Python course

Shell commands

- `ctrl + l
  - clears the shell output
- `ctrl + d`
  - Ends the input
- `ctrl + c`
  - Interrupts the program
- `man "command"
  - Shows the manual page of a shell command
  - Use space and q to navigate in the manual
- `ls
  - = List
  - Lists files and folders in the current directory
  - options
    - -l prints a long list
- `pwd
  - =print working directory
  - Prints the current folder
- `cd "directory"
  - = change directory
  - Instead of "directory" you can type ".." to move one folder up
- `ln -s "source" "target"
  - = link
  - generates a symbolic link
  - options
    - -s generates a symbolic link. As opposed to a hard link, which we don't need?
    - "source" is the directory the link should point to
    - "target" is the name of the link (optional)
- `rm "file or folder"
  - =remove
- `cat "file"`
  - =catenate
  - shows the contents of the file in the terminal output
- `less`
  - shows content of a file in a paginated way
- `head`
  - shows you the first n (standard 10) lines of a file
  - options

    - -n defines the number of lines to be printed
- `grep "pattern" "file"`
    - maybe stands for global regular expression and print
    - outputs matches of the pattern in the file
    - Example

```
# find GGG in fata
grep GGGG data1.fasta

#sort our fasta data ids
grep id data1.fasta | sort
```

    - options
        - -v print non-matches instead of matches
- `awk`
    - awkward little programming language.
- `cut -f "field number" -d "seperator"`
    - seperates an input according to a "seperator" and puts out the "field number"
- `gzcat`
    - Uncompresses file and writes the content to stdoutput
- `time`
    - times the command follwing
- `cp old_name new_name`
    - = copy
- Pipe symbol |
    - Pushes output from one program to another

Python commands and other stuff

Basic python stuff

Data structures

sets

Sets are grups of items that cannot contain duplicates. It is very fast to check if an item is already in a set.
Example

```
names = set()
names.add("Terry")
```

```
names.add("Eva")
names.add("Eva") # will not be added again
```

import

load a package. You can either load a complete package or a module from a package

```
# import complete package
import Bio
Bio.SeqIO.parse(sys.stdin, "fasta")

# Import one module only
from Bio import SeqIO
SeqIO.parse(sys.stdin, "fasta")
```

loops

for loop

Example:

```
        for line in sys.stdin:
                print(line)
```

You can skip one iteration of a loop you can use the `continue` statement

if statement

Example:

```
    if (line.startswith(">")):
        print(line)
```

in statement

with the in statement it can be checked if an element is in a data structure

```
ids = set()
if line in ids:
        print("Found duplicate: ", line, end="")
else
        ids.add(line)
```

String formatting

f-strings

f-strings can contain variables in curled brackets. They are written with an f before the quotes surrounding the string

```
print(f("Variable 1 has the value: {variable_1}"))
```

sys package

The sys package contains some basic functions to interact with the system

```
import sys
```

sys.stdin

Stands for standard input and is used to receive input from the terminal

Argparse package

A package to accept input from the console

Define argument parser

```
parser = argparse.ArgumentParser(description = "Filter FASTA files")
```

add arguments to the parser

```
parser.add_argument(
        "--translate", action="store_true", help="Translate the sequence to AA"
)
```

read an argument from the parser

```
print("translate: ", args.translate)
```

Provide arguments from the shell

```
python3 script.py --translate
```

This will print "translate True"

Biopython package

SeqIO module

SeqIO.parse(input, format)