UNIVERSITÄT DES SAARLANDES

Basic Tools for NLP

Rishu Kumar, Iuliia Zaitova, Chris Hyek

Saarland University

20.11.2023

Topics for today:

- Bash script organization
- Control flow statements
- Demo
- Hands-on session

my folder : `Downloads`

me : `cd downloads`

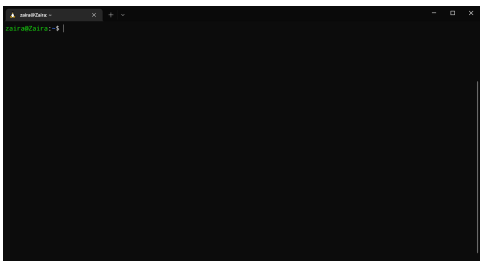
Linux :

it's case-sensitive

Bash scripts

- Shell scripts are plain text files with a ".sh" extension, and they are executed by the shell.

Command-line interface



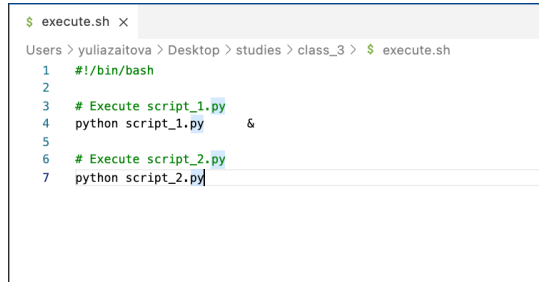
An example shell script

newscript.sh

```
1  #!/bin/bash
2
3  function greeting() {
4    hello="Hello, $name"
5    echo "$hello"
6  }
7
8  echo "Enter name"
9  read name
10
11 val=$(greeting)
12 echo "Return value of the function is $val"
```

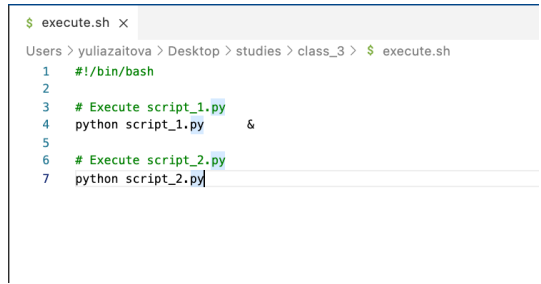
shebang

- It's a line at the beginning, starting with #! followed by the interpreter's path.
  Why?



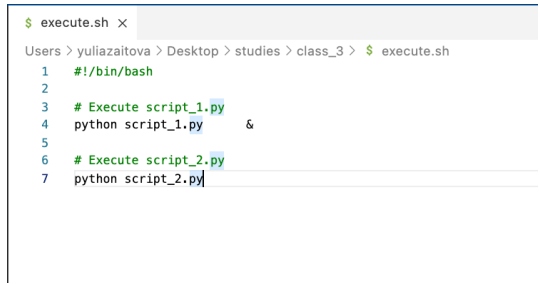
shebang

- It's a line at the beginning, starting with #! followed by the interpreter's path.
  Why?
- Without shebang the shell relies on the system's default behavior for command execution (can be bad)



shebang

- It's a line at the beginning, starting with #! followed by the interpreter's path.
  Why?
- Without shebang the shell relies on the system's default behavior for command execution (can be bad)
- Indicator to other developers and users how the script is intended to be executed



Warm up!

- Download `class_4.zip` from Teams
- Unzip it `unzip class_4.zip`

Passing Arguments to a Script

- Bash scripts can accept arguments from the command line.

```bash
#!/bin/bash

if (( $# != 2));
then
    >&2 echo "Please pass two values seperated by space";
    exit
fi

function greet() {
    echo "Hello, $1! And $2 too!"
}

# Call the function
greet "$1" "$2"
```

Passing Arguments to a Script

- Bash scripts can accept arguments from the command line.
- They can be accessed within the script using special variables.

```bash
#!/bin/bash

if (( $# != 2));
then
    >&2 echo "Please pass two values seperated by space";
    exit
fi

function greet() {
    echo "Hello, $1! And $2 too!"
}

# Call the function
greet "$1" "$2"
```

UNIVERSITÄT DES SAARLANDES

Passing Arguments to a Script

- Bash scripts can accept arguments from the command line.
- They can be accessed within the script using special variables.
- Open `greet.sh` to see :)

```bash
#!/bin/bash

if (( $# != 2));
then
    >&2 echo "Please pass two values seperated by space";
    exit
fi

function greet() {
    echo "Hello, $1! And $2 too!"
}

# Call the function
greet "$1" "$2"
```

UNIVERSITÄT DES SAARLANDES

Passing Arguments to a Script

- Let's run `greet.sh`

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`

```
#!/bin/bash

if (( $# != 2));
then
    >&2 echo "Please pass two values seperated by space";
    exit
fi

function greet() {
    echo "Hello, $1! And $2 too!"
}

# Call the function
greet "$1" "$2"
```

UNIVERSITÄT DES SAARLANDES

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

UNIVERSITÄT DES SAARLANDES

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?
- Why?

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?
- Why?
- `chmod 750 greet.sh`
  `./greet.sh "name1" "name2"`

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?
- Why?
- `chmod 750 greet.sh`
  `./greet.sh "name1" "name2"`
- Try your first name

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?
- Why?
- `chmod 750 greet.sh`
  `./greet.sh "name1" "name2"`
- Try your first name
- Try your full name

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

Passing Arguments to a Script

- Let's run `greet.sh`
- Remember you can do it by `./greet.sh`
- Did it work?
- Why?
- `chmod 750 greet.sh`
  `./greet.sh "name1" "name2"`
- Try your first name
- Try your full name
- Try three names

```
1 #!/bin/bash
2
3 if (( $# != 2));
4 then
5     >&2 echo "Please pass two values seperated by space";
6     exit
7 fi
8
9 function greet() {
10     echo "Hello, $1! And $2 too!"
11 }
12
13 # Call the function
14 greet "$1" "$2"
```

UNIVERSITÄT DES SAARLANDES

Passing Arguments to a Script To quickly go to a specific directory

- `pwd`
- open `~/.bashrc` (or `~/.zshrc`)
- `alias cds="cd <path from pwd command>"`
- `source ~/.bashrc` or `source ~/.zshrc`
- `cds`