Assignment 00

UV Distributed Information Management

Daniel Kocher

Summer semester 2026

Summary:

- **Deadline:** April 05, 2026, 11:59 p.m. (i.e., 23:59) CET.
- **Submission:** Submit a compressed archive (e.g., a `zip` or a `tar.gz` file) that contains your Python3 code (can be the template code for this assignment) and the answers to the questionnaire (can be an empty file for this assignment) via Blackboard.
- **Grading:** 25% Python3 code, 25% answers, and 50% from the after-assignment meeting (cf. Section 5 for details).

---

1 General Remarks

In this assignment, you will learn about the workflow of an assignment and how to submit it via Blackboard [1]. There are no after-assignment meetings for this assignment.

The purpose of this assignment is to briefly look at a Unix-based operating system called *Debian Linux* [2]. You do not neet to write any code yourself but a simple Python3 code will be provided alongside this assignment description and you can submit this code as-is. The same holds for the questionnaire: You can answer the (rather) simple questions, but you can also just include the questionnaire as-is (without answers).

While this assignment will not be graded, Section 5 contains a grading scheme that exemplifies how subsequent assignments will be graded.

Please submit your final Python3 code and the answers to the questionnaire until April 05, 2026, 11:59 p.m. (i.e., 23:59) CET via Blackboard.

1.1 Formatting Conventions

Commands for the Linux command-line tool [3] are written in `TrueType` font [4]. In addition, all commands are in a box that specifies the used command-line tool at the beginning of

[1] Blackboard: `https://elearn.sbg.ac.at`
[2] Debian Linux: `https://en.wikipedia.org/wiki/Debian`
[3] Command-line tool: `https://en.wikipedia.org/wiki/Command-line_interface`
[4] `TrueType` font: `https://en.wikipedia.org/wiki/TrueType`

the title (separated by a dash –, i.e., `terminal` for Linux). The following listing shows an example command that lists all directories within the current directory:

Listing 1: `terminal` – Show directories.

```
ls -l
```

Python3 [5] code is simply wrapped in a box without specifying any command-line tool, and we provide a `.py` file that contains the Python3 code. For this assignment, we assume that all commands are executed using one of the following command-line tools:

– Linux's command-line tool (the so-called `terminal`).

1.2 Support

If anything is unclear or if there is a problem with the submission, please use one of the following communication channels to get help (in this order):

1. **Lecture:** Wednesday 11:30 a.m. - 01:30 p.m. CET (exception: lecture-free periods).
2. **Slack:** `https://dbteaching.slack.com/archives/C08GRUMJ97H` (I will check regularly and do my best to reply fast, but please do not expect me to be available 24/7).
3. **Email:** dkocher@cs.sbg.ac.at (as a last resort).

We recommend to start the assignment early. It is easier for the instructor (and other students) to provide help in time if you identify problems early.

---

2 Assignment Description

Remark: This assignment does not contribute to your grade, but we still exemplify the workflow in the style of an actual assignment using dummy points.

We **highly recommend to read the entire section** (incl. all subsections) before you start working (this should be less error-prone). This assignment has four parts:

1. Set up Debian Linux using VirtualBox, cf. Section 2.2.
2. Familiarize yourself with Debian Linux, cf. Section 2.3.
3. Execute commands in the `terminal` and run a Python3 application, cf. Section 2.4.
4. Answer the questionnaire, cf. Section 2.5.

Follow these steps and refer to the corresponding sections for details. Only parts 1, 3, and 4 contribute to your points on this assignment.

[5] The Python3 Programming language: `https://en.wikipedia.org/wiki/Python_(programming_language)`

2.1 Introduction to (Debian) Linux

Linux is a UNIX-like operating system that is often used for (database) servers, but is also popular among "normal" users as alternative to Microsoft's Windows and Apple's MacOS. At its core, Linux is based on the Linux *kernel* [6], which provides fundamental operating system functionalities. Initially, the Linux kernel has been developed by Linus Torvalds [7] in 1991. There exist many different Linux *distributions* that are based on the Linux kernel but exhibit differences in aspects like visualization, available and pre-installed software, update policy, and so on. In this assignment, we study a specific distribution called *Debian* Linux, which is one of the oldest Linux-based operating systems and the basis for other popular distributions, e.g., Ubuntu [8].

Although most modern Linux distributions have a graphical user interface (GUI) [9], we will use a command-line tool called *terminal* (or *shell*) to execute commands. For example, we can navigate through directories via terminal commands. We will learn some terminal basics in Section 2.3 after we successfully set up Debian Linux in Section 2.2. Afterwards, we go one step further and execute an application (i.e., code) that is written in the Python3 programming language (version 3; cf. Section 2.4).

2.2 (Debian) Linux Setup

In this assignment, we use a *virtual machine (VM)* [10] to set up Debian Linux. A VM can be seen as an application that provides the functionality of a physical computer, i.e., it *virtualizes* the hardware. As a result, we can install *another* operating system (here: Debian Linux) within a VM without compromising our *host system* (i.e., the original operating system that runs the VM as an application, e.g., by starting it on our Windows/MacOS machine). Then, we can start Debian Linux just like any other application. A VM is *sandboxed* [11], i.e., whatever you do *within* the VM has *no effect* on your host system. Therefore, you can play around with the VM and should not be scared of executing commands. In the worst case, you can set up the VM once again (see below). Technically, VMs are quite complex but this simplified viewpoint suffices for now.

Get a Software to Host a VM There are many different ways to install and run a virtual machine. We focus on Oracle's VirtualBox, which is freely available for all common operating systems (including Windows, MacOS, and Linux itself). The first step is to install VirtualBox, e.g., by downloading it from the official website [12] for your system and running the installer (just like for any other software you install).

[6] The Linux kernel: `https://en.wikipedia.org/wiki/Linux_kernel`
[7] Linus Torvalds: `https://en.wikipedia.org/wiki/Linus_Torvalds`
[8] Ubuntu Linux: `https://en.wikipedia.org/wiki/Ubuntu`
[9] Graphical User Interface (GUI): `https://en.wikipedia.org/wiki/Graphical_user_interface`
[10] Virtual machine: `https://en.wikipedia.org/wiki/Full_virtualization`
[11] Sandbox: `https://en.wikipedia.org/wiki/Sandbox_(software_development)`
[12] VirtualBox download: `https://www.virtualbox.org/wiki/Downloads`

Import a VM Image We start VirtualBox and need to import a VM *image* by clicking on *Import* (cf. Figure 1) and choosing an image. In our case, we use a VM image with preinstalled Debian Linux that is available for download via our Nextcloud [13]. Therefore, please download the VM image (about 3-4GB), store it in any location you like, and use it for the import as shown in Figure 2. During the import process, you will be shown the so-called *Appliance Settings* (cf. Figure 3). Most of the time, it suffices to simply continue by clicking on the *Next* button. However, if you experience problems while importing a VM image, please try to uncheck the checkbox for "USB controller" during the import process (cf. Figure 3). Depending on your host system, it will take VirtualBox some time (a few minutes) to import the image, but it should work flawlessly. Then, we have successfully set up our virtual machine with Debian Linux.

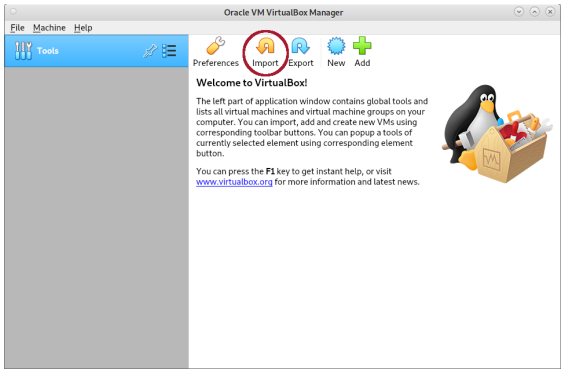

Figure 1: VirtualBox starting screen.

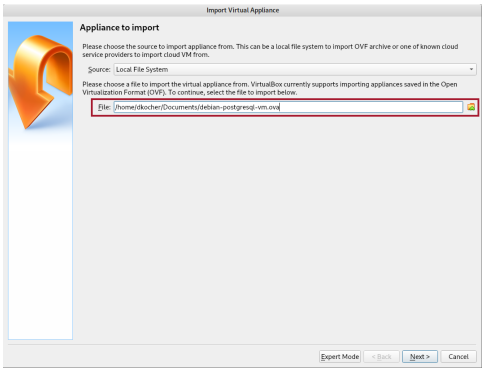

Figure 2: Select a VM image to import.

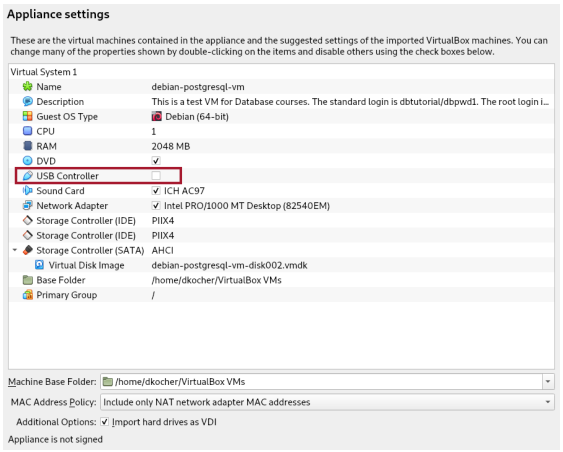

Figure 3: VirtualBox appliance settings.

2.3 Your First Steps in (Debian) Linux

In this section, we discuss the usage of basic Linux commands, i.e., navigation between directories, directory and file creation, and execution of an application (i.e., code).

[13] VM image download link: `https://kitten.cosy.sbg.ac.at/index.php/s/z72cf7n8HFM6tJF`

Start the VM and Login First, we start our VM by double-clicking on the new entry in our VirtualBox (on the left-hand side after successfully importing our image). Then, Debian Linux boots in a new window and we wait for a *Login* screen that asks for username and password. The *credentials* are as follows:

Username: `dbtutorial` **Password:** `dbpwd1`

As a result, we see a Desktop that looks similar to Desktops of other operating systems.

Terminal In order to execute a (pre-defined) command, we need to start a command-line tool. This command-line tool is typically referred to as *terminal* (or *shell*). The application menu in Linux is typically located at the top left, showing three menu items named *Applications*, *Places*, and *System*. In order to start a terminal, we navigate to *Applications* → *System Tools* → *MATE Terminal*, and a white (or black) box pops up with a flashing cursor that waits for your input, prefixed with `dbtutorial@database-tutorial:~$`. For convenience, the Desktop already contains a shortcut symbol to the terminal. We also open the `home` directory of our user by double-clicking on the directory `dbtutorial's Home`, which is the only directory shown on the Desktop. Figure 4 shows the result (without the `pwd` command): A graphical file browser and a terminal that waits for input (side by side).

Navigation/File Browsing Now, we can start using the terminal with three basic commands to browse through your files and directories, namely `pwd`, `ls`, and `cd`. In subsequent assignments, we may use more powerful commands, and a command in the terminal is always *executed with respect to the directory we are currently in* (i.e., the *current directory*). Hence, we need to navigate into a specific directory using our terminal:

`pwd` This command is an abbreviation for *print working directory* [14] and shows the path of the current directory. We use this command to retrieve the current directory.

`ls` This command is an abbreviation for *list* [15] and shows all files and directories that are located in the current directory. This command can be combined with options, e.g., `ls -l` for better readability, or `ls -lah` to show additional information and hidden files/directories. We use this command to see the files and subdirectories (we possibly navigate next) within the current directory.

`cd` This command is an abbreviation for *change directory* [16] and enables us to "move" between directories. We use this command to navigate into other directories.

`pwd` – *Print Working Directory.* First, we want to know the directory we are currently in, the *current directory.* To this end, we enter this command `pwd` into the terminal and hit *Enter* to execute it. Figure 4 shows the result: `/home/dbtutorial`, which is the path of the current directory. There are two directories *levels* that are separated by `/`. The first `/` represents the *root* level, informally the directory that contains *all* other, nested

[14] The `pwd` command: `https://en.wikipedia.org/wiki/Pwd`
[15] The `ls` command: `https://en.wikipedia.org/wiki/Ls`
[16] The `cd` command: `https://en.wikipedia.org/wiki/Cd_(command)`

directories. At the root level, there is a directory named `home`, and the subsequent `/` signals that there is another directory within the `home` directory, specifically a directory named `dbtutorial` (i.e., the home of our user). This directory hierarchy [17] is encoded in the *path*. The right-hand side of Figure 4 visualizes this hierarchy.

Listing 2: `terminal` – Show current directory.

```
1       pwd
```

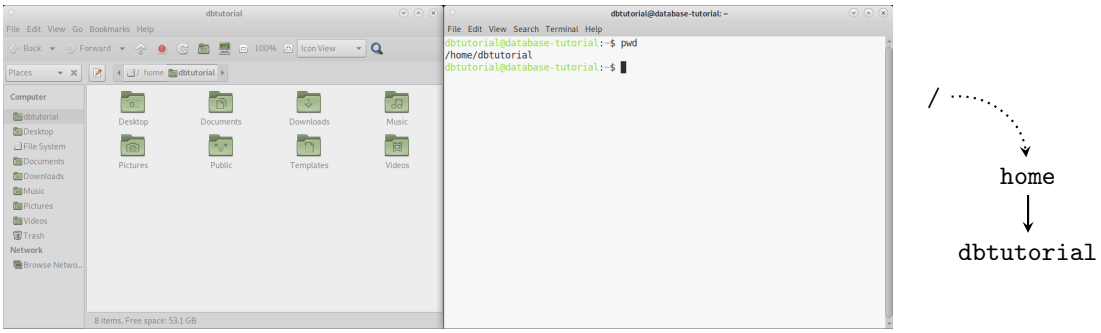

Figure 4: The `pwd` command; showing the current working directory.

Listing 3: `terminal` – Show all files and directories.

```
1       ls -l
```

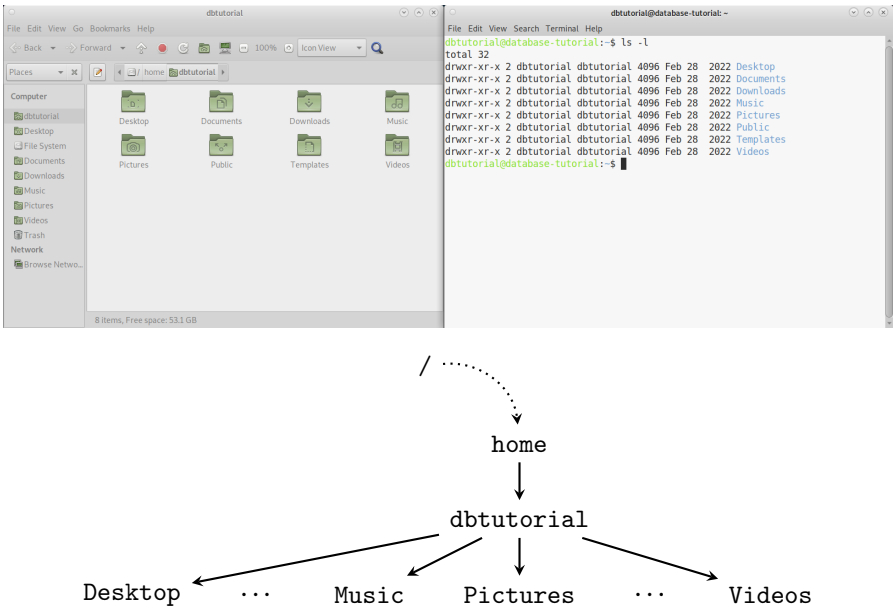

Figure 5: The `ls` command; showing all files and directories.

`ls` – *List*. This command checks the content of the current directory (i.e., `/home/dbtutorial`). We observe that the command lists eight subdirectories, e.g., `Desktop`, `Documents`, `Downloads`,

[17] Hierarchical file system: `https://en.wikipedia.org/wiki/Hierarchical_file_system`

and so on. Moreover, we observe that a command is *executed with respect to the current directory*, i.e., executing `ls` within `/home/dbtutorial` shows a different output than executing `ls` within `/home/dbtutorial/Documents`. Therefore, it is important to know the current directory in order to formulate a command *relative to the current directory.*

`cd` – *Change Directory.* Figure 6 shows the command to actively switch to another directory: `cd`. We can use the information about the subdirectories of the current directory by putting the name of the target subdirectory (to which we want to navigate) after the `cd` command. For example, `cd Documents` switches to the `Documents` subdirectory.

Once again, we used the path `Documents` relative to the current directory. Additionally, we can use an *absolute path*, e.g., by using the `cd` command as shown in the last box of Figure 6. This time, we specify the *full path* starting from the root `/` with all (nested) directories up until our target directory (in this case `Music`).

Listing 4: `terminal` – Move to another directory (relative).

```
1       cd Documents
2       ls -l
```

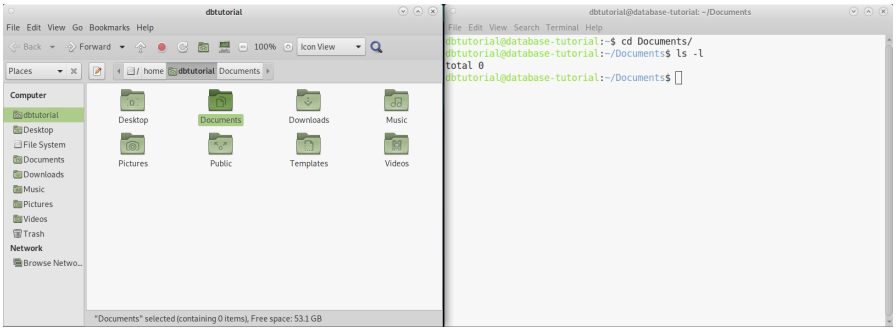

Figure 6: The `cd` command; moving to another directory.

Listing 5: `terminal` – Move to another directory (absolute).

```
1       cd /home/dbtutorial/Music
2       ls -l
```

Finally, most Linux systems typically encode four *shortcuts* that can be used to simplify a command using an absolute and/or a relative path:

`/` Always refers to the entry point of our directory hierarchy. *Example:* `cd /`

`~` Refers to the home directory of the current user, i.e., `/home/dbtutorial` for our `dbtutorial` user. *Example:* `cd ~/Music` is equivalent to `cd /home/dbtutorial/Music`

`..` Refers to the parent directory of the current directory, i.e., the directory one level above in our hierarchical visualization (cf. Figure 5): *Example:* `/home/dbtutorial` is the parent directory of `/home/dbtutorial/Pictures` (cf. Figure 7 for an example).

`.` This shortcut refers to the current directory and is mostly used to execute an application, e.g., the Python3 code of our assignment. *Example:* `cd ./Documents`

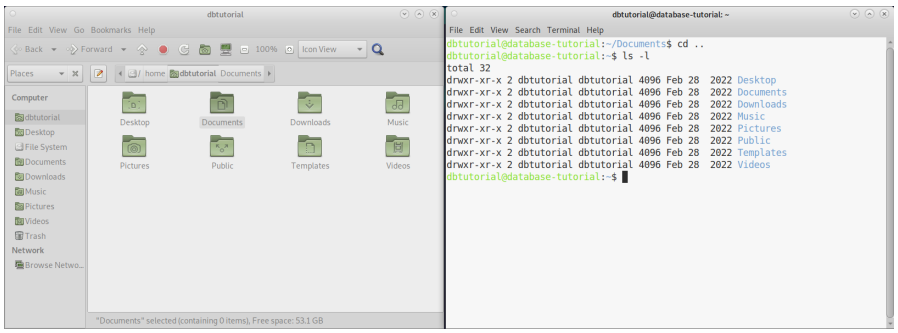

Figure 7: The .. shortcut; moving to the parent directory.

Listing 6: `terminal` – Move to the parent directory.

```
cd ..
ls -l
```

Directory and File Creation Now that we can navigate within our system, we can create new directories and files. We can do this using our graphical file browser (right click) or using our terminal. This can be achieved with the `mkdir` command and the name of the subdirectory we want to create. Importantly, this subdirectory will be created relative to our current directory. Figure 8 shows an example for a subdirectory named `myDirectory` that is created within the `home` directory. We confirm the creation using the `ls` command and navigating into it using the `cd` command (as shown in Figure 9).

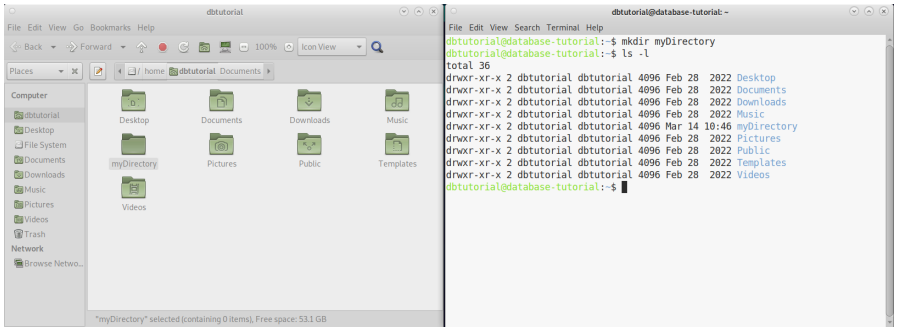

Figure 8: The `mkdir` command; creating a new directory.

Listing 7: `terminal` – Create a new directory.

```
mkdir myDirectory
ls -l
```

For completeness, Figure 10 shows a screenshot of the graphical file dialogue to create a new file in our new directory: *Create Document* → *Empty File*. The system will then ask for a filename, e.g., `assignment0.py` (`.py` is the standard file extension for Python3 code). We typically also want to edit this file, e.g., to modify our code. Basically, we can use any editor to do this and Debian has the `Pluma` editor pre-installed (cf. Figure 11).

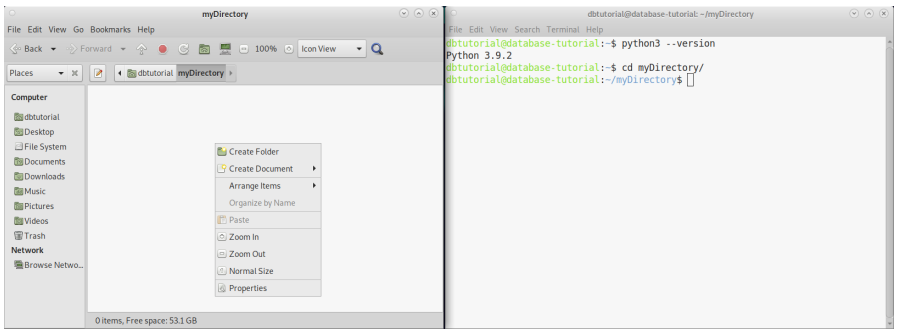

Figure 9: Moving to the newly created directory `/home/dbtutorial/myDirectory`.

Listing 8: `terminal` – Move to the new directory and show its content.

```
1    cd myDirectory
```

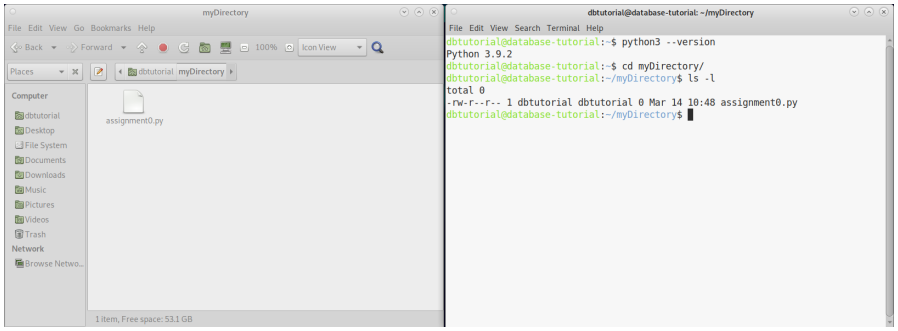

Figure 10: Showing the content of the new directory, i.e., `assignment0.py`.

Listing 9: `terminal` – Show content of the new directory.

```
1    ls -l
```

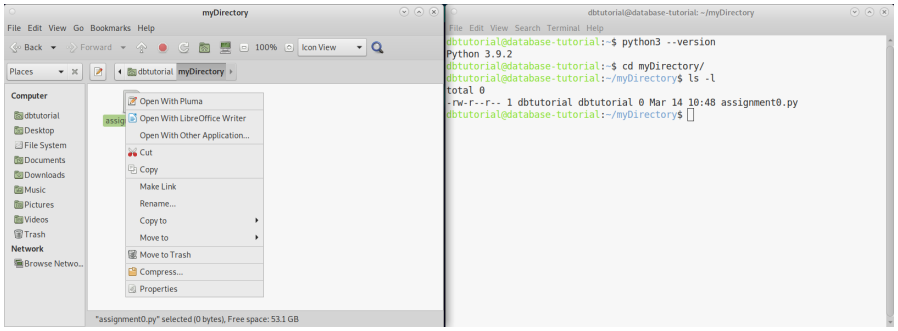

Figure 11: Editing a file with `Pluma`.

2.4 Execution of Python3 Code

Before we can execute the Python3 template code of this assignment, we download the code from the course website using the widely used Linux tool `wget`. The following

command downloads the Python3 template file to the new directory `myDirectory`:

Listing 10: `terminal` – Download Python3 template code from course website.

```
1   wget https://dbresearch.uni-salzburg.at/teaching/2024ss/dim/assignment0-template.py -O
assignment0.py
```

Figure 12 shows `Pluma` with the Python3 template code for this assignment.

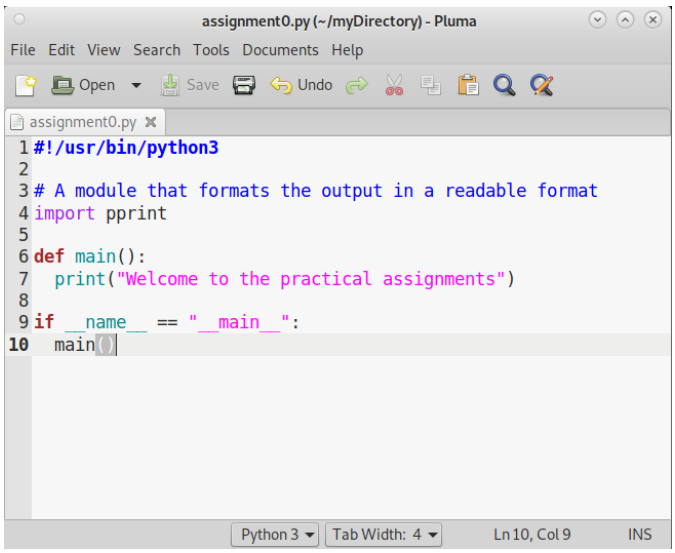

Figure 12: Template Python3 code of this assignment in `Pluma`.

Remark: If you cannot execute the Python3 template code (for whatever reason), please contact the instructor *before* the submission. We will find a reasonable solution together.

After saving the Python3 template code, we now want to execute this code using the terminal. To this end, we first execute the `ls -l` command to study the permissions of our file; the corresponding output is shown in Figure 13. Importantly, the command `ls -l` reports the file permissions on the left-hand side:

`-rw-r-r-`

Each color-coded triple, `rw-`, `r-`, and `r-`, denotes the permissions for a specific user category and encodes the permission to *read*, *write*, and *execute* a file (or directory). For our purpose, it suffices to know that the first triple (highlighted in red) denotes the permissions of the `dbtutorial` user for the file `assignment0.py`, i.e., `rw-`. By now, our user is only allowed to read (`r`) and write (`w`) this file, but not to execute it (signaled by `-`). In order to execute our Python3 code, we first need to modify these permissions to include the permission to execute (`x`) this file. This can be achieved using the `chmod` command [18], which is an abbreviation for *change file mode bits*. Since we want to allow (**+**) our **u**ser to **e**xecute our Python3 code located in the file `assignment0.py`, we call the `chmod` command with `u+x` and the file we want to change, i.e., `assignment0.py`. Figure 13 exemplifies this command together with the effect shown by a call to `ls -l`, which now shows `rwx`.

[18] The `chmod` command: `https://en.wikipedia.org/wiki/Chmod`

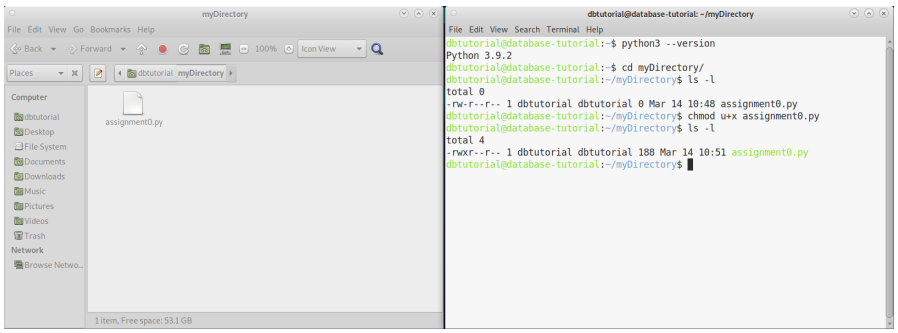

Figure 13: Modifying the permissions to execute our Python3 code.

Listing 11: `terminal` – Allow Python3 code to execute.

```
chmod u+x assignment0.py
```

Once we have all necessary permissions, we can execute our Python3 code as shown in Figure 14: We use the `python3` command in combination with the file that we want to execute, i.e., `assignment0.py`. If everything works as expected, we see the following result of the Python3 code written to the terminal: `Welcome to the practical assignments`

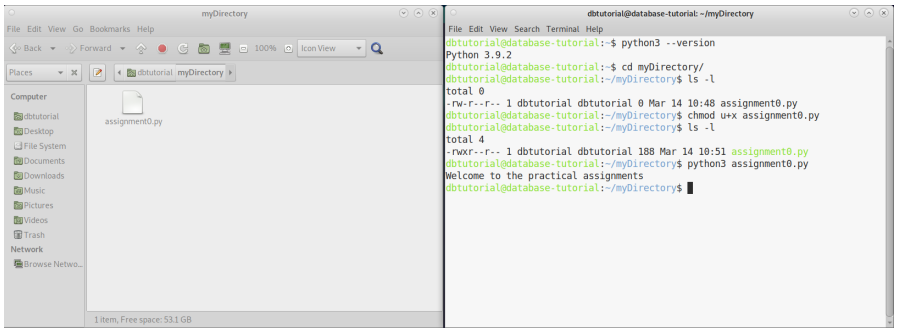

Figure 14: Executing our Python3 code.

Listing 12: `terminal` – Execute the Python3 code.

```
python3 assignment0.py
```

Template Code

Listing 1: Dummy Python3 code with an unused import that may be useful in the future.

```
#!/usr/bin/python3

# A module that formats the output in a readable format
import pprint

def main():
  print("Welcome to the practical assignments")

if __name__ == "__main__":
  main()
```

2.5 Questionnaire

The questionnaire contains questions about this assignment in a separate text file called `assignment0-questionnaire.txt`, which are discussed in the after-assignment meetings.

---

3 Submission

Please submit a single compressed archive (e.g., `.zip` or `.tar.gz`) that contains two files: (a) Your Python3 code (for this assignment: the template code) and (b) the answers to the questionnaire (for this assignment: the empty questionnaire).

Code Please submit a Python3 file (`assignment0.py`) that contains the full code for this assignment. The code must print the welcome message when executed *as submitted*. We will not debug your code, for example, change some variable to make it work. Therefore, please double-check that your Python3 code works as expected.

Questionnaire Answer the questions directly in the file `assignment0-questionnaire.txt`. If you prefer to use a different application to answer the questions (e.g., Microsoft Word and the likes), you must use one of the following formats: `.txt`, `.pdf`, `.odt`, or `.docx`.

---

4 Supplementary Material

This section provides pointers to material that may be helpful to solve the assignment.

- One of the many introductions to the Linux terminal: `https://www.digitalocean.com/community/tutorials/an-introduction-to-the-linux-terminal`

---

5 Grading

For transparency, this section provides a detailed grading scheme for this assignment, i.e., which part contributes how many points to the total number of 20 points.

Code The code contributes at most 5 points and is evaluated based on the following criteria (if the code is executed as submitted; disregarding the credentials):

Max. Points	Criterion
1	The word *Welcome* is correctly printed to the command line (in order).
1	The word *to* is correctly printed to the command line (in order).
1	The word *the* is correctly printed to the command line (in order).
1	The word *practical* is correctly printed to the command line (in order).
1	The word *assignments* is correctly printed to the command line (in order).
5	

Questionnaire The questionnaire contributes at most 5 points and is evaluated based on the following criteria:

Max. Points	Criterion
1.25	Correctness of answer A1.
1.25	Correctness of answer A2.
1.25	Correctness of answer A3.
1.25	Correctness of answer A4.
5	

After-Assignment Meeting The discussion in the after-assignment meeting contributes at most 10 points, which are awarded based on assignment-specific questions.