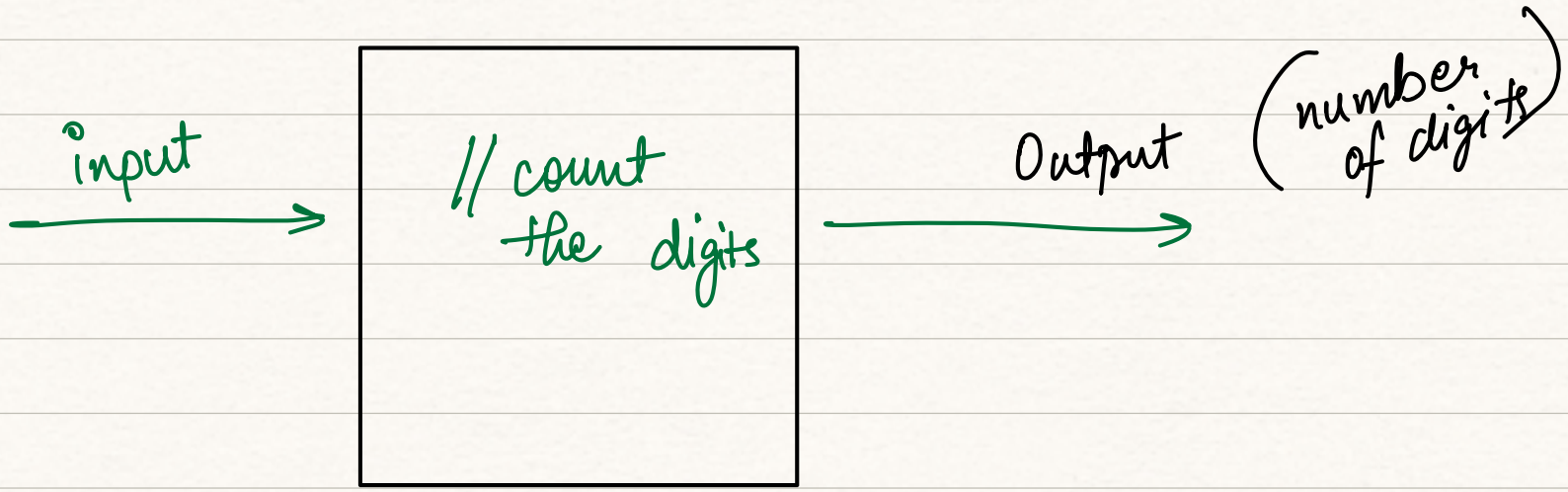
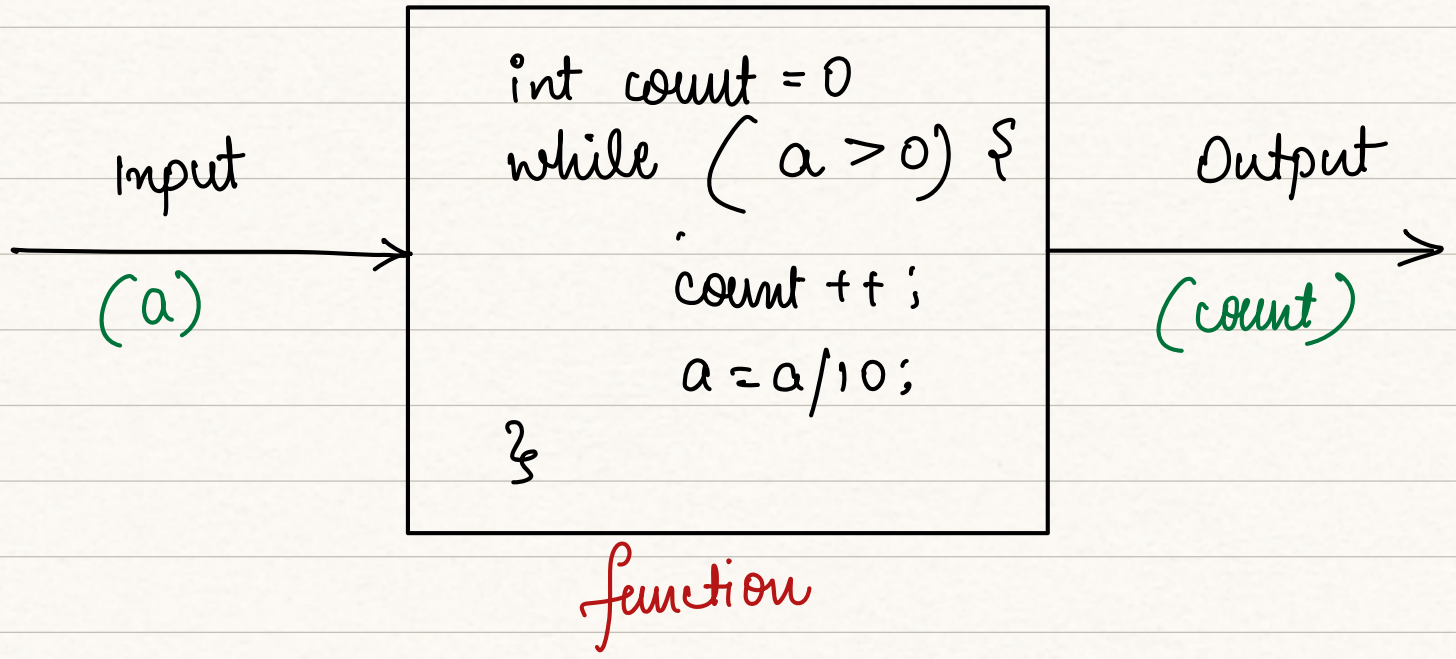
functions

count digits of a number

```
main () {
    Scanner scn = new Scanner (System.in);
    int a = scn.nextInt ();
// count digits of a
    int count = 0;
    while (a>0) {
        count++;
        a= a/10;
    }

int b = scn.nextInt ();
int count2 = 0;
 while (b>0) {
        count2++;
        b = b/10;
}
}
```

Redundancy
more lines
Human error
Difficult to maintain


input
// count the digits
Output
(number of digits)


Input
(a)
int count = 0
while (a > 0) {
count ++ ;
a = a/10;
}
Output
(count)
function

Template

```
(output)
return data-type   function-name (data-type input) {

    // logic to perform

    return ans;

}
```

```
int   count-digits ( int a ) {

    int count = 0
    while ( a > 0) {

        count ++ ;

        a = a/10;

    }

return count;

}
```

a = 194632

count = 6

```
main()  {
    int n = scn.nextInt();
    int ans = count_digits(1416);   // call the function
    S.O.P(ans);
}
```

4

4

```
int x = count_digits(194632);
x = x+1;   // x = 7
S.O.P(x);
```

6

6

X

7

void → return type of function that does not return anything

```
static void sum (int a, int b) {
    S.O.Pln (a+b);
}
```

```
static void sum (int a, int b) {
    S.O.Pln (a+b);
    return;
}
```

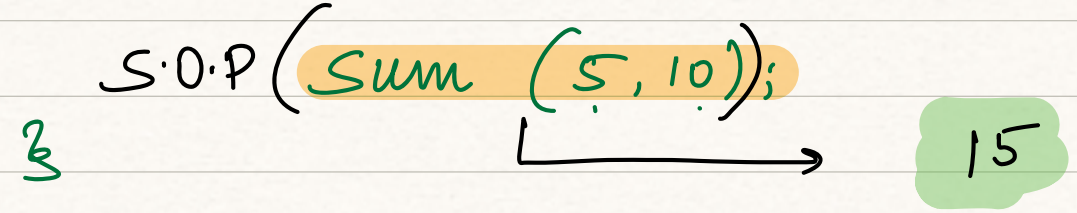
Both are correct ways

```
static int sum (int a , int b) {
        //a = 5 , b = 10
    return a+b;     //  15
}

main ( ) {

    sum (5, 10);
}
```

Ans → None

Since there is no print statement

```
static int sum (int a , int b ) {
    // a = 5 , b = 10
    return a +b;    //  15
}

main ( ) {
    S.O.P( sum (5, 10));
}
```



```
static void sum (int a, int b) {
    return a+b;
}
main ( ) {
    sum (5, 10) ;
}
```

5, 10

// Error

return type is void. So, cannot return

```
static int sum (int a, int b) {
    return a+b+10;
}
main ( ) {
    int ans = sum (5, 10);
    S.O.P (ans);
}
```

5, 10

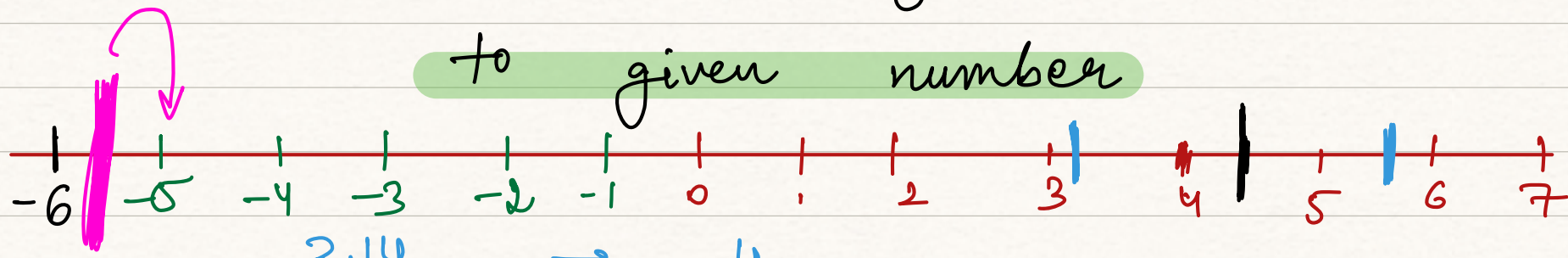
// 5+10+10 → 25

25

25 → 25

ans

ceil → first integer greater than or equal to given number



3.14 → 4

5.67 → 6

2.001 → 3

9.0 → 9

−5.67 → −5

4.4 → 5

0.598 → 1

floor → first integer smaller than or equal to given number

3.14 → 3

5.67 → 5

2.001 → 2

9.0 → 9

0.598 → 0

−5.67 → −6

double — return type

Math.ceil (5.67) ; → 6.0

Math.floor (5.67) ; → 5.0

Math.pow (2, 4) ; → 16.0

(2^4)

can easily typecast to int.