Shell Script: Conditionals

This reading will get you sufficiently familiar with bash *conditionals* for the final project.

Conditionals are ways of telling a script to do something *under specific condition(s)*.

In this reading, you will learn about shell script conditionals using `if` `else` .

If

Syntax:

```
if [ condition ]
then
    statement
fi
```

You must always put spaces around your conditions in the `[` `]` .

Every `if` *condition block must be paired with a* `fi` .

Example

```
$ cat if_example.sh
a=1
b=2
if [ $a -lt $b ]
then
    echo "a is less than b"
fi

$ sh if_example.sh  # sh tells the terminal to run the script if_example.sh using t
a is less than b
```

If-Else

Syntax:

```
if [ condition ]
```

```
2    then
3        statement_1
4    else
5        statement_2
6    fi
```

You don't use `then` for `else` cases.

Example

```
1    $ cat if_else_example.sh
2    a=3
3    b=2
4    if [ $a -lt $b ]
5    then
6        echo "a is less than b"
7    else
8        echo "a is greater than or equal to b"
9    fi
10
11   $ sh if_else_example.sh
12   a is greater than or equal to b
```

Elif

The statement `elif` means "else if" :

Syntax:

```
1    if [ condition_1 ]
2    then
3        statement_1
4    elif [ condition_2 ]
5    then
6        statement_2
7    fi
```

Example

```
1    $ cat elif_example.sh
2    a=2
3    b=2
```

```
if [ $a -lt $b ]
then
    echo "a is less than b"
elif [ $a == $b ]
then
    echo "a is equal to b"
else # Here a is not <= b, so a > b
    echo "a is greater than b"
fi

$ sh elif_example.sh
a is equal to b
```

Nested Ifs

As in other prgramming languages, it’ s also possible to nest if-statements.

Syntax:

```
if [ condition_1 ]
then
    statement_1
elif [ condition_2 ]
    statement_2
    if [ condition_2.1 ]
    then
        statement_2.1
    fi
else
    statement_3
fi
```

Example

```
$ cat nested_ifs_example.sh
a=3
b=3
c=3
if [ $a == $b ]
then
    if [ $a == $c ]
    then
```

```
            if [ $b == $c ]
            then
                echo "a, b, and c are equal"
            fi
        fi
    else
        echo "the three variables are not equal"
    fi

    $ sh nested_ifs_example.sh
    a, b, and c are equal
```

Alternatively, this example could have been simplified to a single if-statement:

```
    a=3
    b=3
    c=3
    if [ $a == $b ] && [ $a == $c ] && [ $b == $c ]
    then
        echo "a, b, and c are equal"
    else
        echo "the three variables are not equal"
    fi
```

`&&` *means "and"*

Bonus: "test"

Sometimes, instead of using brackets around conditions, you'll see the `test` command in use:

Example

```
    $ cat test_example.sh
    a=1
    b=2
    if test $a -lt $b
    then
        echo "a is less than b"
    fi

    $ sh test_example.sh
    a is less than b
```

`test` and `[ ]` are the same command. We encourage using `[ ]` instead as it's more readable.
