UNIX/Linux Course

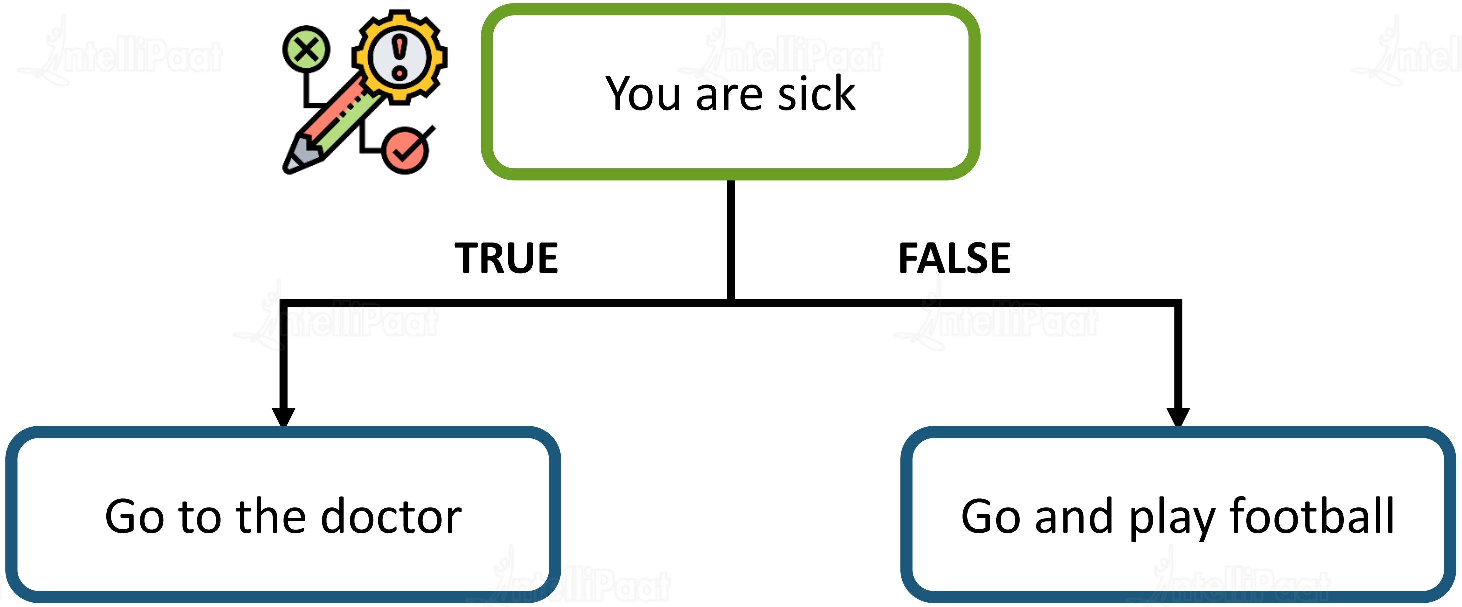
Conditional and Looping statements

Agenda

Conditional Statements

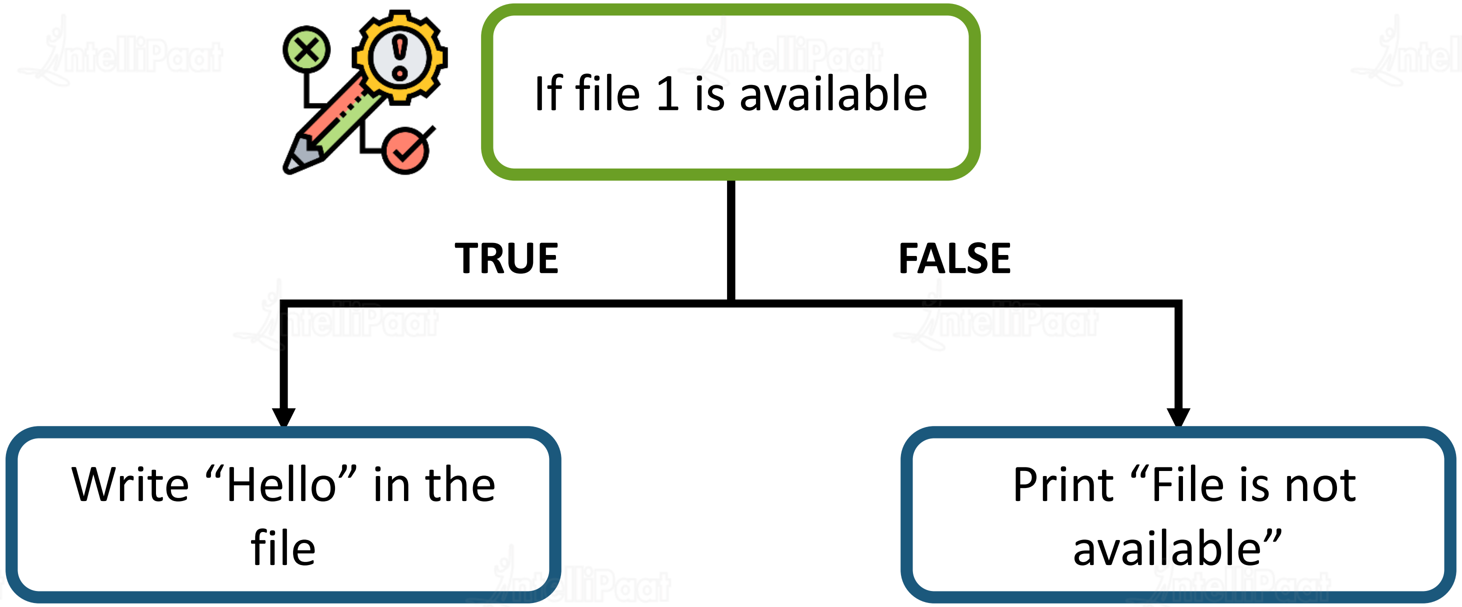
Conditional Statements

These statements are used to change the flow of execution when a provided condition is True or False



Conditional Statements

These statements are used to change the flow of execution when a provided condition is True or False



Using If, If-else, Else If ladder and Nested If statements

Using If Statement

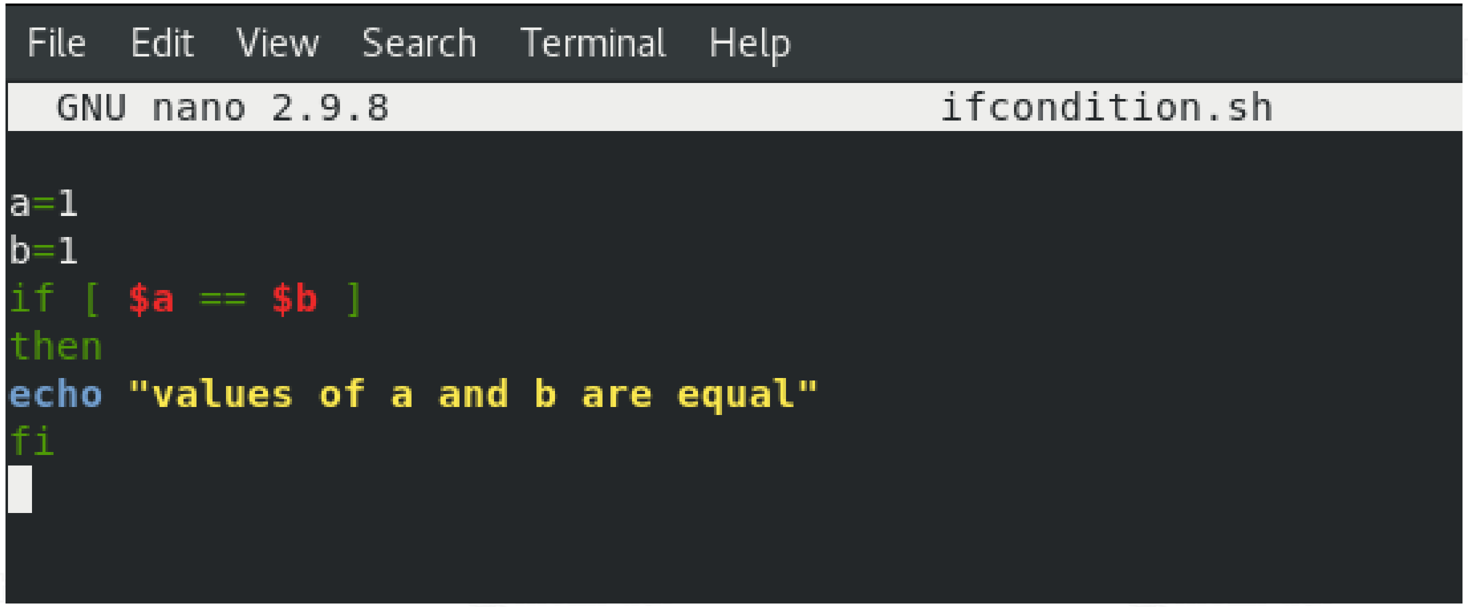
If a condition provided is true, it will do a certain set of actions and if false another set of actions

```
If [Condition]

then

      Statements

fi
```

The statements will execute if the specified condition is True

Using If Statement

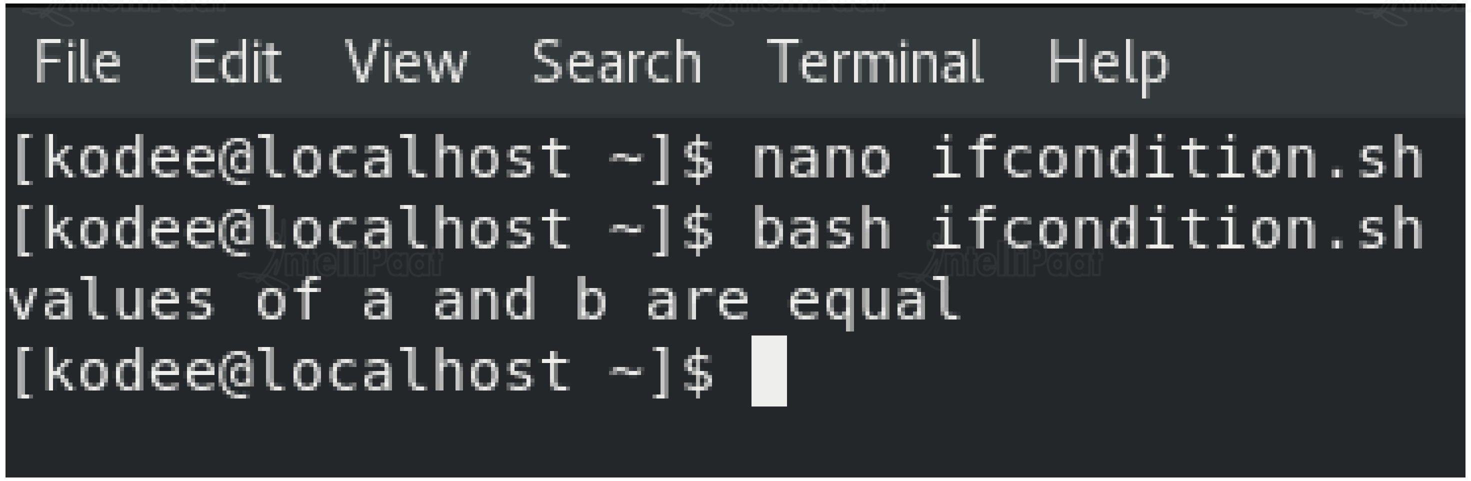
If Statement



```
  GNU nano 2.9.8                     ifcondition.sh

a=1
b=1
if [ $a == $b ]
then
echo "values of a and b are equal"
fi
```

Using If Statement

If Statement



```
[kodee@localhost ~]$ nano ifcondition.sh
[kodee@localhost ~]$ bash ifcondition.sh
values of a and b are equal
[kodee@localhost ~]$ 
```

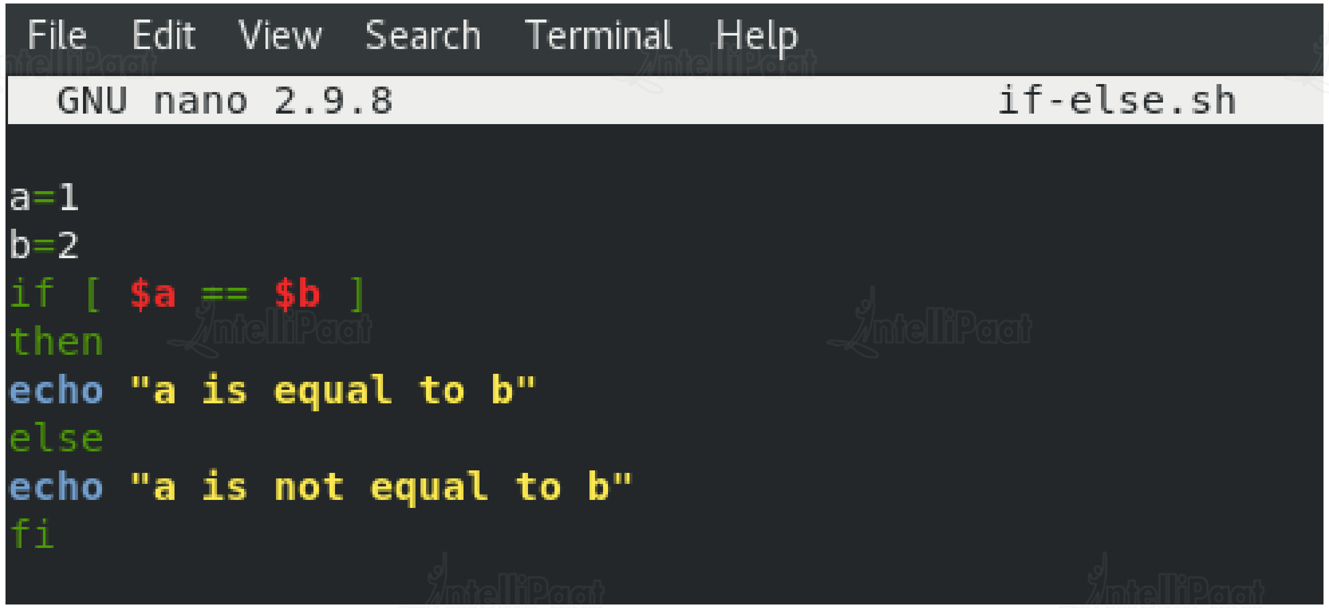
Using If-else Statement

If a condition provided is true, it will enter the statements after **then**. If false, then statements inside **else** will be executed

```
If [Condition]
then
    Statements
else
    Statements
fi
```

Using If Statement

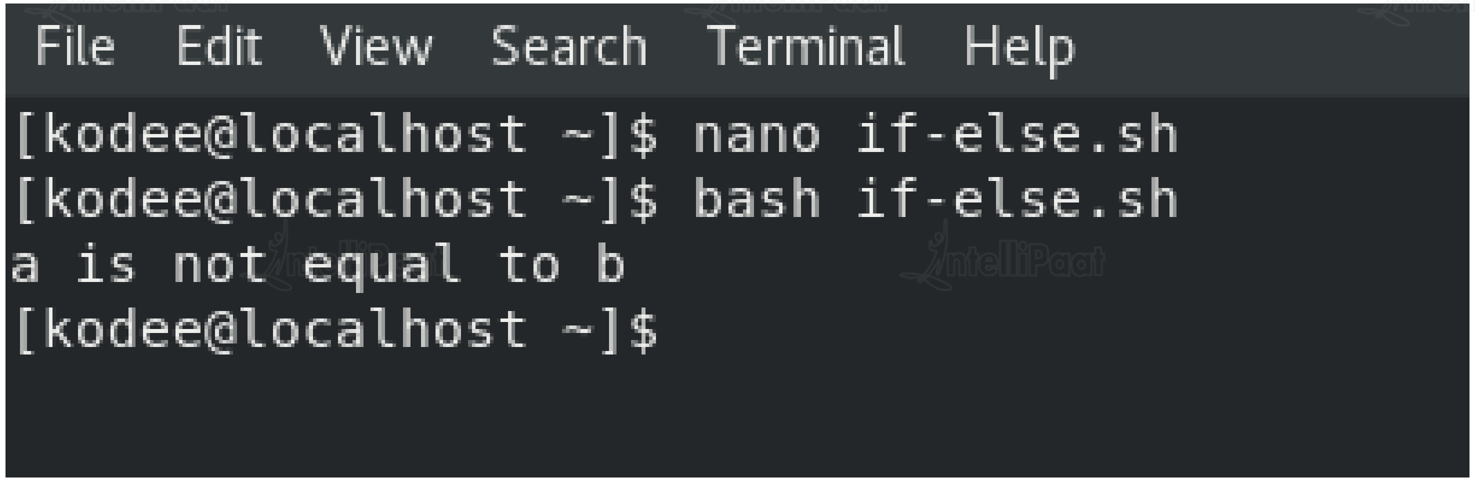
If-else Statement



```
  GNU nano 2.9.8                                       if-else.sh

a=1
b=2
if [ $a == $b ]
then
echo "a is equal to b"
else
echo "a is not equal to b"
fi
```

Using If Statement

If-else Statement



```
[kodee@localhost ~]$ nano if-else.sh
[kodee@localhost ~]$ bash if-else.sh
a is not equal to b
[kodee@localhost ~]$
```

Using Else If Statement

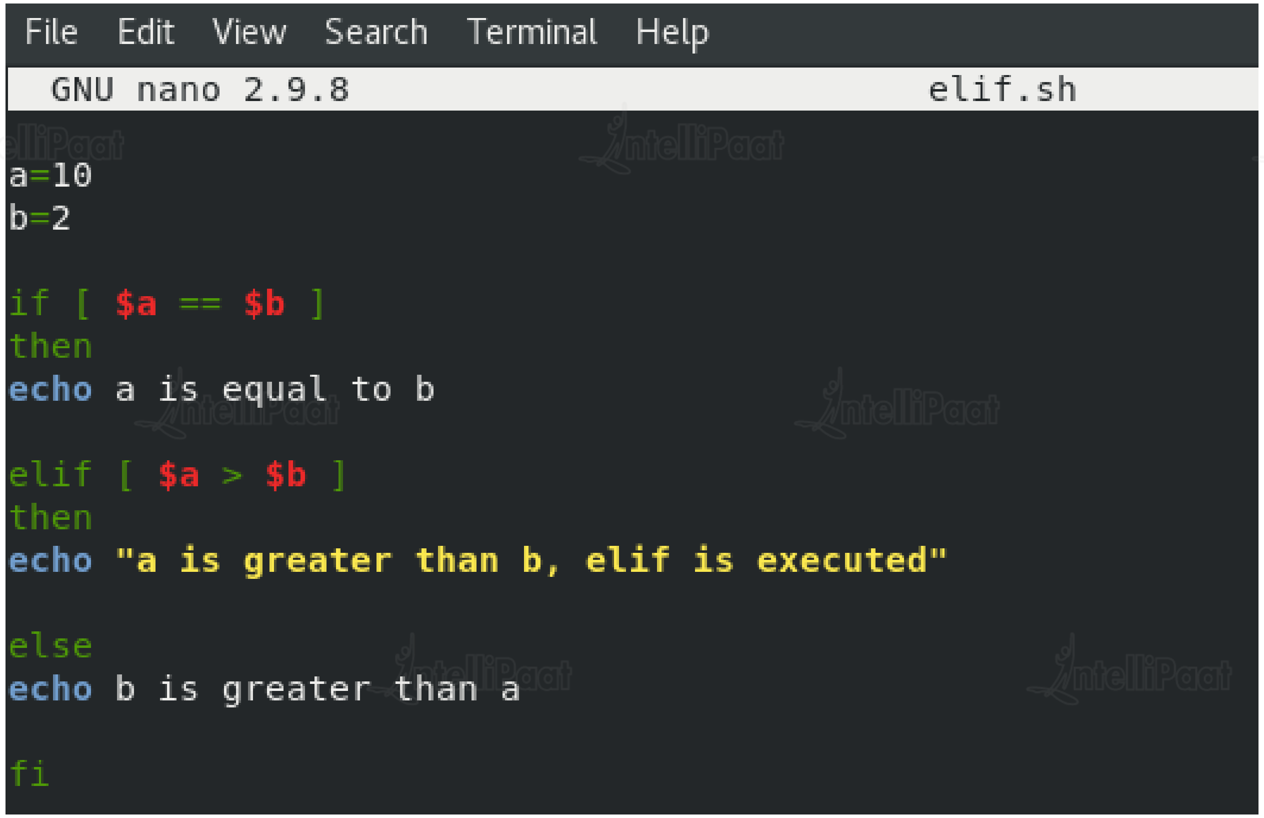
- **If condition1 is True**, Statements inside its **then** will execute.

- **If condition1 is False**, then condition2 is checked.
  - If True, Statements inside it will be executed.
  - If False, the Statements inside else will execute.

```
if [ condition1 ]
then
  Statements
elif [ condition2 ]
then
  Statements
else
  Statements
fi
```

Using Else If Statement

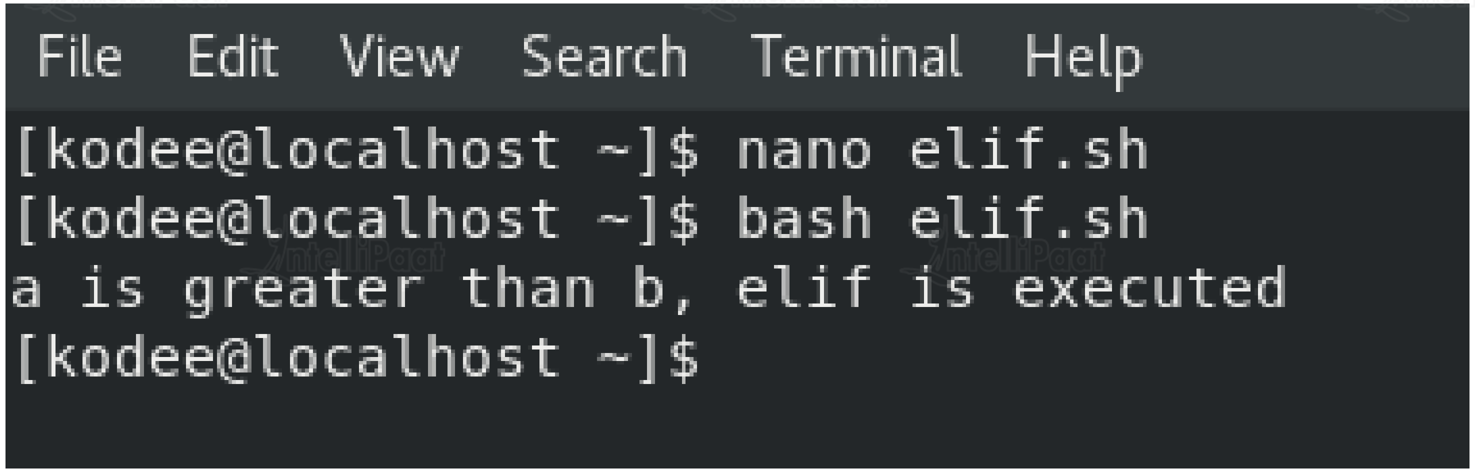
Else If Statement



```
 GNU nano 2.9.8                           elif.sh

a=10
b=2

if [ $a == $b ]
then
echo a is equal to b

elif [ $a > $b ]
then
echo "a is greater than b, elif is executed"

else
echo b is greater than a

fi
```

Using Else If Statement

Else If Statement



```
[kodee@localhost ~]$ nano elif.sh
[kodee@localhost ~]$ bash elif.sh
a is greater than b, elif is executed
[kodee@localhost ~]$
```

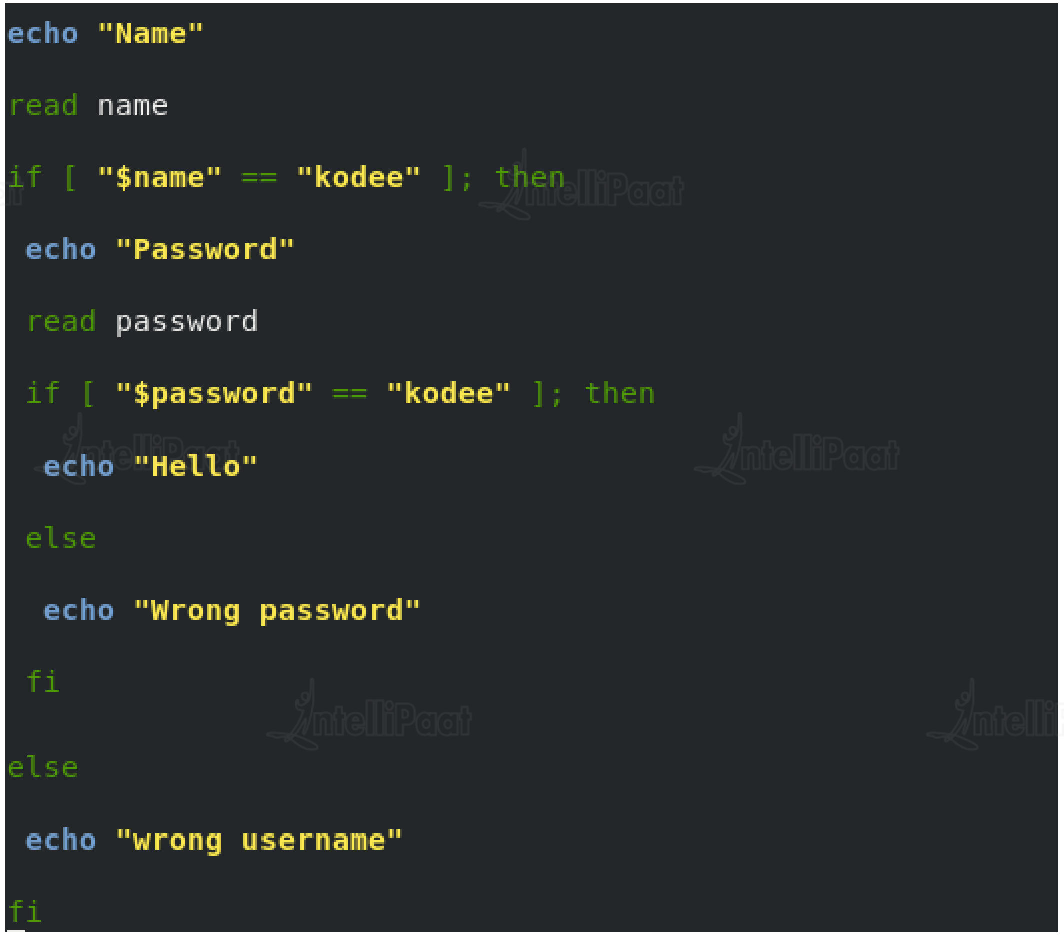
Using Nested If Statement

- **If condition1 is True**, statements inside its **then** will execute.
- **If condition1 is False**, then it goes inside else. Now it checks the **condition2.**
- **If condition2 is True**, statements inside the second if statement is executed.
- **If condition2 is False**, statements inside the second else executes.

```
if [ condition1 ]
then
  Statements
else
then
    If [ condition2 ]
         Statements
    else
         then
            Statements
fi
```

Using Else If Statement

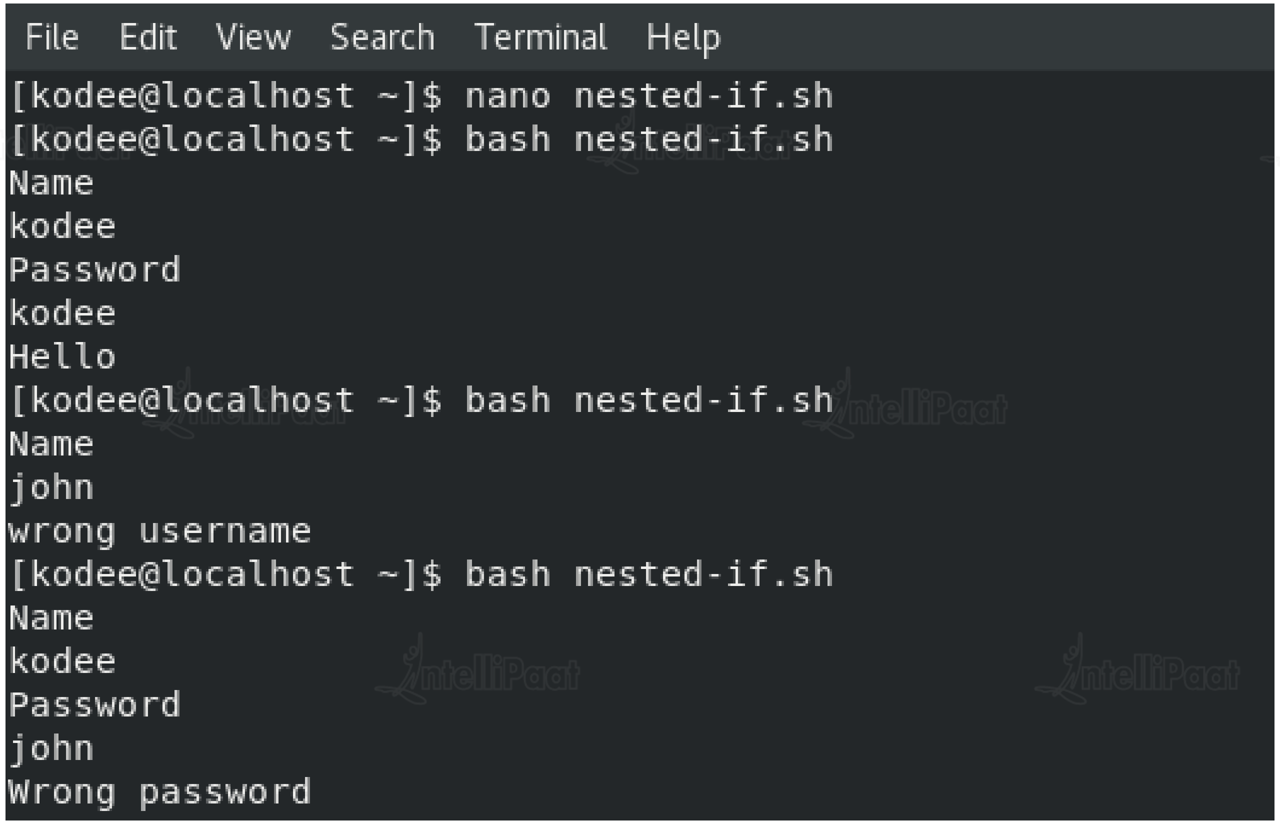
Nested If Statement



```
echo "Name"
read name
if [ "$name" == "kodee" ]; then
 echo "Password"
 read password
 if [ "$password" == "kodee" ]; then
  echo "Hello"
 else
  echo "Wrong password"
 fi
else
 echo "wrong username"
fi
```

Using Else If Statement

Nested If Statement



```
[kodee@localhost ~]$ nano nested-if.sh
[kodee@localhost ~]$ bash nested-if.sh
Name
kodee
Password
kodee
Hello
[kodee@localhost ~]$ bash nested-if.sh
Name
john
wrong username
[kodee@localhost ~]$ bash nested-if.sh
Name
kodee
Password
john
Wrong password
```

Looping Statements

Looping Statements

Types of loops

While Loop

If the given command is TRUE, loop executes. If FALSE, comes out of loop

Until Loop

Same as while, but it will loop until the test case becomes true

For Loop

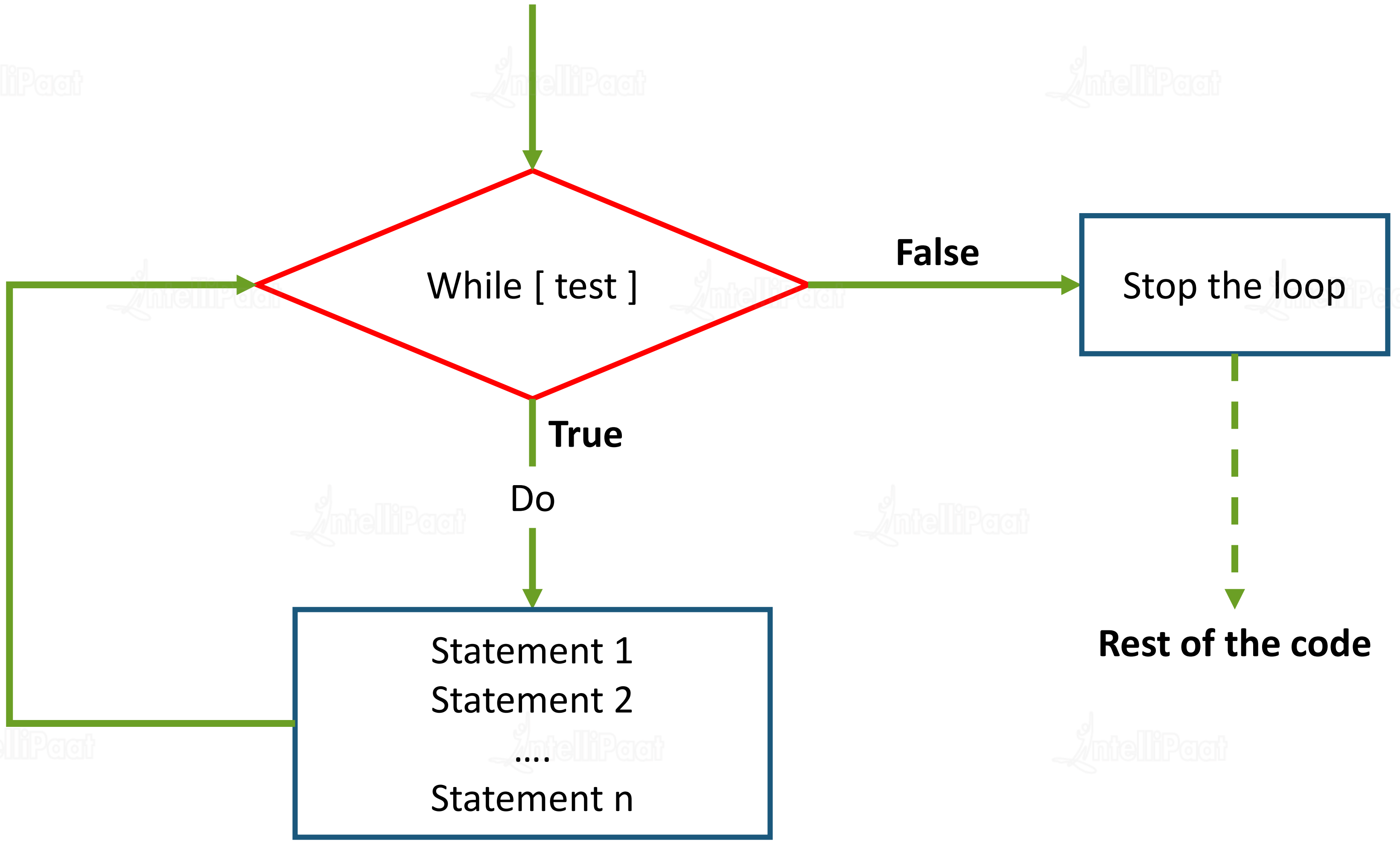
It uses a given set of data to iterate until the given command is FALSE

While loop

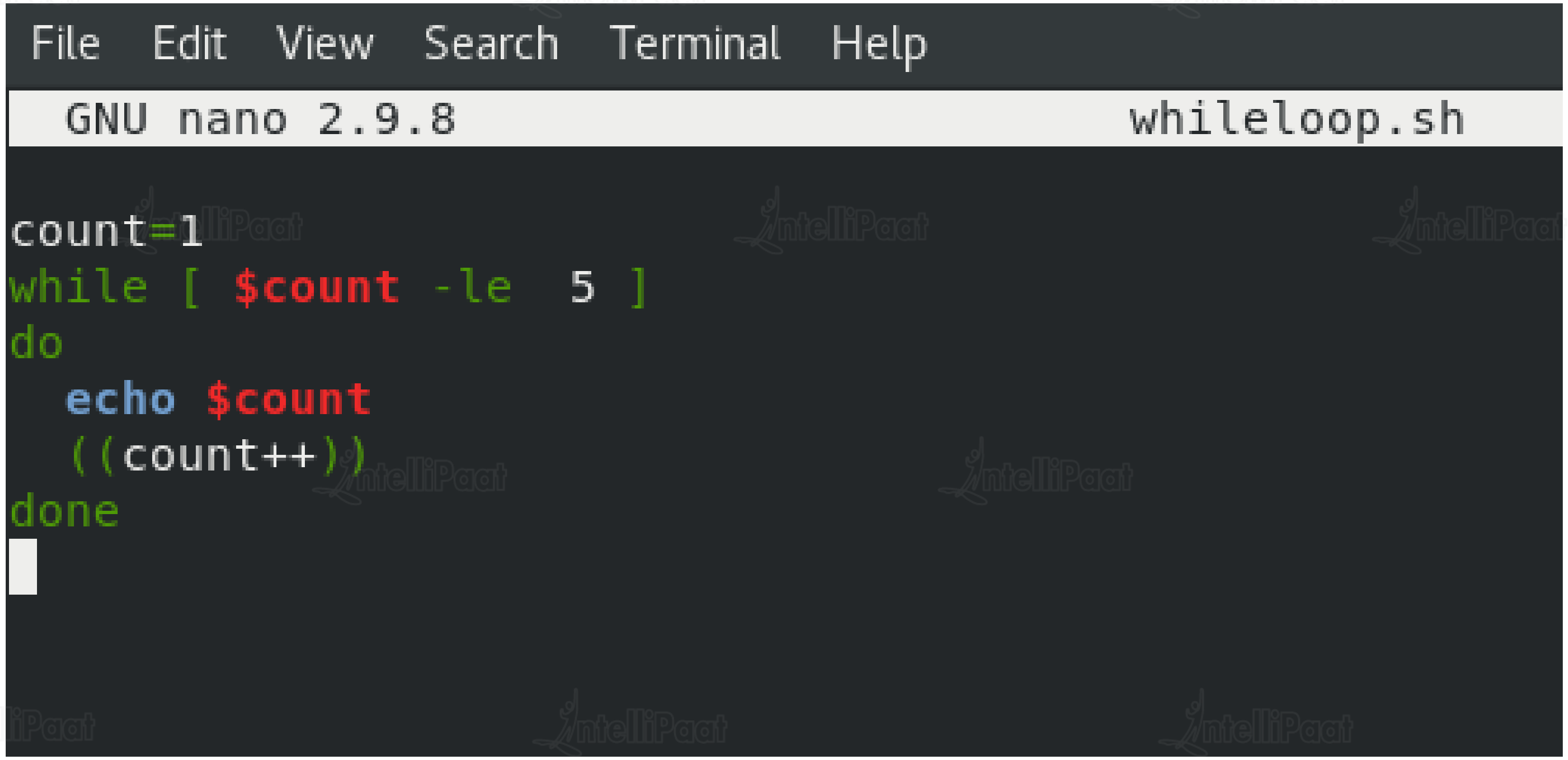
It is simple. When the command is true, it keeps executing the statements

```
while [ command ]
do
    Statements
done
```

While loop


While [test]
False
Stop the loop
True
Do
Statement 1
Statement 2
....
Statement n
Rest of the code

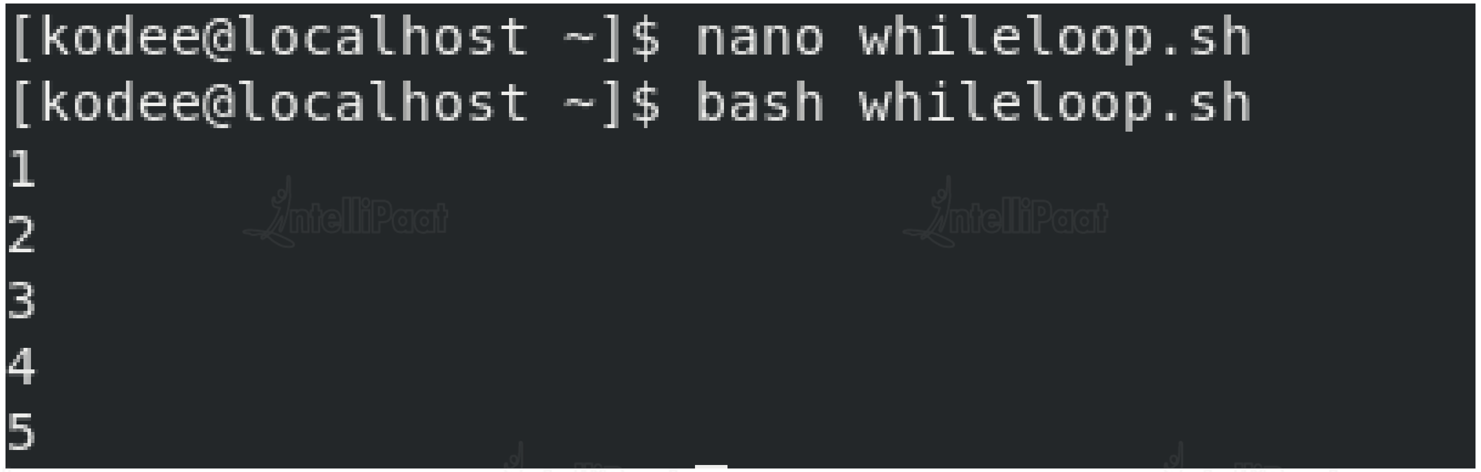
While loop



```
  GNU nano 2.9.8                                    whileloop.sh

count=1
while [ $count -le  5 ]
do
  echo $count
  ((count++))
done
```

While loop

Output



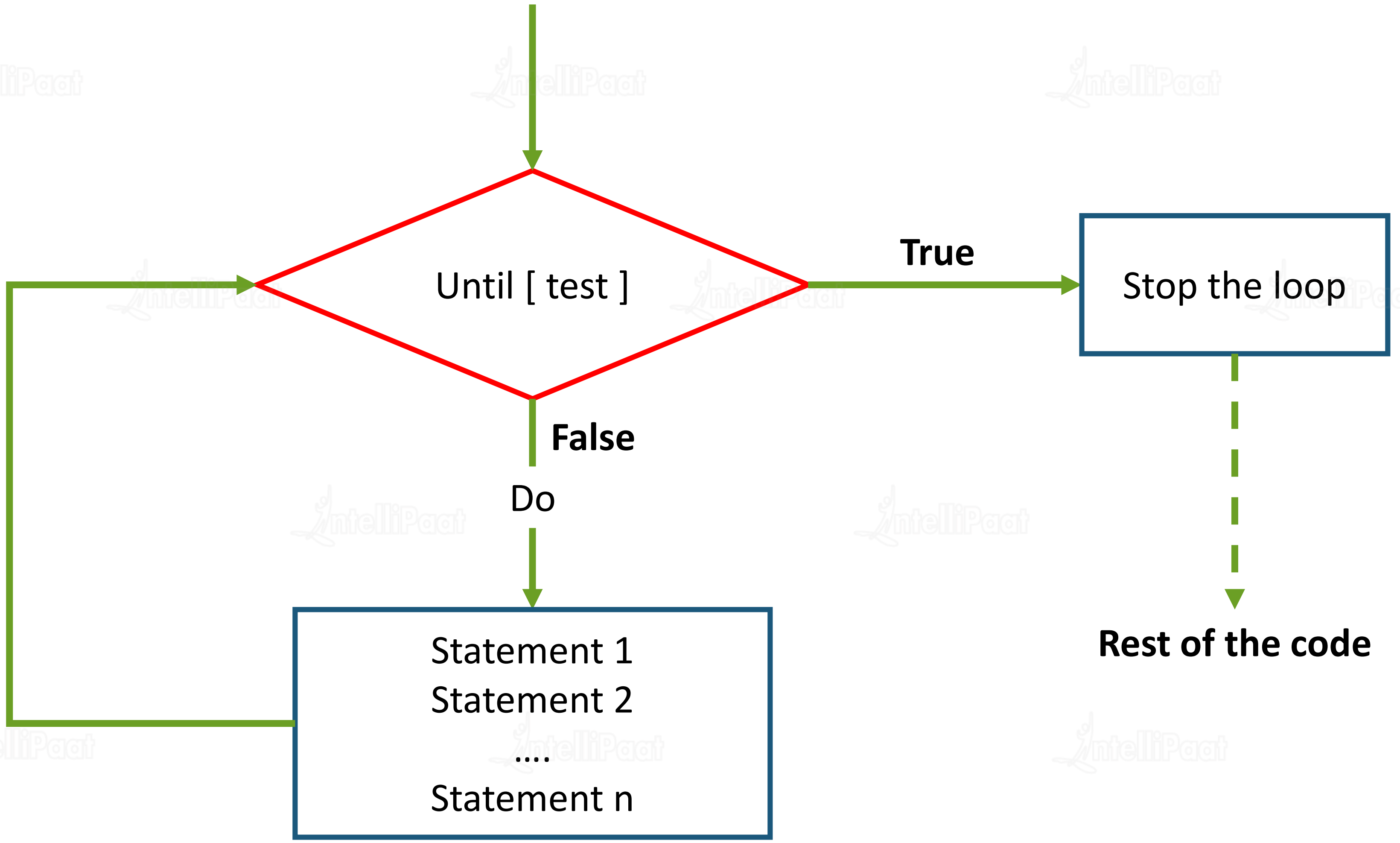
```
[kodee@localhost ~]$ nano whileloop.sh
[kodee@localhost ~]$ bash whileloop.sh
1
2
3
4
5
```

Until loop

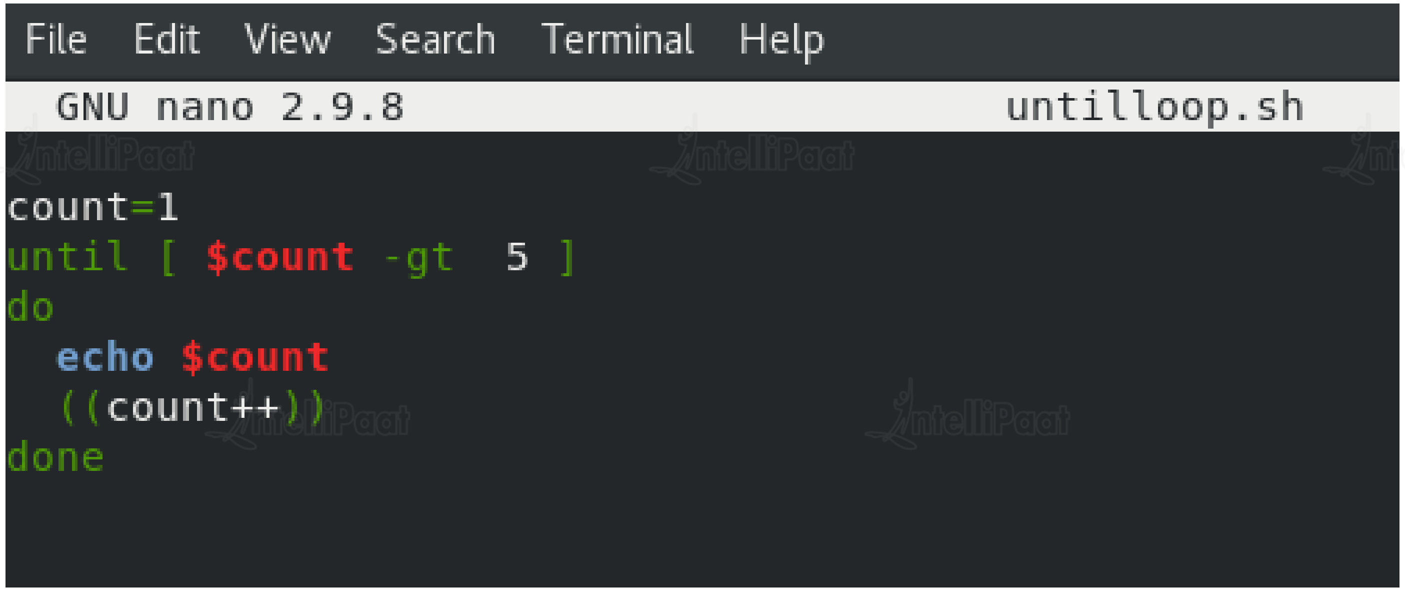
It keeps executing the statements until the command becomes True

```
until [ test case ]

do

      Statements

done
```

Until loop


Until [test]
True
Stop the loop
False
Do
Statement 1
Statement 2
....
Statement n
Rest of the code

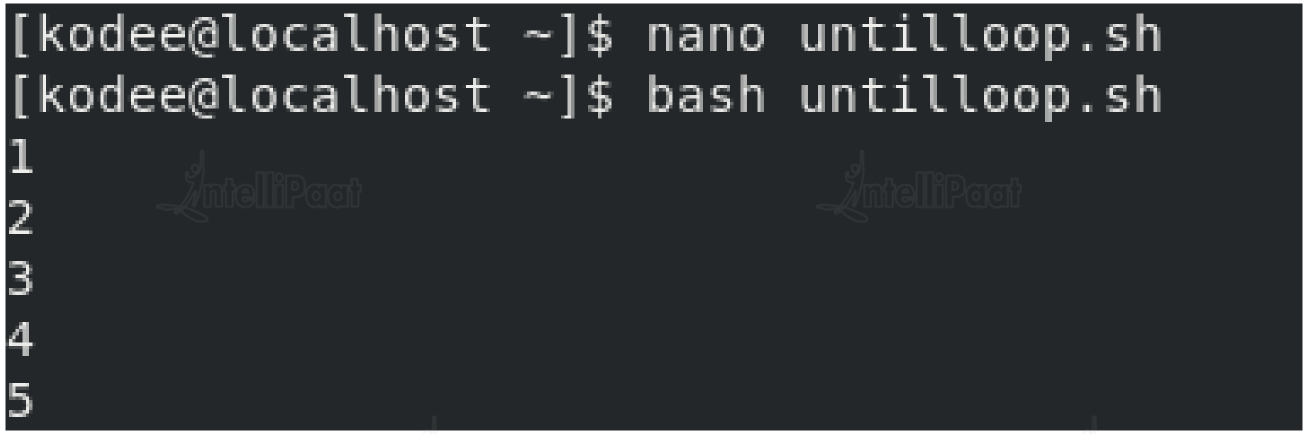
Until loop



```
 GNU nano 2.9.8                                   untilloop.sh

count=1
until [ $count -gt  5 ]
do
  echo $count
  ((count++))
done
```

Until loop

Output



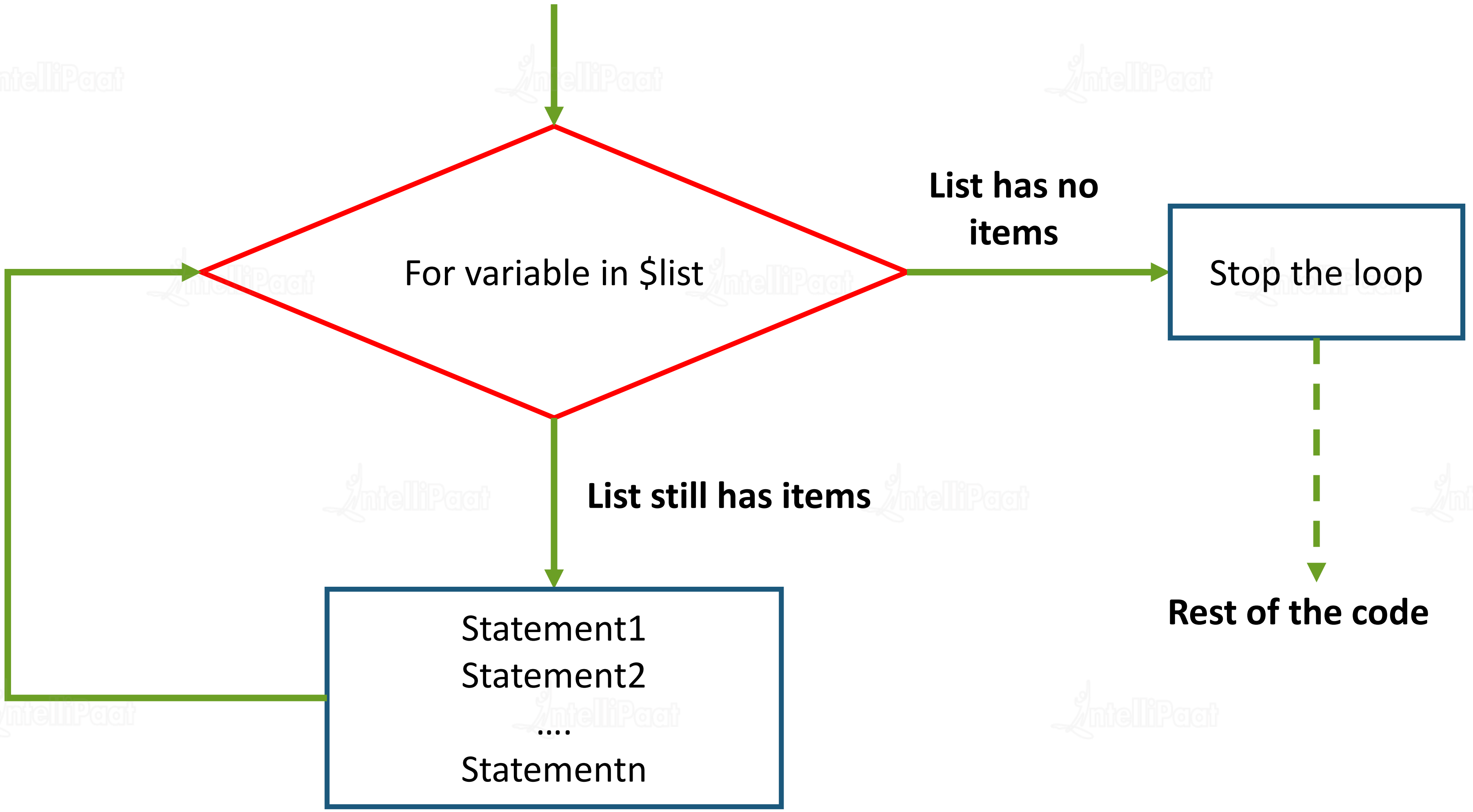
```
[kodee@localhost ~]$ nano untilloop.sh
[kodee@localhost ~]$ bash untilloop.sh
1
2
3
4
5
```

For loop

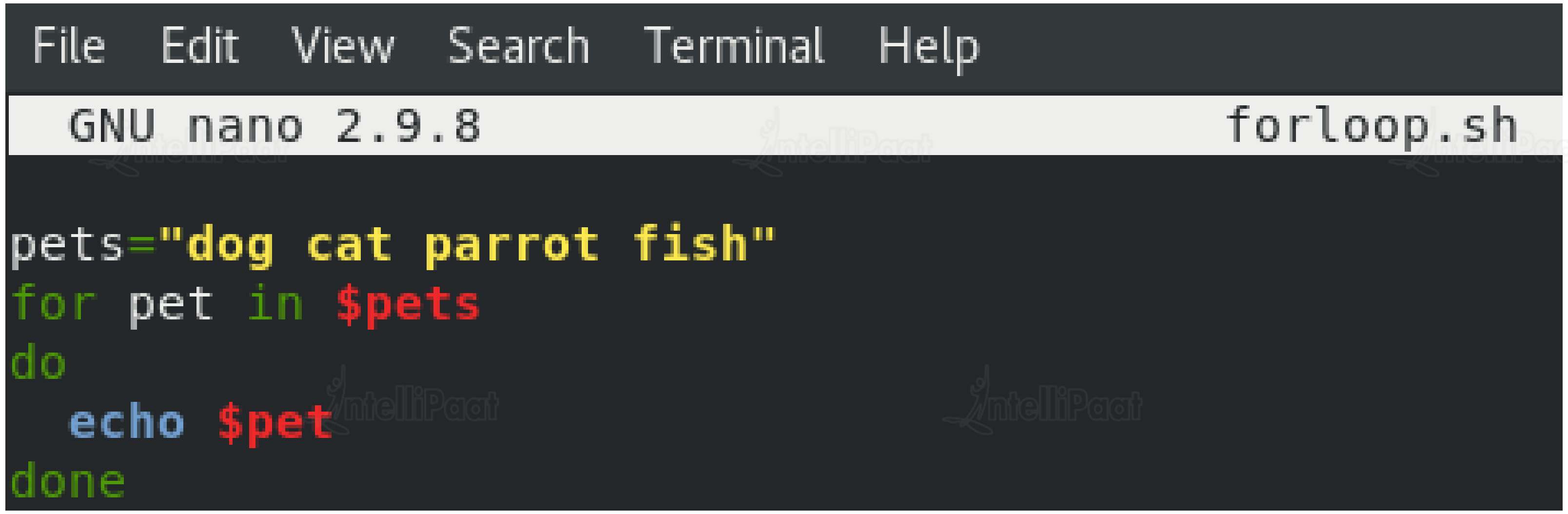
According to the given list, it executes the commands for each item

```
for variable in <list>

do

    Statements

done
```

For loop


For variable in $list
List has no items
Stop the loop
List still has items
Statement1
Statement2
....
Statementn
Rest of the code

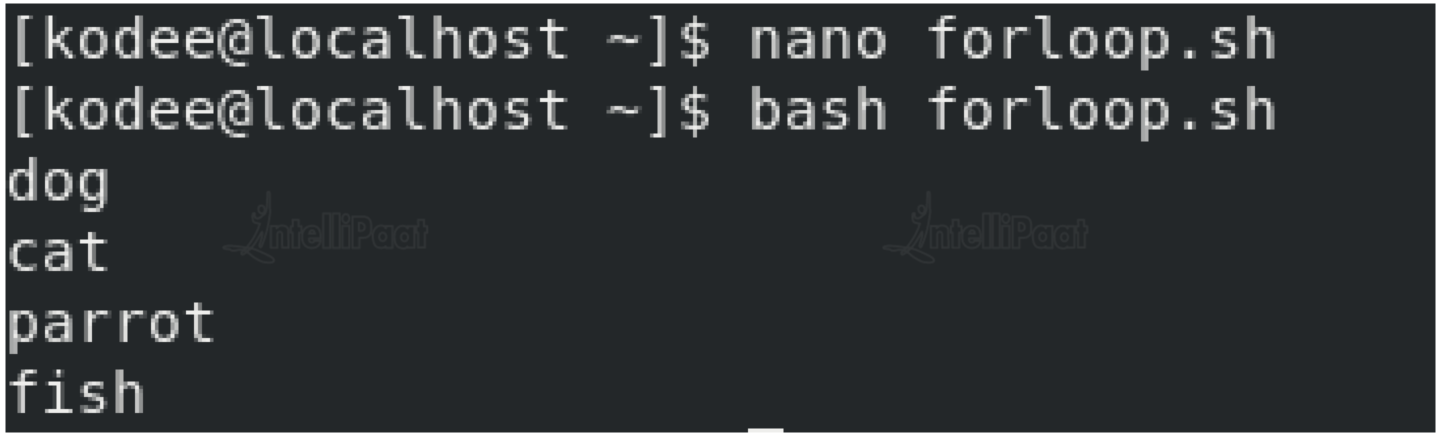
For loop



```
pets="dog cat parrot fish"
for pet in $pets
do
  echo $pet
done
```

For loop

Output



```
[kodee@localhost ~]$ nano forloop.sh
[kodee@localhost ~]$ bash forloop.sh
dog
cat
parrot
fish
```

Flow control statements

Flow control statements

Break

Tells Bash to leave the loop whenever it encounters a Break statement

Continue

Tell Bash to stop the current iteration and start a new iteration altogether

Flow control statements

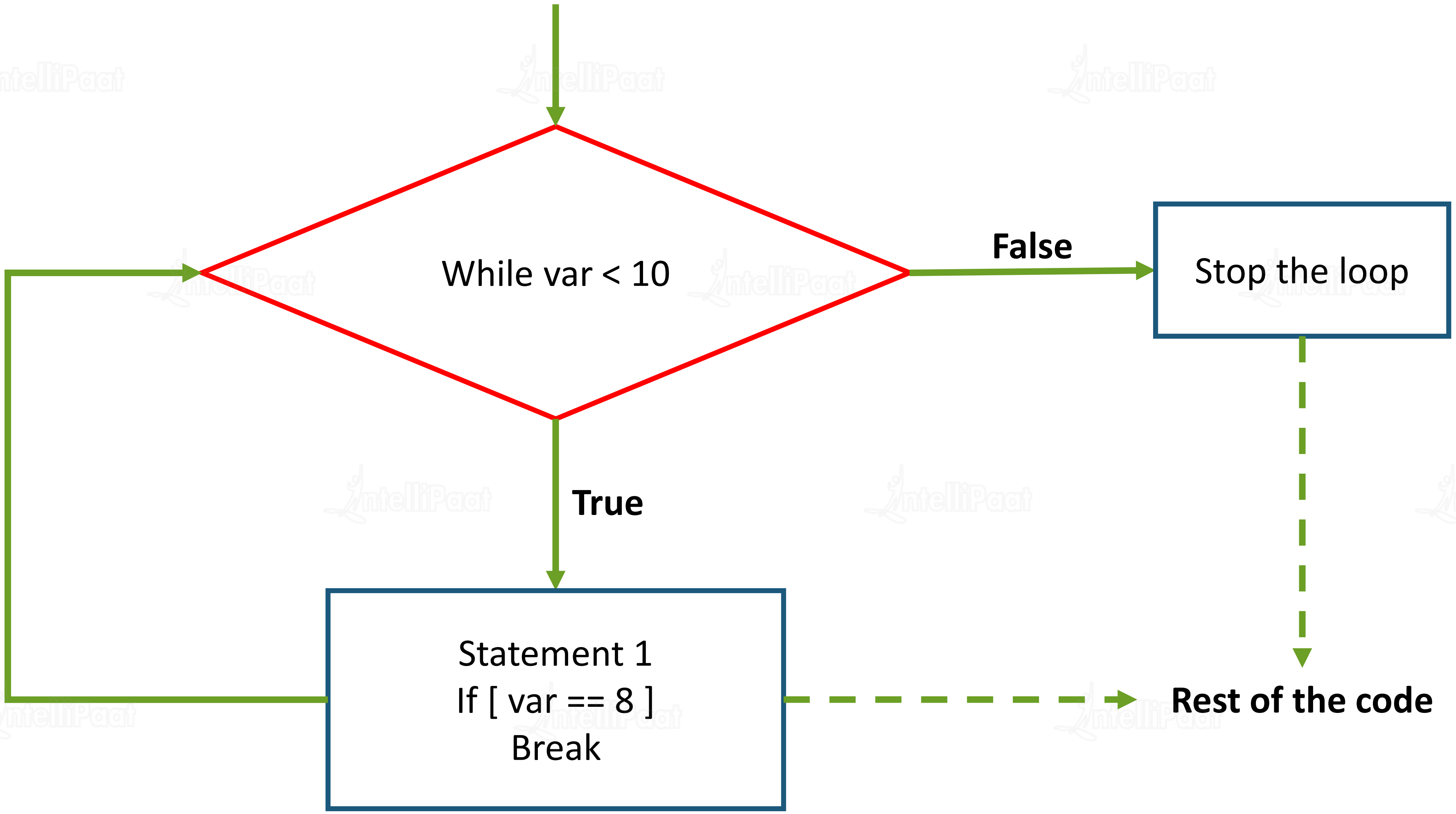
Break

```
while [ command ]

do

    If [ command ]

    then

            break
    fi

done
```

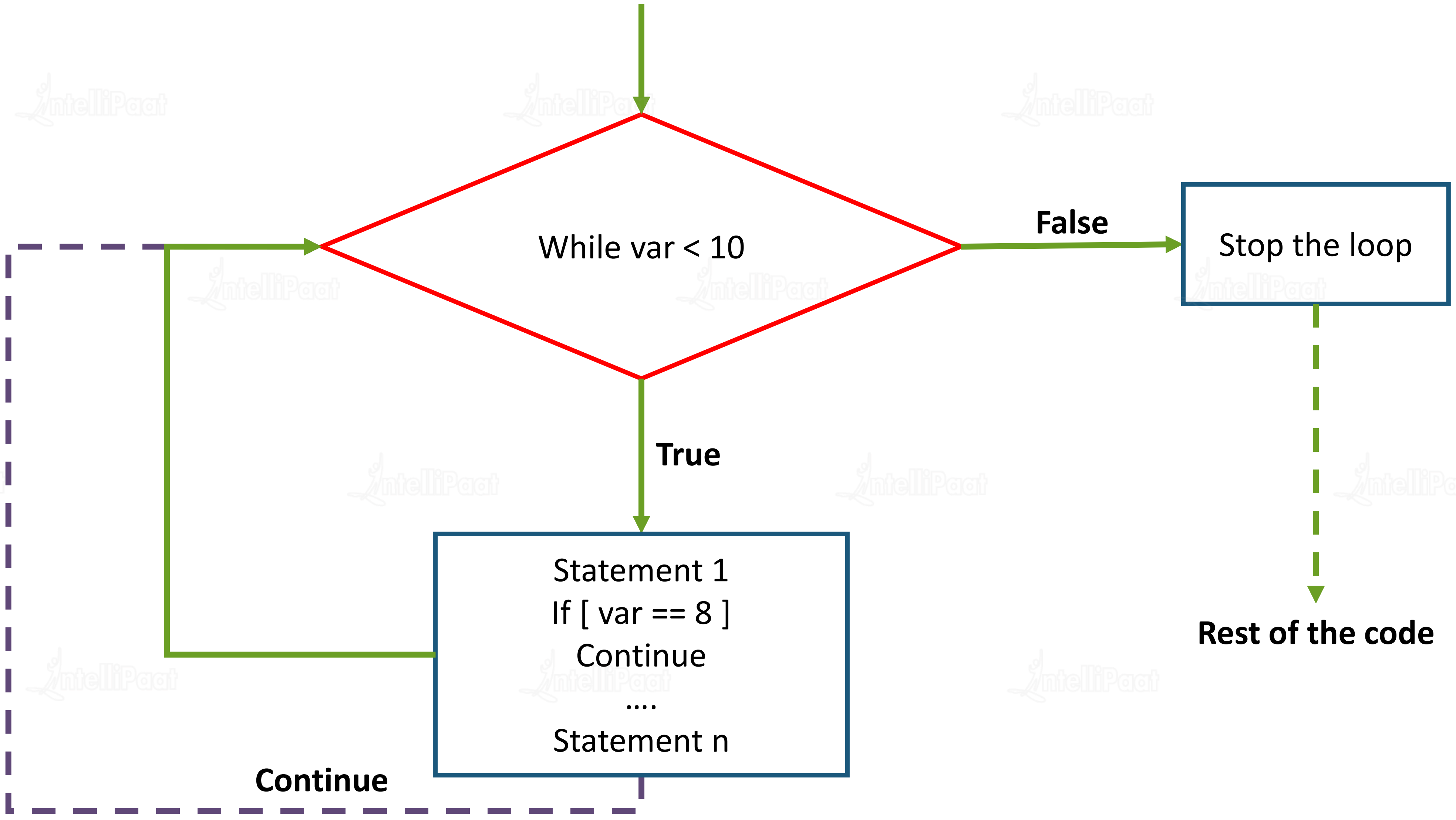
Continue

```
while [ command ]

do

    If [ command ]

    then

            continue
    fi

done
```

Break statement



Continue statement



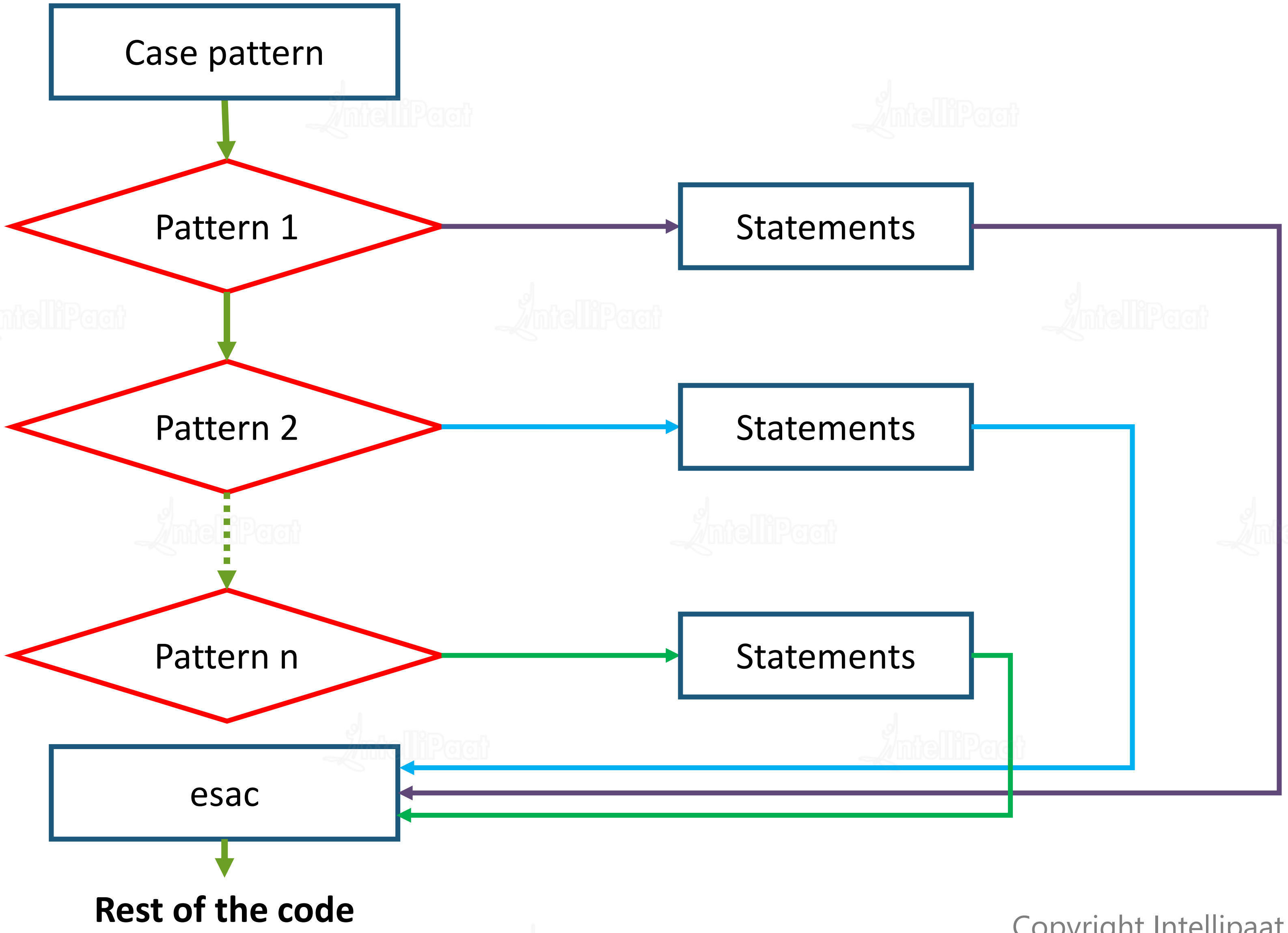
Using case…esac Statement

Using case...esac Statement

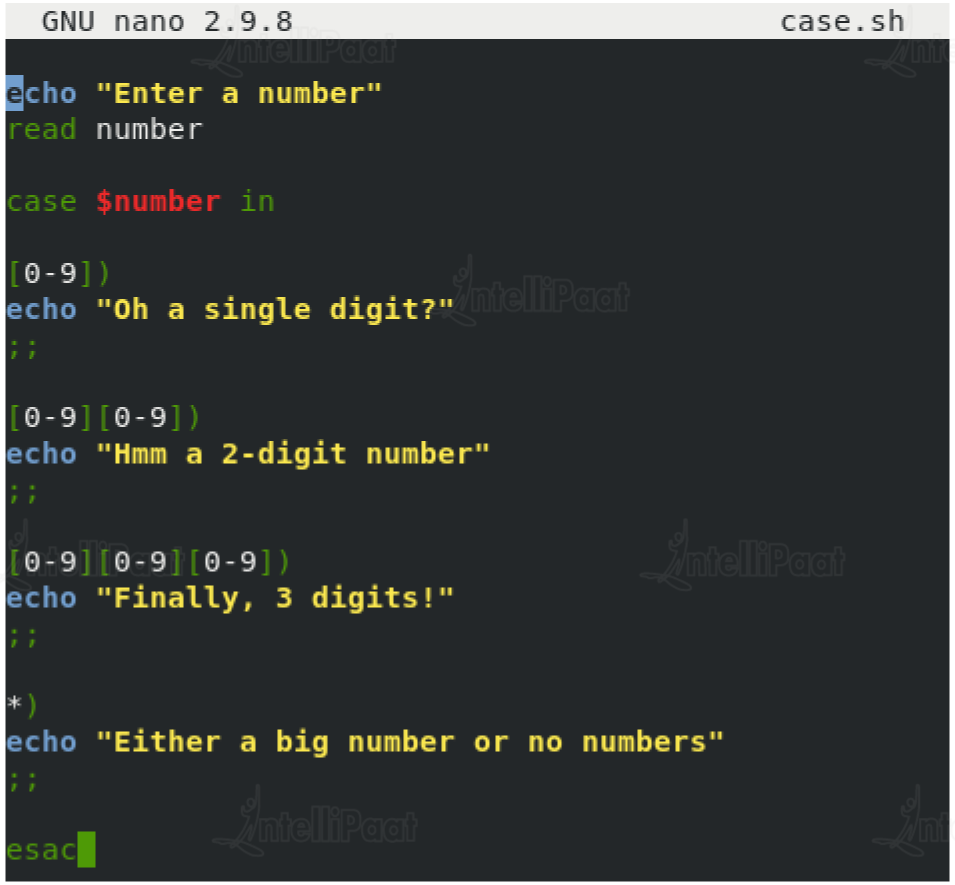
Use Switch statement to choose between multiple options and execute a set of statements under the selected option

```
case word in
        pattern1)
                Statement(s)
                ;;
        pattern2)
                Statement(s)
                ;;
        pattern3)
                Statement(s)
                ;;
        *)
                Default Statement(s)
                ;;
esac
```

Using case…esac Statement



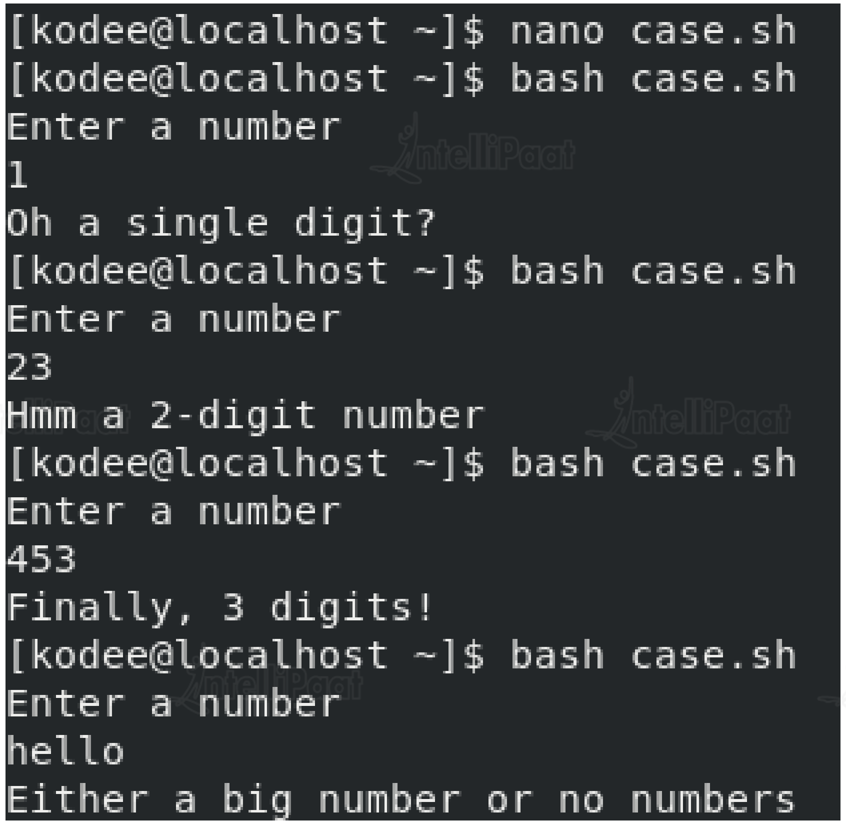
Using case…esac Statement



```
GNU nano 2.9.8                                          case.sh

echo "Enter a number"
read number

case $number in

[0-9])
echo "Oh a single digit?"
;;

[0-9][0-9])
echo "Hmm a 2-digit number"
;;

[0-9][0-9][0-9])
echo "Finally, 3 digits!"
;;

*)
echo "Either a big number or no numbers"
;;

esac
```

Using case…esac Statement

Output



```
[kodee@localhost ~]$ nano case.sh
[kodee@localhost ~]$ bash case.sh
Enter a number
1
Oh a single digit?
[kodee@localhost ~]$ bash case.sh
Enter a number
23
Hmm a 2-digit number
[kodee@localhost ~]$ bash case.sh
Enter a number
453
Finally, 3 digits!
[kodee@localhost ~]$ bash case.sh
Enter a number
hello
Either a big number or no numbers
```

Quiz

Quiz

1. What is the most basic conditional statement?

A. why

B. for

C. if

D. while

Quiz

1. What is the most basic conditional statement?

A. why

B. for

[✓ C. if]

D. while

Quiz

2. Which statement is used to stop the current iteration and start a new iteration?

A. Break

B. Close

C. Do

D. Continue

Quiz

2. Which statement is used to stop the current iteration and start a new iteration?

A. Break

B. Close

C. Do

[✓ D. Continue]

Quiz

3. Why use Case … esac statement?

A. Best looping statement

B. Execute statements under a chosen pattern

C. Execute statements when the loop iterates 5 times

D. Create a case file and close a case file

Quiz

3. Why use Case … esac statement?

A. Best looping statement

B. Execute statements under a chosen pattern [✓]

C. Execute statements when the loop iterates 5 times

D. Create a case file and close a case file

Quiz

4. For loop needs a list for iteration in Linux. True or False.

A. True

B. False

Quiz

4. For loop needs a list for iteration in Linux. True or False.

[✓ A. True]

B. False

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor