Conditional statements

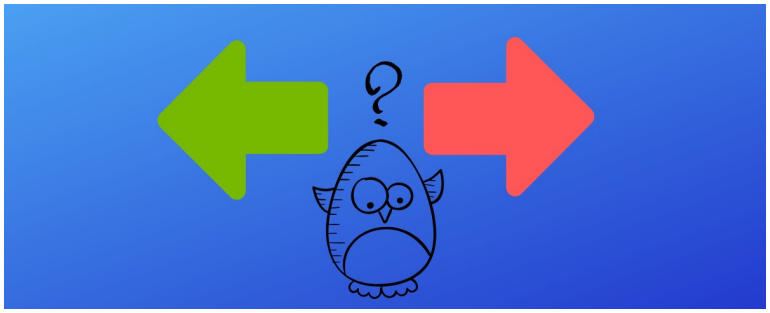
Decision making is one of the most fundamental concepts of computer programming.

Like in any other programming language

```
if statement
if.. else statement
if.. elif.. else statement
```

in Bash can be used to decide whether to execute or not a block of commands based on a test

Are you older than 18?



`if.. else statement`

TEST-COMMAND – test expression syntax

```
[ expression ]
```

Note there must be a space character after [and before]

Look up the man page for test to see all the possible operators you can use to test expression

```
man [ ]   # there must be one space between [ ]
```

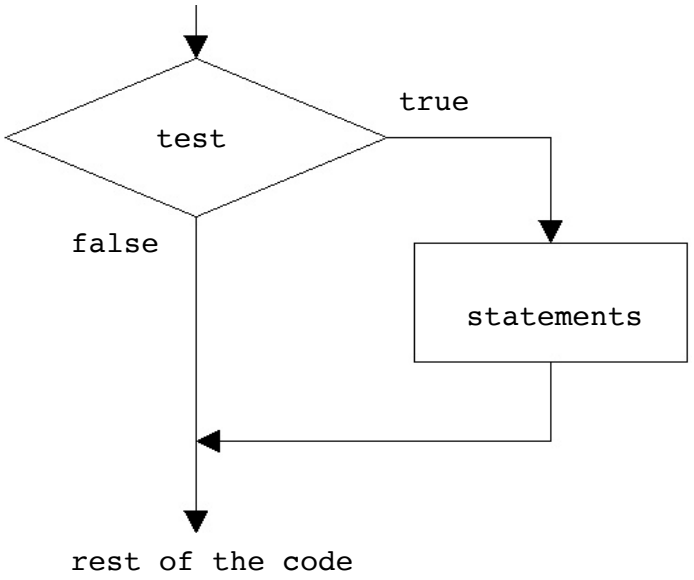
Below the ones we use in this course

```
 Operators for existence of files and directories
[ -d dirname ]   True if dirname exists and is a directory.
[ -f filename ]  True if filename exists and is a regular file.

String operators
[ s1 = s2 ]       True if the strings s1 and s2 are identical.
[ s1 != s2 ]      True if the strings s1 and s2 are not identical.

Arithmetic operators with integer numbers n1 and n2
[ n1 -eq n2 ] True if the integers n1 and n2 are equal.
[ n1 -ne n2 ] True if the integers n1 and n2 are not equal.
[ n1 -gt n2 ] True if the integer n1 is greater than the integer n2.
[ n1 -ge n2 ] True if n1 is greater than or equal to n2.
[ n1 -lt n2 ] True if n1 is less than n2.
[ n1 -le n2 ] True if n1 is less than or equal to n2.
```

simple if statement

If the TEST-COMMAND is TRUE, the STATEMENTS get executed.
If TEST-COMMAND is FALSE, nothing happens, the STATEMENTS get ignored.

```
if TEST-COMMAND
then
    statements
fi
```



Example: Write this in a script called if-num.bash:

```
age=20
if [ $age -gt 18 ]
then
    echo I can vote
fi

echo out of the if-statement
```

Simple If statement examples

if-num.bash

```
age=20
if [ $age -gt 18 ]
then
    echo I can vote
fi
```

In a script called if-string.bash

```
s1=abc
s2=def
if [ $s1 != $s2 ] #Test if the strings s1 and s2 are not identical.
then
   echo strings $s1 and $s2 are different
fi
```

Simple If statement examples

In a script called if-file.bash

```
script=script1.bash
if [ -f $script ] #Test the existence of file script1.bash within current directory
then
  echo file $script exists in my current directory
fi
```

In a script called if-file1.bash

```
script=script1.bash
if [ -f ~/Desktop/$script ] #Test the existence of file script1.bash within Desktop directory
then
  echo file $script exists in Desktop directory
fi
```

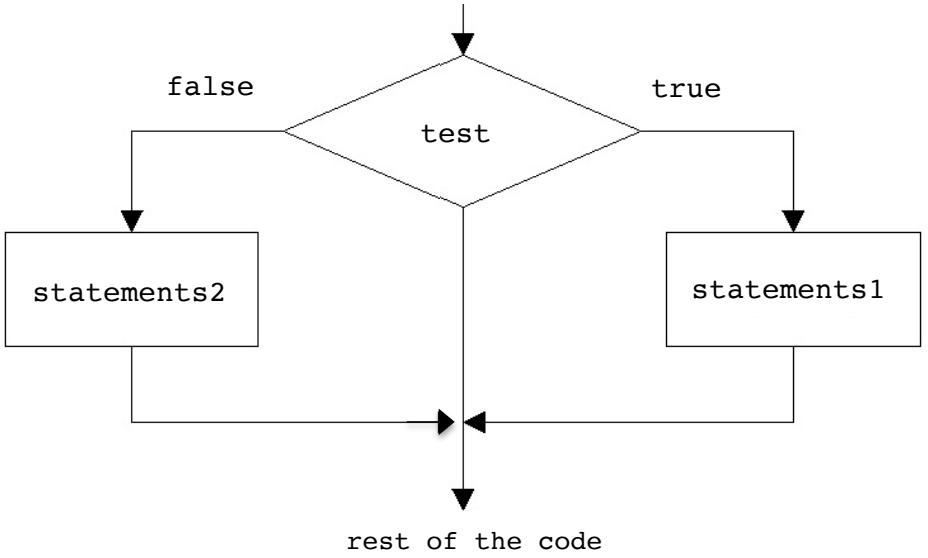
In a script called if-dir.bash

```
if [ -d ~/data-temp ] #Test the existence of directory data-temp within home directory
then
  echo directory data-temp exists in my home directory
fi
```

if .. else statement

If the TEST-COMMAND is TRUE, the STATEMENTS1 get executed.
If TEST-COMMAND is FALSE, the STATEMENTS2 get executed.

```
if TEST-COMMAND
then
    statements1
else
 statements2
fi
```



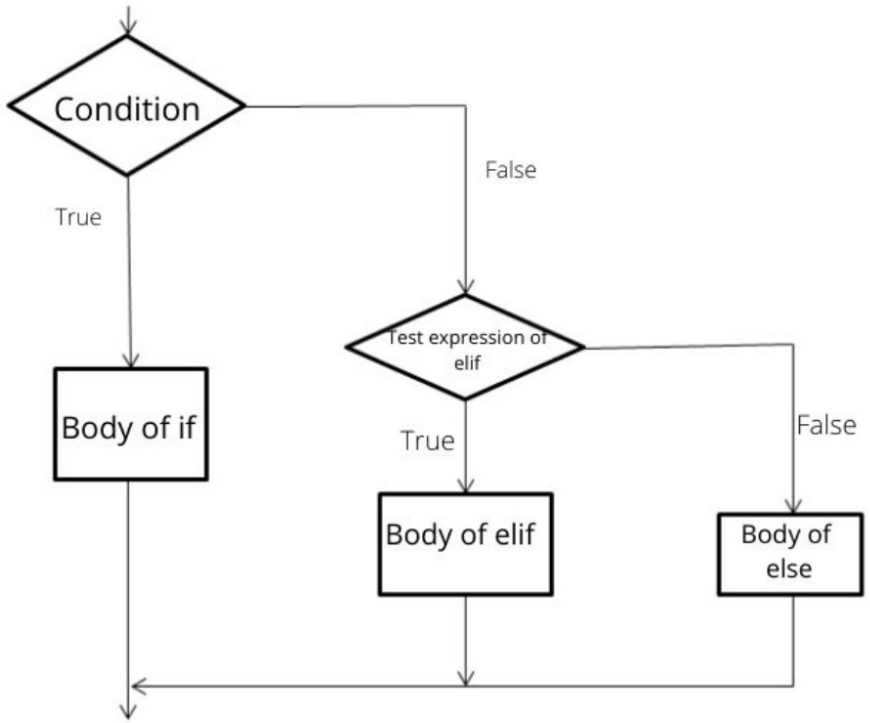
Example: modify if-num.bash to:

```
age=10
if [ $age -gt 18 ]
then
    echo I can vote
else
    echo I cannot vote
fi
```

If .. elif .. else statement

```
if  TEST-COMMAND1
then
     statements1
elif TEST-COMMAND2
then
   statements2
else
   statements3
fi
```

The conditions are evaluated sequentially. Once a test is True, the remaining conditions or tests are not performed, and the bash shell moves to the end (`fi`).



Example: modify if-num.bash:

```
age=18
if [ $age -gt 18 ]
then
    echo I can vote
elif [ $age -eq 18 ]
then
   echo I just turned 18 so I can vote
else
    echo I cannot vote
fi
```

Conditional statements inside a for loop

Simple for loop:

```
for i in {1..5}
do
    echo $i
done
```

Example: if .. else statement inside a for loop
In a script called if-for.bash test this out

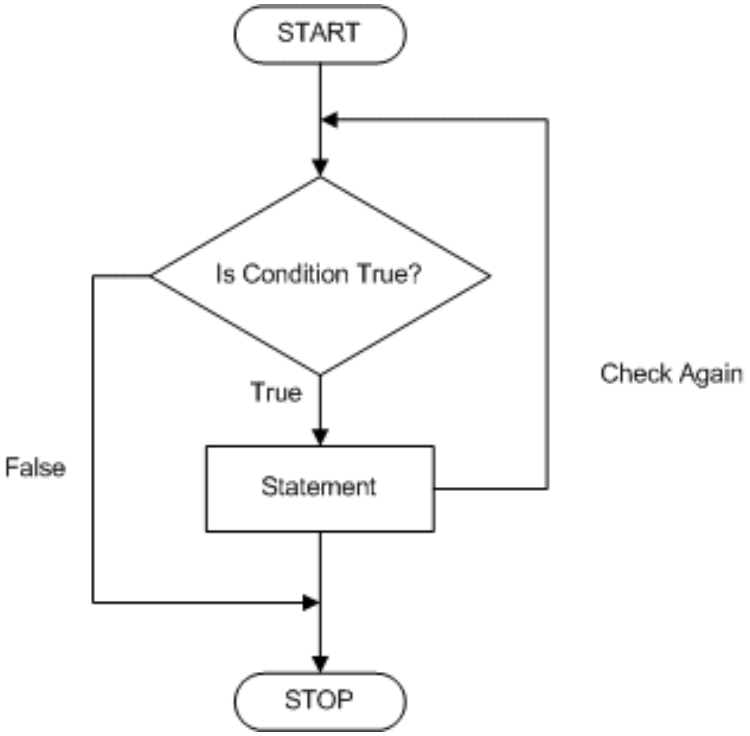
```
for i in {1..10}
do
        if [ $i -le 3 ]
        then
            echo $i
        else
            echo $i greater than 3
        fi
done
```

Try to play with the first test condition, and try to add an elif condition

The while loop

The **while** construct allows for repetitive execution of a block of commands as long as the test is True. When the test is False, the while loop is exited.

```
while TEST-COMMAND
do
   statements
done
```



Example: in a script called my-while.bash

```
myvar=0
while [ $myvar -ne 10 ]
do
  echo $myvar
  myvar=$(( myvar + 1 ))
done
```

Notice that variable myvar:
- is set before the while loop so as the test starts True
- changes value within the while loop, so at a certain iteration the test is false, and the while loop stops

Activity – test conditions for existence of files and directories

```
Operators for files and directories
[ -d dirname ]   True if dirname exists and is a directory.
[ -f filename ]  True if filename exists and is a regular file.
```

ex1. Make a script called **A8-dir.bash**
In it include an **if-else statement** that does the following:
If a directory called data-temp exists in your home directory
    list the content of that directory in long format
Otherwise:
    print a statement that data-temp directory does not exist in your home directory

Activity - test with numbers

Check if a number is divisible by 2.

To test this, the remainder of division by 2 should be 0. For this you can use the modulo operation (sometimes called modulus) %, which finds the remainder of division of one number by another

Try these

```
echo $((1 % 2))   # 1 mod 2 equals 1
echo $((2 % 2))   # 2 mod 2 equals 0

[  $((4 % 2)) –eq 0 ] #test condition
```

ex2. Make a script called **A8-fizz-buzz.bash** and in it:

- Set variable i equal to some number of your choice
- Write an **if -else statement**

  If the number stored in variable i is divisible by 2

  print `fizz`

  Otherwise

  print `buzz`

Activity

ex3. In a script called **A8-strings.bash** do the following:

The user should enter from the command line one of the following colors: red or blue
Make a variable for this.

If the the color entered by the user is blue
    print *blue is a relaxing color*
If the color entered by the user is red
    print *red is an exciting color*
Otherwise
    print *you should enter either red or blue*

Think hard about which if statement structure you should use.
Do not use redundant logic!

Activities - while loop

ex4. Make a script called **A8-while.bash** and write in it

```
myvar=0
while [ $myvar -ne 10 ]
do
  echo $myvar
  myvar=$(( myvar + 1 ))
done
```

Modify the script to:
Q1. Make a while loop that does not loop
Q2 Make a while loop that never stops. (Ctrl-C to stop the script)

Activities – loop and if statements

ex5. Make a script called **A8-temp.bash** and in it

Use file temp-clean1.dat

a. Make a list of unique state codes. Use a pipeline of commands (look at the options of sort) and command substitution. Store the list in variable state.
   If you do not know how to do this part, make this variable
   `state="AZ CA CO NM NV UT"`

b. Make a for loop over the state codes by using variable state, and within the for loop use a simple if statement,
   If the state code is CA
      print to screen the maximum temperature recorded in CA

Optional

In a script called **A8-vote.bash** write a modified version of the code reported below in order to obtain this output. You should include a for loop. Do not hardcode numbers!

```
At age 15 I cannot vote
At age 16 I cannot vote
At age 17 I cannot vote
I just turned 18 so I can vote
At age 19 I can vote
At age 20 I can vote
```

```
age=18
if [ $age -gt 18 ]
then
    echo I can vote
elif [ $age -eq 18 ]
then
   echo I just turned 18 so I can vote
else
    echo I cannot vote
fi
```

Optional- Fix redundant logic

In a script **A8-fix-guess.bash,** we want to implement a simple script where the user can guess a color. The user will enter a color name from the command line, and you run the script like:

```
. A8-fix-guess.bash red
```

The code below is meant to implement this test:

If the user color is equal to the color defined in the script:

print to screen *you got my color*

Otherwise:

print to screen *your color does not match my color.*

```
usercolor=$1
color=yellow

if [ $usercolor = $color ]
then
   echo you got my color
fi
if [ $usercolor != $color ]
then
   echo your color does not match my color.
fi
```

However, the code contains redundant logic. Your task is to fix the redundant logic.

Optional - for loop and if-statements

Make a script **A8-fizz-buzz-again.bash** to print a a total of 5 Fizzes, 5 Buzzes

Hint: make a for loop over numbers 1 to 10. Fizzes should be printed if a number is divisible by 2, and buzzes should be printed otherwise

Optional – looping, selecting and counting

In a script called **A8-loop-dir.bash** make one for loop over all the contents of your current working directory. The repeated task of the for loop is to count the number of directories present in your home working directory.

Print to screen the final count.

Generate the loop values by using a meta-character that matches all the contents of your current directory. The loop should be a counting loop with a nested test condition for existence of directory.

Do not use ls anywhere in your script. Think about which if-statements you should use.

Below are some parts of the code. Your task is for you to complete it.

```
for i in ~/
do
   if [   ]
   then
      count=$count


done
echo The final count is
```

Submit to A8:

A8-dir.bash	0.5 point
A8-fizz-buzz.bash	0.5 point
A8-strings.bash	1 point
A8-while.bash	0.5 point
A8-temp.bash	0.5 point

- Optional
  - A8-fizz-buzz-again.bash - Optional
  - A8-vote.bash - Optional
  - A8-loop-dir.bash - Optional