# Exercises

# wget, easy

create a new folder, and invoke the following command:

```
wget uniprot.org/uniprot/Q640A9.txt
```

What does the file describe?[1]

[1] For reading the file contents, you do not have to type `Q640A9.txt` completely. Use `tab` completion.

# Using commands, medium

The file looks a bit cryptic, isn't it? Now try to leverage the shell to extract significant info from the file.

- use `grep` to find out which organism this file belongs to[2]
- use `tail` to find out the nucleotide sequence in the file
- which parameters did you use to find out the results?
- use `grep` and `|` to find out how many cross-references are in the file[3]

[2] Quote your search terms if they include spaces, e.g. `grep ' ' Q640A9.txt` searches for lines which include a space

[3] look for lines which contain RN

# Creating shell scripts, medium

write a shell script which uses the former commands that you used to extract data. The file to extract the data from must be provided as a parameter. Include other interesting information. In next slide you see an example behavior of the script that you should write.

# Creating shell scripts II, medium

```
$ sh showprotein.sh Q640A9.txt
Analyzed file:
Q640A9.txt

Description:
DE   SubName: Full=Hypothetical LOC496417 {ECO:0000313|EMBL

Organism species:
OS   Xenopus tropicalis (Western clawed frog) (Silurana tro

Literature reference count:
3

Sequence:
SQ   SEQUENCE   220 AA;  25123 MW;  003F4258E81F9B37 CRC64;
     ...
     NIYRTTLNYN CGGSKKWFSS SVQLELCEET SPCIIYTENL
//
```

# Parameters in scripts, medium

Q640A9 is an id from Uniprot protein repository. Edit your script that it only gets any Uniprot id as a parameter, downloads and shows details about the protein. The following command should create the same output from the former exercise:

```
sh showprotein.sh Q640A9
```

# grep, hard

`tail` does not work for all proteins. Invoke your script with the parameter `P25963`. Does it show the whole sequence?

Can you find a better solution than using `grep -A`?

# Shell vocabulary, medium

Find the following commands:

- prints current working directory
- changes the working directory
- shows contents of the folder
- puts the contents of files together and prints them
- shows the first lines of a file
- shows the last lines of a file

# Shell vocabulary, medium II

- shows a file in pages
- deletes files
- copies a file
- moves a file
- creates a new folder
- deletes an empty folder

# Shell vocabulary, medium III

- changes permissions of files
- prints a text
- downloads files
- counts the words, lines, ... in a file
- shows differences between files
- searches for a regular expression in a file/s
- show the user's manual of a command

# Creating shell scripts II, easy

create the script `inspectdir.sh`, which accomplishes following tasks:

- changes the directory to the directory given as parameters
- print the name and lists the content of the directory
- does the same for the directory one level higher

Example:

```
$ sh inspectdir.sh /home/goekce/iti
File listing for
/home/goekce/iti
is
shells.pdf shells-exercises.pdf
Listing for one level higher directory
/home/goekce
is
diary.txt iti
```

# Creating shell scripts III, medium

create the script `archive.sh` which does the following:

- creates a directory named `archive`
- copy the file which is provided as an argument to `archive`
- delete the file which is provided as an argument
- print at every step what is currently being done

Example behavior in next page

# Creating shell scripts III, medium II

```
$ ls
archive.sh test.txt
$ sh archive.sh test.txt
Created the directory archive.
Copied test.txt to archive.
Deleted test.txt
Done!
$ ls
archive.sh archive
$ ls archive
test.txt
```

# Creating shell scripts IV, medium

create the script `countsheep.sh` which searches for the word `sheep` in the given file as an argument

```
$ sh countsheep.sh how-to-sleep.txt
100
```

# Creating shell scripts V, medium

create the script `summary.sh` which takes a file as an argument and does the following:

- prints the first two lines of the file
- prints the last two lines of the file
- prints how many words the file has

Example behavior:

```
$ sh summary.sh long-story.txt
Once upon a time there was a white dove,
which fell in love with a wicked eagle.
If they did not die,
they still live happily ever after.
27218
```