**Quest00**

Remember to git add && git commit && git push each exercise!

We will execute your function with our test(s), please DO NOT PROVIDE ANY TEST(S) in your file

For each exercise, you will have to create a folder and in this folder, you will have additional files that contain your work. Folder names are provided at the beginning of each exercise under submit directory and specific file names for each exercise are also provided at the beginning of each exercise under submit file(s).

**Introduction**

Welcome to the Shell environment quest.

We will dive into the environment of the famous command box. It's the Command center of the computer, very powerful but might be a little confusing at first sight.

In this quest, you will learn how to create directories and files with the right permissions, to read the listing directory command and to git push all your exercises. You will also use the archive command tar and you will finish by writing your first script.

If we compare learning how to stand up before being able to walk then run in order to enjoy playing a sport with this quest you will learn to stand up! :-)

**My First File**

* Submit directory: ex00
* Submit file: ["my\_first\_file"]

For this very first assignment, you will be asked to create a file called my\_first\_file in your directory.

Which directory? It's specified above this sentence (Submit directory: ex00 ;-))

Each assignment follow the same structure, you have to create a directory, go to the directory (cd) and work on your assignment.

Proceed by creating a file with the command: touch as you've seen in the video.

To verify if everything is in order: ls

You should see the file my\_first\_file.

*Tip* To test if your exercise(s) is/are correct(s), you can execute the command gandalf in your terminal.

**My First File With Content**

* Submit directory: ex01
* Submit file: ["my\_first\_file\_with\_content"]

Create a file called my\_first\_file\_with\_content in your directory. Update its size.

* Size needs to be: 40.

How can a file be size of 40? Each character (or letter) is 1, so you simply need to add 40 characters inside the file. ;-)

In order to print this information, use the command ls -l (command is ls and -l is one option).

**Example00**

$>ls -l

xxxxxxxxx x xxxxx xxxxx 40 May 25 14:30 my\_first\_file\_with\_content

$>

We've replaced some values with x, it means they are not important at the moment. You can disregard them :)

**Tips**

$> is a very common way to represent a prompt.

How to read the above Example00:

* You are in a shell (the $>) and you've executed the command ls -l.
* It has printed: xxxxxxxxx x xxxxx xxxxx 40 May 25 14:30 my\_first\_file\_with\_content
* And the last line is reprinting the prompt ($>).

*Tips* The terminal is the little box where you execute command at the bottom part of Docode. To test if your exercise(s) is/are correct(s), you can execute the command gandalf in your terminal.

**My First File With Content And Perms**

* Submit directory: ex02
* Submit file: ["my\_first\_file\_with\_content\_and\_perms.tar"]

**Part 00**

Create a file called my\_first\_file in your directory. Update its size THEN its permissions. (in this order: because you will be asked to remove the permission to WRITE to a file which will be problematic to add a content inside it ;-))

* Size needs to be: 40 (be creative, here, how a file can have a size of 40...? :))
* Permissions need to be: r--r-xr-x (Google chmod? :-))

In order to print this information, use the command ls -l (command is ls and -l is one option).

**Example00**

$>ls -l

-r--r-xr-x 1 login group 40 May 25 14:30 my\_first\_file\_with\_content\_and\_perms

$>

**Tips**

$> is a very common way to represent a prompt.

How to read the above Example00:

* You are in a shell (the $>) and you've executed the command ls -l.
* It has printed: -r--r-xr-x 1 login group 40 May 25 14:30 my\_first\_file\_with\_content\_and\_perms
* And the last line is reprinting the prompt ($>).

**Tip**

The terminal is the little box where you execute command at the bottom part of Docode.

**Part 01**

Once you’ve finished the previous steps, execute the following command to create the file that will be submitted:

tar -cf my\_first\_file\_with\_content\_and\_perms.tar my\_first\_file\_with\_content\_and\_perms

*Tip* To test if your exercise(s) is/are correct(s), you can execute the command gandalf in your terminal.

**My Gitaddcommitpush**

* Submit directory: ex03
* Submit file: ["file"]

Time to Git!

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. @Wikipedia

1. Create a file with the word "content" as the actual content of the file (disturbing?!)

$>echo "content" > file

1. Add that file to the commit (feel free to Google what a commit is!)

$>git add file

1. Prepare the commit with a message (please find below an example of a commit message; you should provide an accurate message for each commit you will do during your software engineer's life)

$>git commit -m "I swear I will ALWAYS add a great message for my commits"

1. Push the commit

$>git push

1. Now we know you know how to use git. No excuses and ALWAYS PUSH YOUR WORK. :-)

Extra: 0. git log

1. git status
2. git clone

**My Z**

* Submit directory: ex04
* Submit file: ["my\_z"]

Create a file called my\_z that returns "Z", followed by a new line, whenever the command cat is used on it.

$>cat my\_z

Z

$>

"Z" Stands For Zorro. :-)

**My Mid Ls**

* Submit directory: ex05
* Submit file: ["my\_midLS"]

It's time to write your first script. A script is a file which contains commands and you will execute it to run them. It's very useful to combine multiple commands. In this first script, you will only have one command.

In a my\_midLS file, write a command line that will list all files and directories in your current directory (except for hidden files or any file that starts by a dot - yes, that includes double-dots), by order of time modified. Make sure directories' names are followed by a slash character.

$>cat my\_midLS

#!/bin/sh

#ls -OPTIONSSS :-)

$>

Example of output:

$>chmod 755 my\_midLS

$>mkdir dir00 dir01 dir02

$>touch file00 file01

$>./my\_midLS

file01 file00 dir02/ dir01/ dir00/ my\_midLS

$>

**Tips** What has not been asked for should not be done! man ls will be your friend! You have to file the right combination of option to make the command ls to add the / at the end of each directories and sort the content by time ;-)

Git doesn't store empty directories, so you will need an additional step: for each directory, create a file without any content called: .gitkeep

**Example**

$>pwd

/home/docode/quest00/ex04

$>ls -lRa

total 8

drwxr-xr-x 8 login group 256 May 25 10:10 .

drwxr-xr-x 9 login group 288 May 25 10:10 ..

drwxr-xr-x 3 login group 96 May 25 10:10 dir00

drwxr-xr-x 3 login group 96 May 25 10:10 dir01

drwxr-xr-x 3 login group 96 May 25 10:10 dir02

-rw-r--r-- 1 login group 0 May 25 10:10 file00

-rw-r--r-- 1 login group 0 May 25 10:10 file01

-rwxr-xr-x 1 login group 9 May 25 10:10 my\_midLS

./dir00:

total 0

drwxr-xr-x 3 login group 96 May 25 10:10 .

drwxr-xr-x 8 login group 256 May 25 10:10 ..

-rw-r--r-- 1 login group 0 May 25 10:10 .gitkeep

./dir01:

total 0

drwxr-xr-x 3 login group 96 May 25 10:10 .

drwxr-xr-x 8 login group 256 May 25 10:10 ..

-rw-r--r-- 1 login group 0 May 25 10:10 .gitkeep

./dir02:

total 0

drwxr-xr-x 3 login group 96 May 25 10:10 .

drwxr-xr-x 8 login group 256 May 25 10:10 ..

-rw-r--r-- 1 login group 0 May 25 10:10 .gitkeep

$>