Iteration / Loop

Problem: Find total weights of all pumpkins.

```
int weight[] = { 6, 3, 2, 5 };
int total = 0;

total = weight[0] + weight[1] + weight[2] + weight[3];

System.out.println("Total = " + total);
```

Output:
16

```
int weight[] = { 6, 3, 2, 5, 6, 8, 7, 1 };
int total = 0;

for (int i=0; i<weight.length; i++) {
   total = total + weight[i];
}

System.out.println( "Total = " + total );
```

Output:
38

Iteration / Loop

- There are 3 types of loops in Java:
  - while
  - do - while
  - for

while

```
while (condition) {

    // your code here

}
```

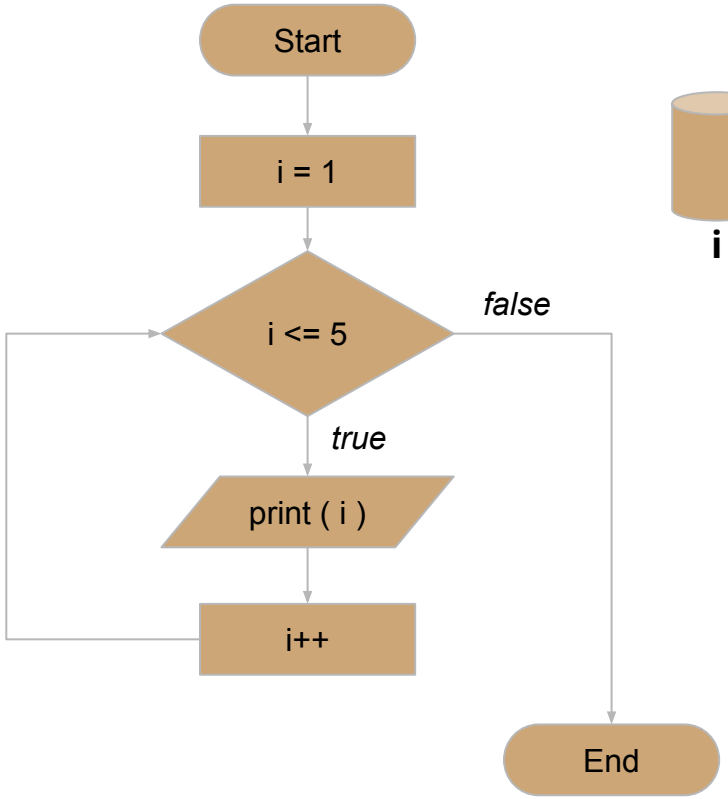
- *This loop **repeats** statements inside it, as long as the condition is **true**.*
- *When the condition is **false**, the program execution **exits** from the loop.*
- If the condition is always ***true***, the loop will never stop and will make an **infinite loop**.
- If the condition is always ***false***, it will never enter into the loop.

```
int i = 1;

while (i <= 5) {
    System.out.println( i );
    i++;
}
```

Output:
1
2
3
4
5

while: Flowchart




```
int i = 1;

while (i <= 5) {
     System.out.println( i );
     i++;
}
```

Output:
1
2
3
4
5

do-while

```
do {

        // your code here

} while (condition);
```

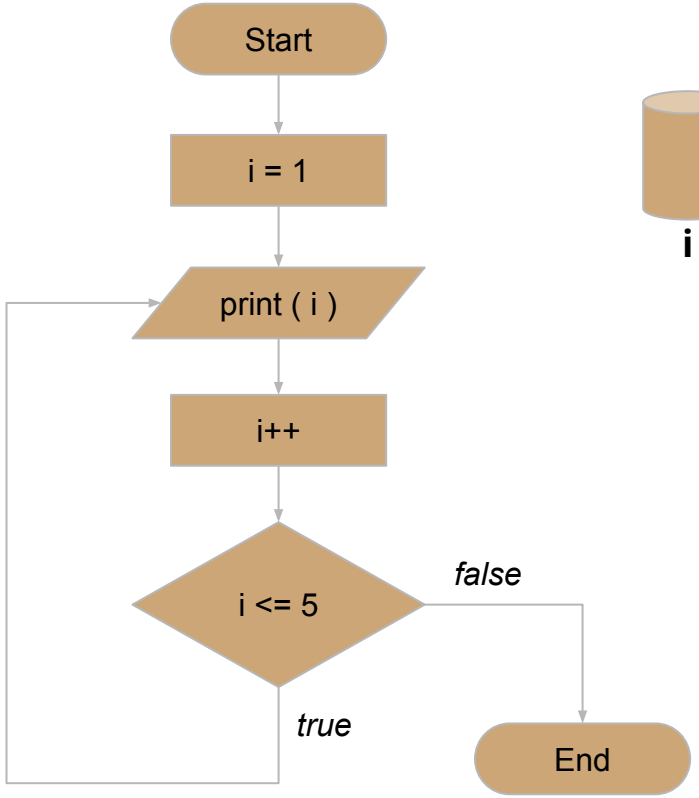
- *This loop **executes** the statements inside it **first** and then check the condition.*
- *If initial condition is **false**, the statements will be executed at least **once**.*
- *Add a semicolon after while condition.*

```
int i = 1;

do {
      System.out.println( i );
      i++;
} while (i <= 5);
```

Output:
1
2
3
4
5

do-while: Flowchart




```
int i = 1;

do {
     System.out.println( i );
     i++;
} while (i <= 5);
```

Output:
1
2
3
4
5

do

```
int i = 10;

while (i <= 5) {
     System.out.println( i );
     i++;
}
```

Output:

do-while

```
int i = 10;

do {
     System.out.println( i );
     i++;
} while (i <= 5);
```

Output:

10

while

```
int i = 1;

while (i <= 5) {
     System.out.println( i );
     i++;
}
```

Output:
1
2
3
4
5

for

```
for (int i = 1; i <= 5; i++) {
     System.out.println( i );
}
```

```
for (initialization; condition; iteration) {

     // your code here

}
```

for

Problem: Print numbers in reverse order, 5 to 1.

```
for (int i = 5; i >= 1; i--) {
     System.out.println( i );
}
```

Output:
5
4
3
2
1

for

Problem: Add all numbers of an array

```
int[] nums = {11, 4, 18, 73, 65};
int total = 0;

for (int i = 0; i < nums.length; i++) {
     total += nums[i];
}

System.out.println( total );
```

Output:

171

for: Flowchart

Problem: Add all numbers of an array

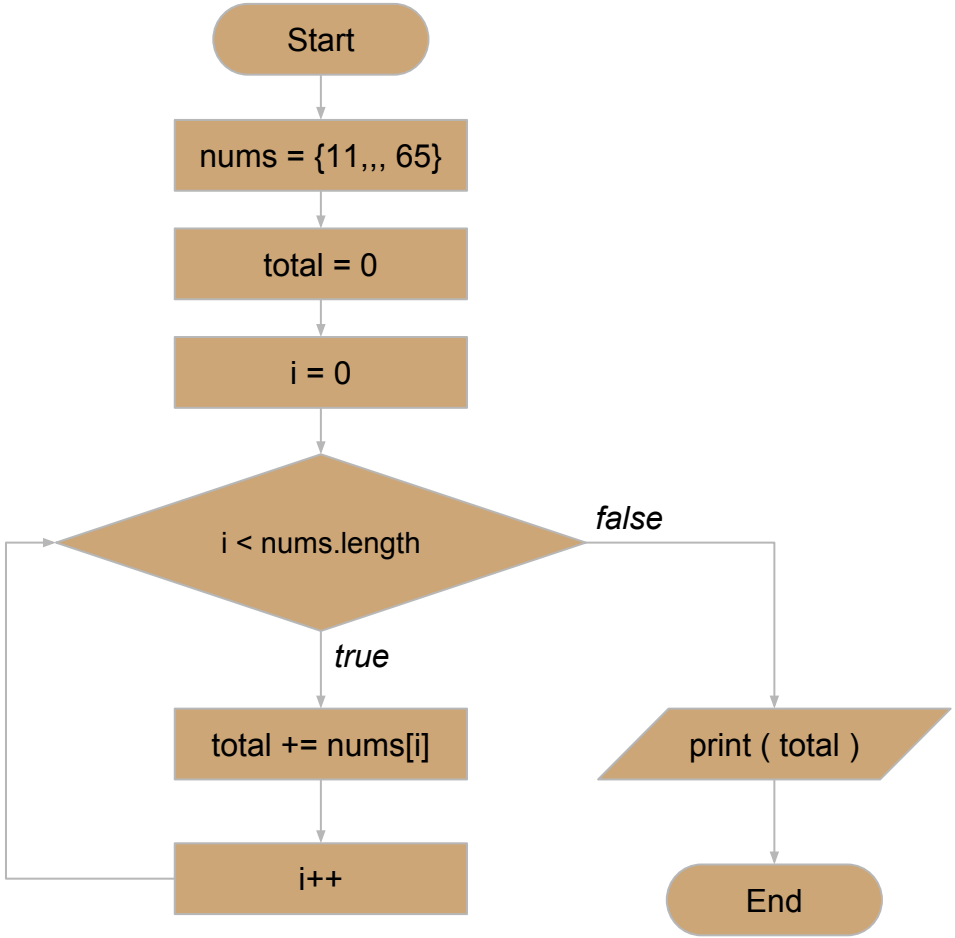
```
int[] nums = {11, 4, 18, 73, 65};
int total = 0;

for (int i = 0; i < nums.length; i++) {
     total += nums[i];
}

System.out.println( total );
```

Output:

171



for

Problem: Add together all numbers of an array.

```
int[] nums = {11, 4, 18, 73, 65};
int total = 0;

for (int i = 0; i < nums.length; i++) {
     total += nums[i];
}

System.out.println( total );
```

Output:

171

for-each style

(enhanced for loop)

```
int[] nums = {11, 4, 18, 73, 65};
int total = 0;

for (int n: nums) {
     total += n;
}

System.out.println( total );
```

Output:

171

Nested Loop

Problem: Print a character star **(*)** in a pattern where ***1st*** line has only one star, ***2nd*** line has 2 starts and ***nth*** line has **'n'** numbers of stars.

```
int n = 5;

for (int i = 0; i < n; i++) {

     for (int j = 0; j <= i; j++) {
          System.out.print("* ");
     }

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

Input:

n = 5

Output:

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

Nested Loop

Problem: Print a character star **(*)** in a pattern where ***1st*** line has only one star, ***2nd*** line has 2 starts and ***nth*** line has **'n'** numbers of stars.

```
int n = 5;

for (int i = 0; i < n; i++) {

     for (int j = 0; j < n; j++) {
          System.out.print("* ");
     }

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

Input:

n = 5

Output:

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