UV Distributed Information Management
Summer semester 2023

Assignment 00

Summary:

    Deadline: March 26, 2023, 11:55 pm (aka 23:55) CET.

    Extended Deadline: None (as this is a test assignment).

    Submission: Submit a compressed archive (e.g., a `zip` or a `tar.gz` file) that contains your Python3 code (can be a copy of the given code for this assignment) and the answers to the questionnaire (can also be an empty file for this assignment) via Blackboard.

    Grading: 55% Python3 code, 45% answers (incl. meeting; cf. Section 5 for details).

---

1 General Remarks

In the course of this assignment, we will learn about the workflow of an assignment and how to submit it via Blackboard. Furthermore, we will have a test run of an after-assignment meeting and briefly look at a Unix-based operating system called *Debian Linux*[1]. For this assignment, you are not required 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).

Please submit your final Python code and the answers to the questionnaire until March 26, 2023, 11:55 pm (aka 23:55) CET via Blackboard [2] (no late submission for this test assignment).

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

```
Listing 1: terminal – Show directories.
1    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 (in order to make using the template easier).

---

[1]Debian Linux: `https://en.wikipedia.org/wiki/Debian`
[2]Blackboard: `https://elearn.sbg.ac.at`
[3]Command-line tool: `https://en.wikipedia.org/wiki/Command-line_interface`
[4]TrueType font: `https://en.wikipedia.org/wiki/TrueType`
[5]The Python Programming language: `https://en.wikipedia.org/wiki/Python_(programming_language)`

For the remainder of this assignment description, we assume that the commands are executed using one of the following command-line tools:

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

1.2 Support

Remark: Please notify the instructor as soon as possible if the assignment description is (partially) unclear or if there is a problem with the submission.

If you have troubles understanding the assignment, please use one of the following communication channels to get help (in this order):

1. **Lecture:** mondays 10:15 am - 12:00 pm CET, wednesdays 01:00 - 02:30 pm CET (exception: lecture-free periods).

2. **Slack:** `https://dbteaching.slack.com/archives/C04QHH4TR7B` (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).

Typically, we recommend to start the assignments early. In case of a problem, it is easier for the instructor and other students to provide help in time if you identify the problem early.

2 Assignment Description

Remark: This assignment does not contribute to your grade in any regard (i.e., you are not awarded points). However, we still exemplify the workflow of an assignment by splitting it as if it was a real assignment, and as if each part contributes a particular number of dummy points.

We **highly recommend** that you read this section (including all the subsections) to the end before you start working (this should be less error-prone). We split this assignment into 4 parts:

1. Set up Debian Linux using VirtualBox, cf. Section 2.2.

2. Familiarize yourself with Debian Linux and the command-line tool that is typically referred to as `terminal`, cf. Section 2.3.

3. Try to execute some example commands using the `terminal` and to run the given Python3 application, cf. Section 2.4.

4. Answer the questionnaire, cf. Section 2.5.

Only parts 1, 3, and 4 contribute to the overall grade of this assignment. We recommend to follow these steps and refer to the corresponding sections for details. In addition, we briefly introduce Linux as an operating system and Debian as one specific variant of Linux in Section 2.1.

2.1 Introduction to (Debian) Linux

In contrast to Windows, Linux is a UNIX-like operating system that is often used for (database) servers, but is also popular among "normal" user as alternative to Windows and MacOS. At its core, Linux is based on the Linux *kernel*[6], which has been developed by Linus Torvalds[7] in 1991 and provides core functionality for operating systems. In general, there exist many

---

[6] The Linux kernel: `https://en.wikipedia.org/wiki/Linux_kernel`
[7] Linus Torvalds: `https://en.wikipedia.org/wiki/Linus_Torvalds`

different so-called Linux *distributions* that are all based on Linux but exhibit differences in aspects like visualization, available and pre-installed software, update policy, and the likes. In this assignment, we will briefly study a specific distribution called *Debian* Linux, which is one of the oldest Linux-based operating systems and the basis for other distributions, e.g., Ubuntu[8].

Although most modern Linux distributions have a graphical user interface (GUI)[9], we typically use a command-line tool called *terminal* or *shell* to interact with Linux by executing commands, e.g., to navigate through our directories. We will learn some basics in Section 2.3 after we successfully set up Debian Linux in Section 2.2. Once we know the basics, we can go one step further and execute a code that is written in the Python programming language (cf. Section 2.4).

2.2 Setting up (Debian) Linux

In this assignment, we will use a so-called *virtual machine (VM)*[10] to set up Debian Linux. A virtual machine can be seen as an application that provides the functionality of a physical computer – it *virtualizes* the hardware. As a result, we can install another operating system (in our case Debian Linux) within such a virtual machine without changing our *host system* (i.e., our original operating system we run the virtual machine as application on, e.g., our Windows instance). As a result, we can start Debian Linux just like any other application. In general, a virtual machine is a so-called *sandbox*[11], meaning that whatever you do *within* the virtual machine has no effect on your host system. Consequently, you can play around with the virtual machine and, in the worst case, you can delete it and set it up again (as described below). Virtual machines are quite complex but this simplified viewpoint suffices for now.

Getting 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, Linux, and MacOS). 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).

Importing a VM Image   Then, we start VirtualBox and need to import a virtual machine *image* by clicking on *Import* (cf. Figure 1) and choosing an image. In our case, we use a given image that already includes Debian Linux and is available for download via our Nextcloud[13]. Therefore, please download the 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 during the import process, please try to uncheck the checkbox for "USB" during the import process. 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.

2.3 First Steps in (Debian) Linux

In this section, we will briefly discuss and showcase the usage of some basic commands to interact with Debian Linux. This includes the navigation between directories, creating new directories and files, and executing Python3 code.

---

[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

[13]The download link for our VM image: https://kitten.cosy.sbg.ac.at/index.php/s/AeBcR5kMAApm33L
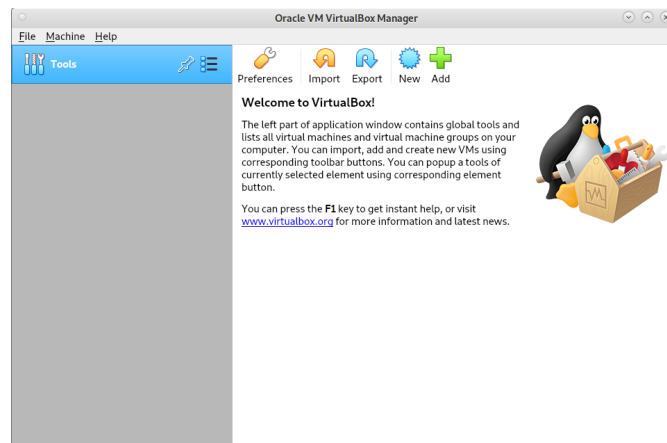
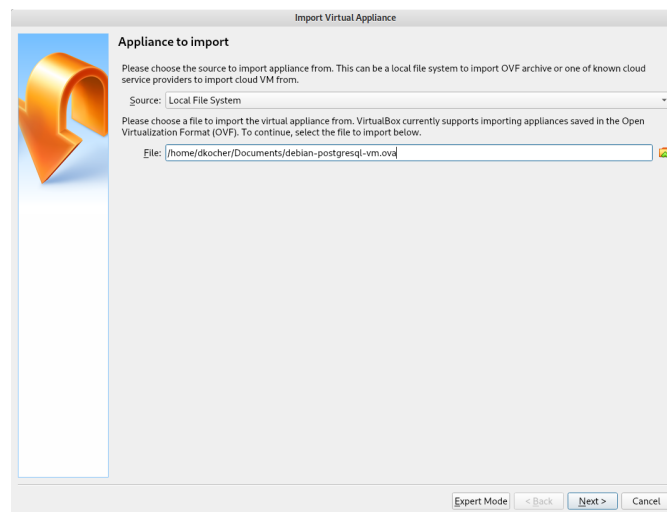Figure 1: VirtualBox starting screen.

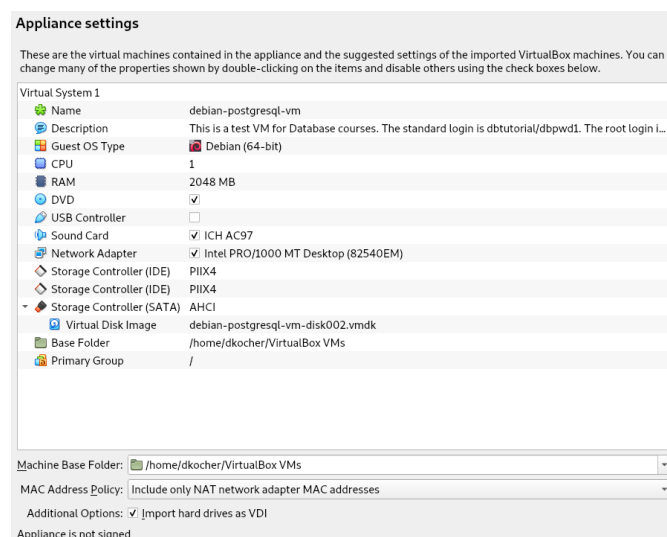Figure 2: Selecting the image to be imported into VirtualBox.

Figure 3: VirtualBox appliance settings.

Starting the VM and Login   First of all, we need to run our virtual machine by double-clicking on the new entry in our VirtualBox (on the left-hand side after successfully importing our image). Then, a new window will open and boot Debian Linux; nothing to do until we see a *Login* screen that asks for username and password. The credentials are:

<div align="center">

Username: dbtutorial   **Password:** dbpwd1

</div>

Then, you see a screen that looks similar to a normal Desktop in other operating systems.

Terminal   In order to execute a (pre-defined) command, we need to start a command-line tool that is capable of doing so. This command-line tool is typically referred to as *terminal* or *shell*. In contrast to other operating systems, the application menu in Linux is typically located at the top left, saying *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:~$. Before we start using 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 at the Desktop. As a result, you should now see something similar to what is shown in Figure 4 (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 our terminal. In this paragraph, we will study three basic commands to browse through your files and directories in the terminal (instead of the graphical file browser), namely pwd, ls, and cd. In subsequent assignments, we may want to use more powerful commands *within* specific directories. Therefore, we need to know how to navigate to a specific directory using our terminal as we typically cannot execute commands in the graphical file browser. Summary of navigation commands:

pwd This command is an abbreviation for *print working directory*[14] and shows the path of the current directory. We use this command to know our current location within our system.

ls This command is an abbreviation for *list*[15] and shows all files and directories that are located in the directory we are currently in. This command can be used with the -l option: ls -l (for better readability), or with the -lah option: ls -lah (for additional information and hidden files/directories). We use this command to find out which files we may open or in which directories we can navigate next.

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

First, we want to know the directory we are currently in. To this end, we type the command pwd into the terminal and hit *Enter* to execute it; Figure 4 shows the result. We get /home/dbtutorial as response, 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 directories. At this first level, there is a directory called home, and the subsequent / signals that there is another directory within the home directory, namely a directory called dbtutorial (i.e., the home of our user). This hierarchy of directories[17] is encoded in the so-called *path*, which is the response to our pwd command. A visualization of this (rather simple)
Figure 6 showcases the command to actively switch to another directory, namely the cd command. We know the subdirectories of our current directory and we can use this information:

---

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

we put the name of the target subdirectory (to which we want to switch) after the `cd` command. For example, `cd Documents` switches to the `Documents` subdirectory. Afterwards, we can check the content of our *new* current directory (i.e., `Documents`) using the `ls` command again. In this case, the directory is empty and we make an important observation: each single command is *executed with respect to the current directory*, i.e., `ls` executed in `/home/dbtutorial` shows a different output than `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. Alternatively, we can formulate a command using an *absolute path*. For example, we can adapt the `cd` command as shown in the last box of Figure 6. This time, we specify the full path starting from the root `/` and containing all directories until our target directory (in this case `Music`).

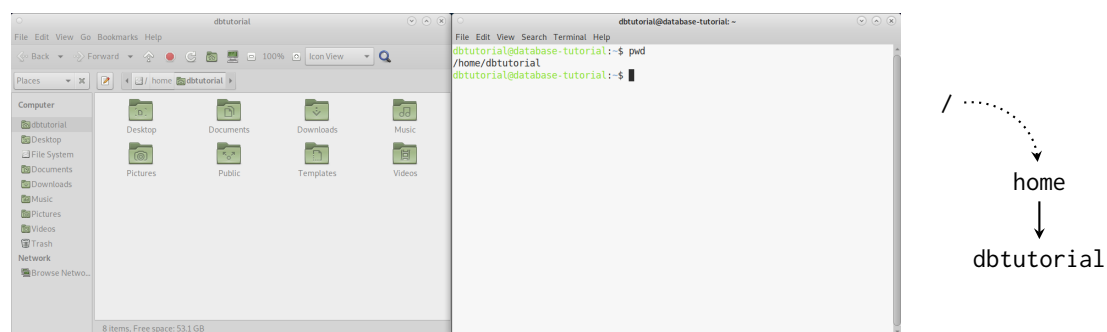Listing 2: `terminal` – Show current directory.

```
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.

Finally, most Linux systems typically encode 4 *shortcuts* that can be used to simplify a command:

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

~   This shortcut refers to the home directory of the current user, i.e., `/home/dbtutorial` for our `dbtutorial` user. *Example:* `cd ~`

..   This is a shortcut for the parent directory, i.e, the directory one level above in our hierarchical visualization: For example, the directory `/home/dbtutorial` is the parent directory of `/home/dbtutorial/Desktop`, `/home/dbtutorial/Pictures`, and so on (cf. Figure 5). Figure 7 shows an example of its usage.

.   This shortcut refers to the current directory and is mostly used when we want to execute a program, e.g., the Python code of our assignment. *Example:* `cd ./Documents`.

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

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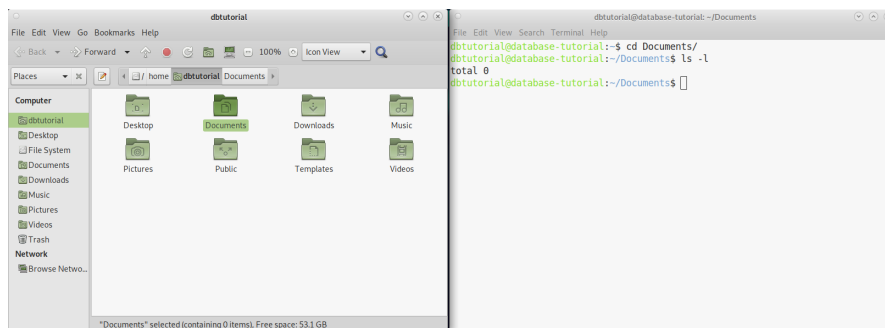

Listing 5: `terminal` – Move to another directory (absolute).

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

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

Listing 6: `terminal` – Move to the parent directory.
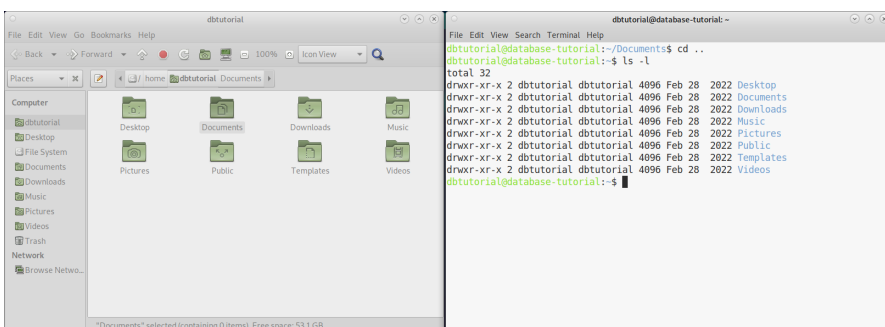
```
1    cd ..
2    ls -l
```


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

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

Figure 10 also contains a screenshot of the file dialogue to create a new, empty file in our new directory: *Create Document → Empty File*. The system will then ask us to give our new, empty file a name, for example, `assignment0.py` (with `.py` being the standard file extension for code written in the Python3 programming language). We typically also want to edit this file, e.g., we want to add or modify our Python3 code. Basically, we can use any editor to do this; in Debian, an editor called `Pluma` is pre-installed and can be used as shown in Figure 11: *Open With Pluma*. Figure 12 shows `Pluma` with the Python3 code that is given as template code for this assignment.

Listing 7: `terminal` – Create a new directory.
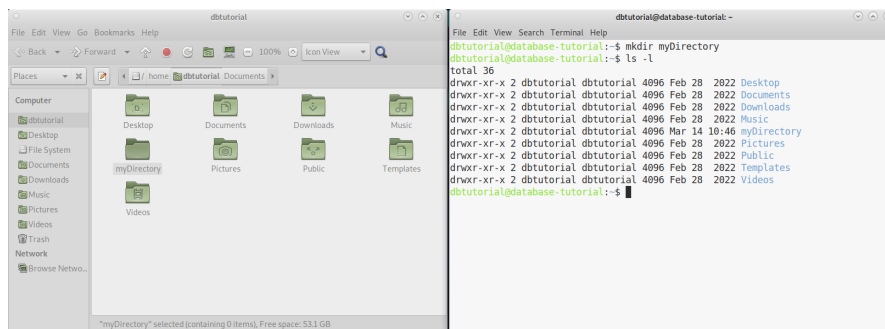
```
mkdir myDirectory
ls -l
```


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

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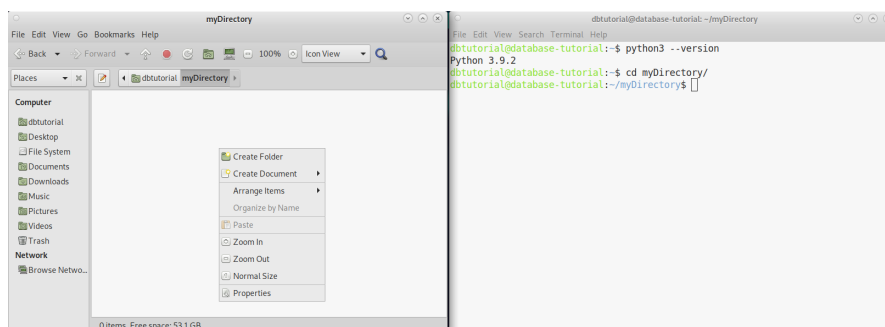
```
cd myDirectory
```


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

> **Listing 9:** `terminal` – Show content of the new directory.
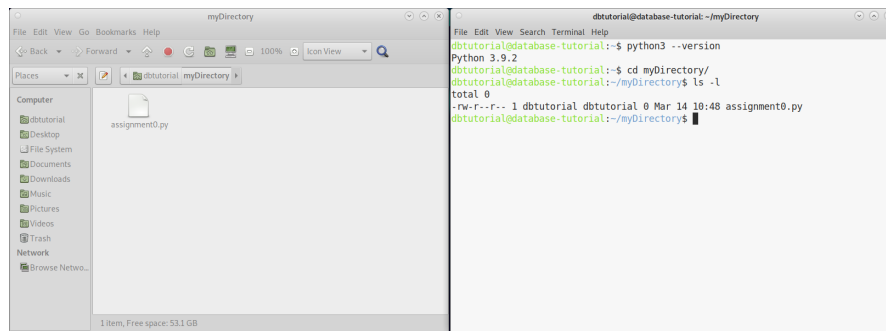>
> ```
> 1    ls -l
> ```


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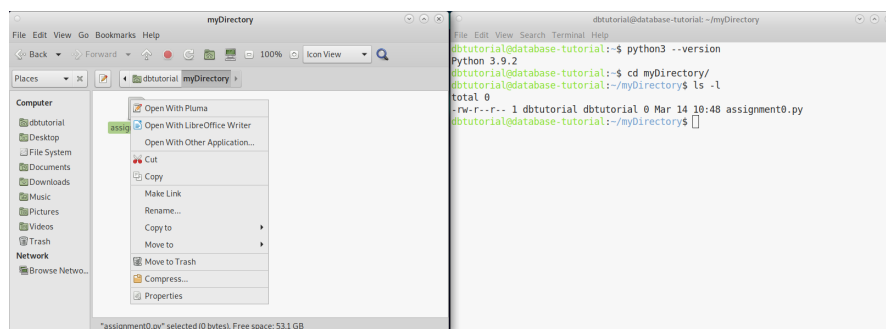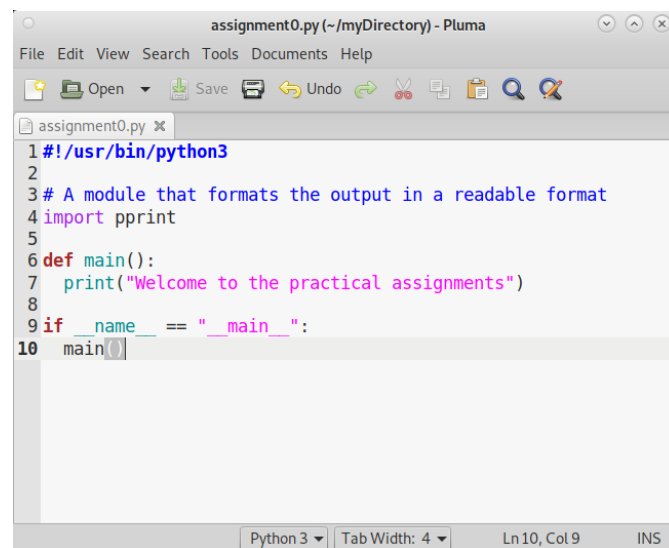

Figure 11: Editing a file with `Pluma`.

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

2.4 Execution of Python Code

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

After saving our 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. The `-l` option of `ls` provides additional information, most importantly the file permissions on the left-hand side:

<p align="center"><code>-rw-r--r--</code></p>

Each color-coded triple, `rw-`, `r--`, and `r--`, denotes the permissions for a specific group of users 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), `rw-`, denotes the permissions of our user dbtutorial for the file assignment0.py. By now, our user is only allowed to read (`r`) and write (`w`) this file, but has no permission to execute it (signaled by the `-`). 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 with the chmod commands, which is an abbreviation for _change file mode bits_. Since we want to allow (**+**) our **u**ser to e**x**ecute our Python3 code located in the file assignment0.py, we call the chmod command with u+x and the file we want to change it for, i.e., assignment0.py. Figure 13 exemplifies this command together with the effect shown by a subsequent call to `ls -l`, which now shows `rwx`.

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 then observe that the execution of our Python3 code results in one line that is printed to the terminal:

<p align="center"><code>Welcome to the practical assignments</code></p>

Listing 10: `terminal` – Allow Python code to execute.
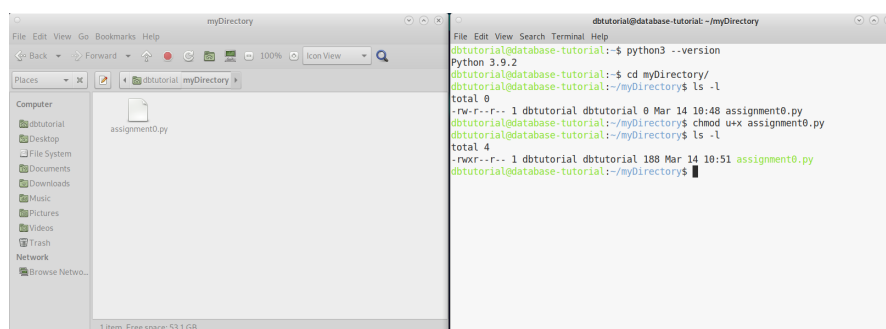
```
1    chmod u+x assignment0.py
```

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

Template Code

Remark: We also provide the template code in a separate file assignment0-template.py (to simplify the otherwise cumbersome process of copying the template from this PDF).

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

```python
1   #!/usr/bin/python3
2
3   # A module that formats the output in a readable format
4   import pprint
5
6   def main():
7     print("Welcome to the practical assignments")
8
9   if __name__ == "__main__":
10    main()
```
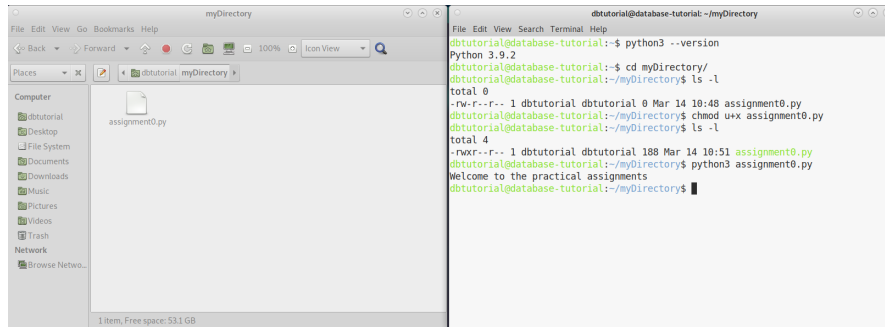
```
1    python3 assignment0.py
```


Figure 14: Executing our Python3 code.

2.5 Questionnaire

The questionnaire contains questions about this assignment. These questions are potentially discussed during the after-assignment meetings. The questionnaire can be found in a separate text file called `assignment0-questionnaire.txt`.

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 provided code can be submitted as-is) and (b) the answers to the questionnaire (for this assignment, the questionnaire can be submitted as-is).

Code   Please submit a single Python3 file (`.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

Remark: The recommended formats are `.txt` and `.pdf`.

You can answer the questions directly in the text file `assignment0-questionnaire.txt`. If you prefer to use a different application to answer the questions (e.g., Microsoft Word and the likes), you are welcome to do so. In any case, the submitted file must be in one of the following formats: `.txt`, `.pdf`, `.odt`, `.doc`, or `.docx`.

4 Supplementary Material

This section provides a list of 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 the sake of transparency, this section provides more details on the grading of this assignment, i.e., which part contributes how many points to the total number of 18 points.

Code The code contributes at most 10 points and is evaluated based on the following criteria (if the code is executed as submitted; disregarding the credentials):

Max. Points	Criterion
2	The word *Welcome* is correctly printed to the command line (in order).
2	The word *to* is correctly printed to the command line (in order).
2	The word *the* is correctly printed to the command line (in order).
2	The word *practical* is correctly printed to the command line (in order).
2	The word *assignments* is correctly printed to the command line (in order).
10	

Questionnaire The questionnaire contributes at most 8 points and is evaluated based on the following criteria (taking the discussion during the after-assignment meeting into account):

Max. Points	Criterion
2	Correctness of answer A1.
2	Correctness of answer A2.
2	Correctness of answer A3.
2	Correctness of answer A4.
8	