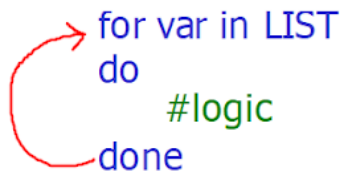


## 4. Shell Scripting – loops

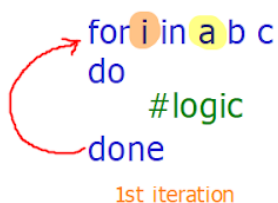
Loops are used to repeatedly execute a set of instructions for a number of times. The number of times for loop runs completely depends on the number of items in the LIST as per the bash scripting syntax.

**for loop:**



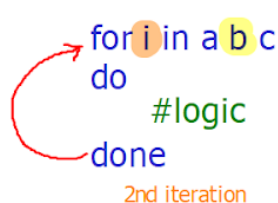
```
for var in LIST
do
    #logic
done
```

Here, var is the iterative variable. Multiple elements can be in place of LIST like list or array. Whenever the loop runs, iterative variable will be assigned with one item in the list automatically. The loop will be terminated only when all the items in the LIST are assigned to the iterative variable.



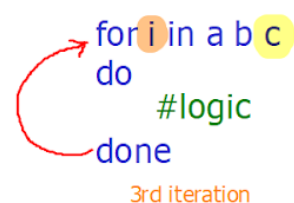
```
for i in a b c
do
    #logic
done
```

1st iteration



```
for i in a b c
do
    #logic
done
```

2nd iteration



```
for i in a b c
do
    #logic
done
```

3rd iteration

In the above example LIST of items are a, b and c. So, the loop runs for 3 times.

For 1<sup>st</sup> iteration, the value of iterative variable i = a

For 2<sup>nd</sup> iteration, the value of iterative variable i = b

For 3<sup>rd</sup> iteration, the value of iterative variable i = c

The for loop can be executed for a number of times. For that, exact number of items should be there in place of LIST in for loop. To run a for loop exactly for a particular number of times usually **seq** command is used. **seq** command will generate sequence of numbers. Some examples of seq command usage are given below:

**seq 10** → generate sequence from 1 to 10

**seq -2 10** → generate sequence from -2 to 10

**seq 3 1 10** → generate sequence from 3 to 10 with difference between adjacent terms in sequence as 1

**seq 2 2 15** → generate sequence of alternate numbers from 2 to 15

**seq 10 -1 4 → generate descending order sequence from 10 to 4**

List of numbers can be also be generated using the following format:

**{starting\_number..ending\_number}**

### **Nested for loop:**

Nested for loop consists of one or more for loop inside another one depending on requirement like to create patterns or matrices.

```
for var1 in LIST1
do
    #logic
    for var2 in LIST2
    do
        #logic
    done
done
```

Here, var1 and var2 are iterative variables. Outer for loop will be terminated only after the termination of inner for loop. For every item in LIST1, inner for loop will run for the total number of items in LIST2.

### **While loop:**

while loop always run based on one condition. The logic inside while loop won't be executed unless the condition is true. The while loop won't be terminated until the condition becomes false.

```
while [ condition ]
do
    #logic
done
```

### **Implementation of do-while loop using while loop:**

Loop similar to do-while loop in c-programming can be implemented in bash scripting using while loop.

In the below script, entire logic has to be repeated based on user input and as per the condition given for the while loop. The user input is read into a variable choice and depending on its value the loop will behave from the second iteration. But, as

the logic comes inside the while loop, for the first execution of it, we have to make the condition of while true. For this purpose, the variable choice is initialized with 'y'. The while loop will enter the second iteration only if the choice is either 'y' or 'Y' as per the above execution.

```
choice=y
```

```
while [ $choice = y -o $choice = Y ]  
do  
    #logic  
    read -p "Do you want to continue?(y/n): " choice  
done
```

