Nested Loops

Student task 1: Link

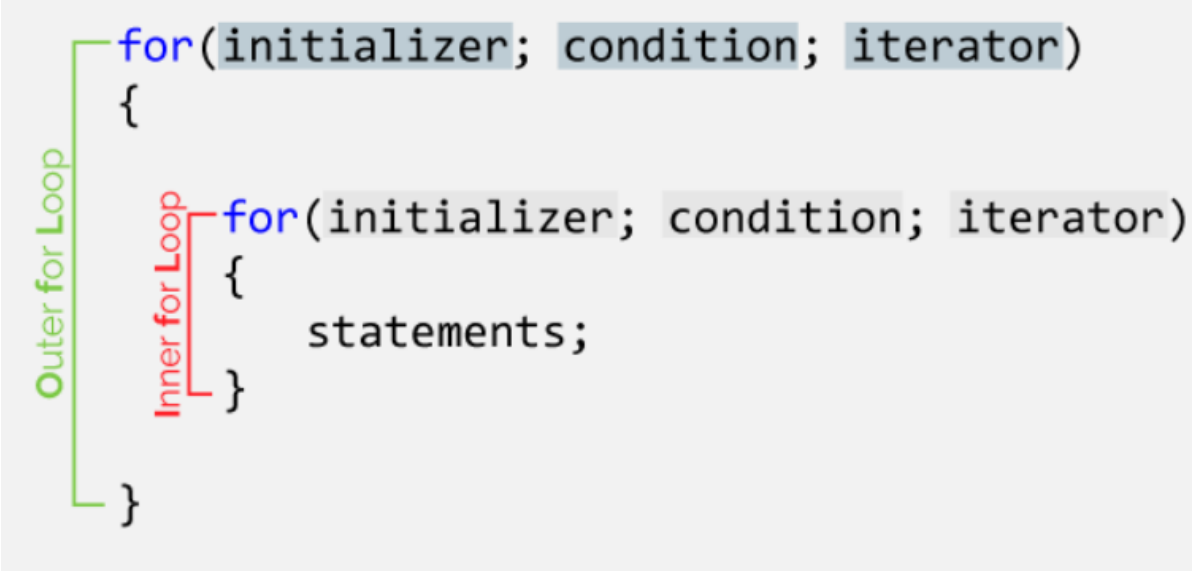
```
// you have to print "Hello" for the 5 times in console.
// use for loop to print it.
```

Nested Loop

- A nested loop is a loop within a loop.
- Let's try to understand with the analogy
    - Suppose you went to a Gol Gappa Shop and you ate 10 gol-gappas in a sequence. You can consider this as a loop where you have eaten 10 gol gappas in a sequence.
    - In another scenario, Suppose you went to a Gol Gappas Shop with the 5 other family members. Each Member of the family ate 10 gol-gappas. You

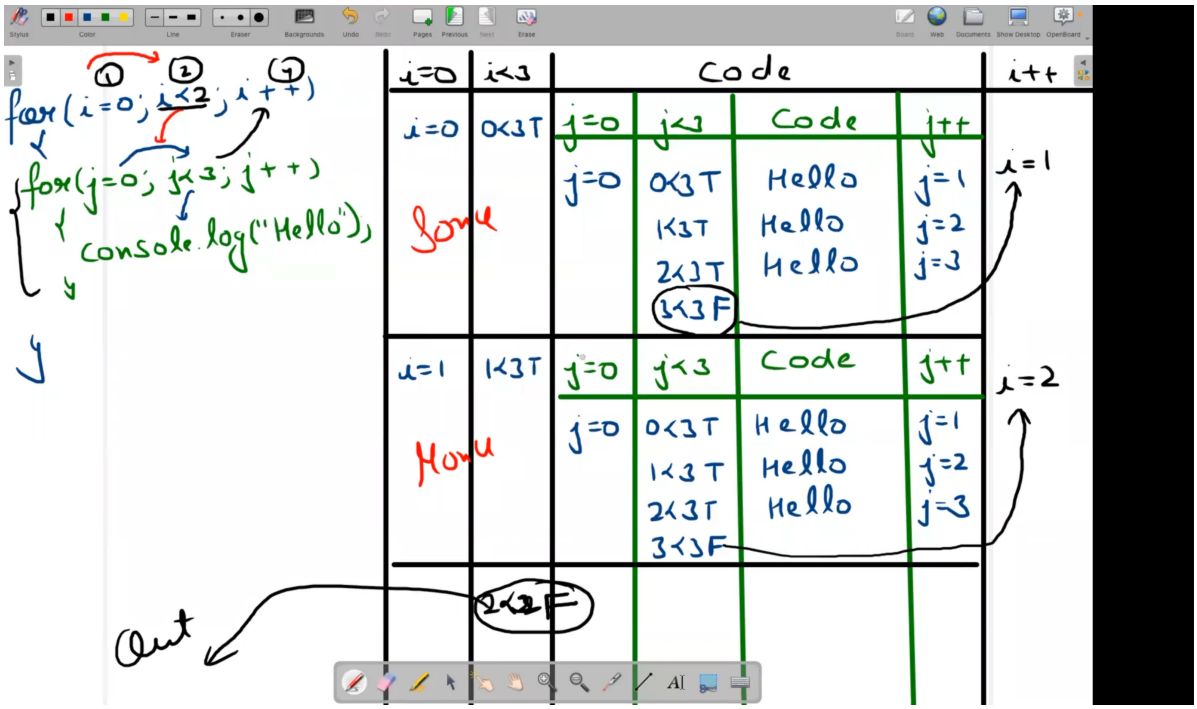
can consider this as a loop of 10 gol-gappas which run 5 times because of the 5 family members.

Syntax



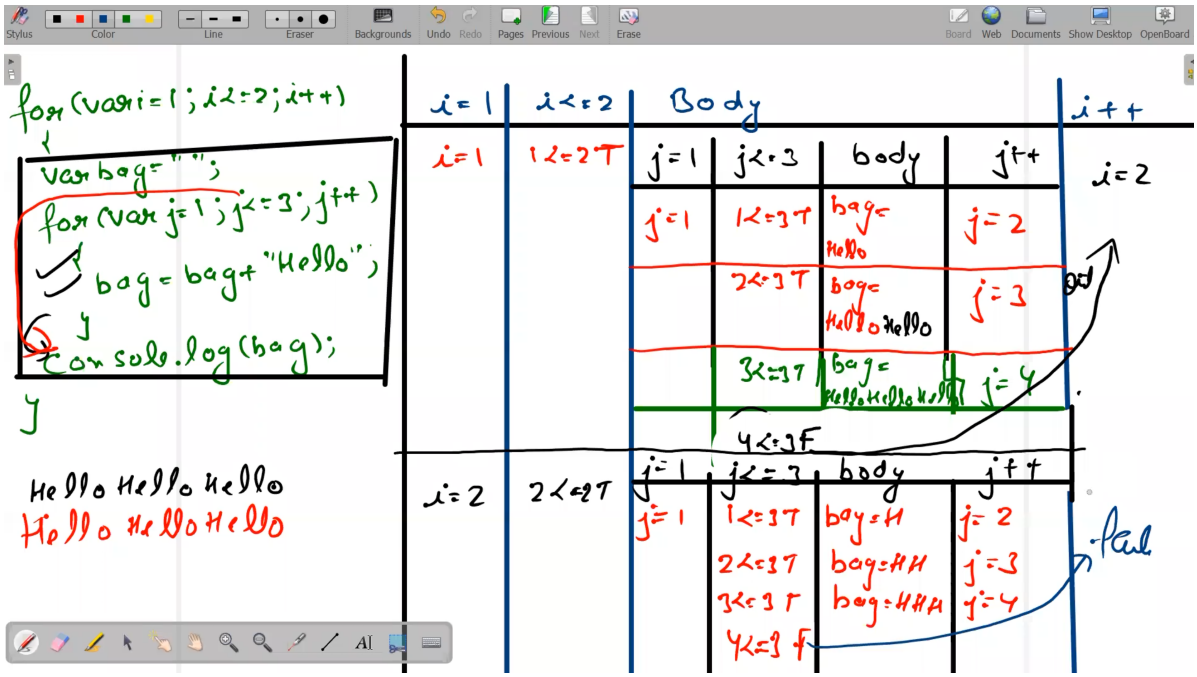
Dry Run of problem

Code 1 : print Hello in vertical order using nested loop.




```
for(var i=0;i<2;i++)
{
for(var j=0; j<3; j++){
    console.log("Hello");
  }
}
```

Code 2 : print Hello in horizontal order using nested loop




```
for(var i=1;i<=2;i++)
{
  var bag="";
  for(var j=1; j<=3; j++){
    bag = bag + "Hello ";
  }
  console.log(bag);
}
```

Father-Son Marathon

There was a father who trains his son for next olympics for running competition.

Every day Father use to give some target's to his son

Code 3 : Father gave the son a target , of completing 10 sets . Each set contains 10 Rounds of a field.

```
for(var father=1; father<=8; father++)
{
    for(var son = 1; son<=10; son++)
    {
      console.log("Father count",father,",Son completed ",
son);
    }
}
```

Father-Son planting trees

Once upon time, There was a farther and son who were farmers. They once decided that they do plantation of trees in their field. Since Father is very old, he unable to do that much work whereas son is pro-active, that's why father responsibility is to make sure how many fields are done whereas son has the responsibility of putting the seeds in the field.

Code 4 : Father has 5 fields . Each fields son needs to put 10 seeds

```
// * * * * *
// * * * * *
// * * * * *
// * * * * *
// * * * * *

for(var father=1; father<=5; father++)
{
  var bag = "";
  for(var son=1; son<=10; son++)
  {
    bag = bag + "*";
  }
  console.log("Field",father,bag);
}
```

Code 5 : Father has 5 fields. Son needs to put the seeds in increasing order.

Field 1 → 1 seed

Field 2 → 2 seed

Field 3 → 3 seed

Field 4 → 4 seed

Field 5 → 5 seed

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

```
for(var father=1; father<=5; father++)
{
```

```
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + "* ";
  }
  console.log(bag);
}
```

Code 6 : Print the below pattern

Print the below pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
for(var father = 1; father<=5; father++)
{
  var bag = "";
  for(var son = 1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Code 7 : Father has 5 fields. Son needs to put the seeds in decreasing order.

Field 1 → 5 seed

Field 2 → 4 seed

Field 3 → 3 seed

Field 4 → 2 seed

Field 5 → 1 seed

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

```
or(var father=5; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag +"* ";
  }
  console.log(bag);
}
```

Code 8 : Print the below reverse pattern

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
for(var father=5; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Code 9 : Combining Code 6 and Code 8 form a pyramid.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
for(var father = 1; father<=5; father++)
{
  var bag = "";
  for(var son = 1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

```
for(var father=4; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Nested Loop with While

Code 10 : Use While Loop. Print the below pattern

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

var father=1;
while(father<=6)
{

  var son=1;
  var bag = "";
  while(son<=10)
  {
    bag=bag+"*";
    son++;
  }
  console.log(bag);
```

```
  father++;
}
```

Break and Continue

Break

Sultan and Bahubali were the good friends , but Sultan was the king whereas Bahubali does n't have any kingdom. Later Sultan gave one part of his empire to Bahubali with a condition that Bahubali will never try to cross the status of Sultan and if he tries that then he will attack on the Prithviraj clan.

Code 11 : Using Break

```
for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi>sultan)
    {
      break;
    }

    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}
```

In the above code, whenever the value of Bahubali become greater than Sultan, At that point the inner loop of bahuballi will break [It means sultan attacked on his clan because Bahubali betray him, by not following his conditions]

Continue

The below code gives the same output as the above code but the only difference is on break is that the execution will stop completely but in case of continue the process will keep on skipping the step and execution will end only once the loop will done.

Code 12 : Using Continue

```
for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi>sultan)
    {
      continue;
    }

    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}
```

Code 13 : Break vs Continue . Predict the Output

```
**BREAK**

for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi == sultan)
    {
      break;
    }
```

```
    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}

**CONTINUE**

for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {
    if(bahuballi == sultan)
    {
      continue;
    }
    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}
```

Code 14 : Print the below pattern

```
1
*
1 2
* *
1 2 3
* * *
1 2 3 4
* * * *
1 2 3 4 5
* * * * *

for(var father=1; father<=5; father++)
{

  var bag1 = "";
```

```
  for(var son=1; son<=father;son++)
  {
    bag1 = bag1+son+" ";
  }
  console.log(bag1)

  var bag2 = "";
  for(var son2=1; son2<=father;son2++)
  {
    bag2 = bag2+"*"+" ";
  }
  console.log(bag2)
}
```

IW Assignment

Code 15 : Print the Calendar date

```
Problem 1: Print the Calendar date in the below format
1-1
2-1
3-1
.
.
.
.
.
31-1
.
.
.
.
.
31-12
```

```
for(var month=1; month<=12 ; month++)
{

  var days = 31;

  if(month==2)
  {
    days = 28;
  }
  else if(month==4 || month==6 || month==9|| month==11)
  {
    days = 30;
  }

  switch(month)
  {
    case 1:
            console.log("January");
            break;

    case 2:
            console.log("February");
            break;

    case 3:
            console.log("March");
            break;
  }

  for(var day=1; day<=days; day++)
  {
    console.log(day,"-",month);
  }
```

```
}
```

Code 16 : Print Prime Numbers from 1 to given limit

```
Problem 2: Print Prime Numbers from 1 to given limit

var limit = 1000;

for(var number=1; number<=limit; number++)
{

      var factors=0;

      for(var i = 1; i<=number; i++)
      {
        if(number%i==0)
        {
          factors++;
        }
      }

      if(factors == 2)
      {
        console.log(number,"is a prime number");
      }
      else
      {
        console.log(number,"is not a prime number");
      }

}
```

Code 17 : Print a box pattern

```
Problem 3: Print a box pattern using *

**********
*        *
*        *
*        *
*        *
*        *
*        *
*        *
*        *
**********

for(var i = 1; i<=10; i++)
{
  var bag="";
  for(var j=1; j<=10; j++)
  {
    if(i==1 || i==10)
    {
      bag = bag + "*";
    }
    else
    {
      if(j==1 || j==10 )
      {
        bag = bag+"*";
      }
      else
      {
        bag = bag+" ";
      }
    }

  }
```

```
  console.log(bag);
}
```