2) for loop

to execute a set of commands for a certain no of times.

syntax

```
for var inlist
do
command 1
command 2
done
```

ex

```
for P_name in supriya gowda
do
echo $P_name
done
o/p :- supriya
       gowda
```

2) until loop

Syntax

```
until [conditional statement]do
command 1
command2
done
```

Ex:
```
number=1
until [ $number -gt 10]do
echo $number
((number++))
done
```

O/P

```
1
2
3
4
5
6
7
8
9
10
```

<u>if & else</u>

```
if [ expression 1 ]
then
   statement(s) to be executed if ex 1 is true.
elif [ ex 2 ]
then
   Statement (s) to be executed if ex2 is true.
elif [ex3]
then - - -                      - - ex3
~~then~~ else
statem (s)      - -    ? if no expression is true.
fi
```

ex

```
a=10
b=20

if [ $a == $b ]
then
 echo "a" is equal to b"
elif [$a -gt $b]
then
 echo "a" is greater than b"
elif [ $a -lt $b]
```

• while loop

Syntax:

```
while [condition] do
command 1
command 2
done
```

Ex:

```
number=1
 while [ $number -lt 11 ] do
  echo $number
  ((number++))
  Done
```

O/P

```
1
2
3
4
5
6
7
8
9
10
```

Continue statement

continue statement skips the remaning commands inside the body of the enclosing loop for the current iteration & passes program control to the next iteration of loop

```
i=0
while [[ $i -lt 5 ]]; do
  ((i++))
if [[ "$i" == '2' ]]; then
    continue
fi
echo "Number: $i"
done
echo 'All Done'
```

Output;

```
    Number : 1
    Number : 3
    Number : 4
    Number : 5
    All Done
```

```
then
echo " a is less than b"
else
echo "none of condition met"
fi
```
O/P
a is less than b

Break statement

it terminates the current loop and passes program control to the commands that follows the terminated loop.

```
i=0
while [[ $i -lt 5]]
do
  echo "Number : $i"
   ((i++))
if [[ $i -eq 2]] ; then
   break
 fi
done

echo 'All Done'
```

```
O/P :. Number : 0
               1
          All Done
```