

x = 100

11

12

S = Value of x is 100


"Value of x is ${x}"