

UV Distributed Information Management

Summer semester 2024

Assignment 00

Summary:

Deadline: March 15, 2024, 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: 50% Python3 code, 50% 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 an optional test run of an after-assignment meeting or alternatively, three example question-answer pairs will be provided to you via Blackboard ¹ after the assignment deadline.

The purpose of this test assignment is to briefly look at a Unix-based operating system called *Debian Linux* ². 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).

For your convenience, Section 5 contains a grading scheme that exemplifies how subsequent assignments will be graded. However, this test assignment will not be graded whatsoever.

Please submit your final Python code and the answers to the questionnaire until March 15, 2024, 11:55 pm (aka 23:55) CET via Blackboard (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:

¹Blackboard: <https://elearn.sbg.ac.at>

²Debian Linux: <https://en.wikipedia.org/wiki/Debian>

³Command-line tool: https://en.wikipedia.org/wiki/Command-line_interface

⁴TrueType font: <https://en.wikipedia.org/wiki/TrueType>

Listing 1: terminal – Show directories.

```
1 ls -l
```

Python3⁵ 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).

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:** wednesdays 01:00 - 02:30 pm CET (exception: lecture-free periods).
2. **Slack:** <https://dbteaching.slack.com/archives/C06MFP830GH> (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 four 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.

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

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 fundamental operating system functionalities. In general, there exist many 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. For example, we can navigate through our 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. Once we know the basics, we can go one step further and execute a program (i.e., code) that is written in the Python programming language (version 3; cf. Section 2.4).

2.2 (Debian) Linux Setup

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., the original operating system that runs the virtual machine as application, e.g., by starting it on our Windows or MacOS machine). Thus, 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 should not be scared of breaking the virtual machine. In the worst case, you can delete the virtual machine from your host system and set it up once again (see below). Virtual machines 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).

Import 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 while importing a VM image, please try to

⁶The Linux kernel: https://en.wikipedia.org/wiki/Linux_kernel

⁷Linus Torvalds: https://en.wikipedia.org/wiki/Linus_Torvalds

⁸Ubuntu Linux: <https://en.wikipedia.org/wiki/Ubuntu>

⁹Graphical User Interface (GUI): https://en.wikipedia.org/wiki/Graphical_user_interface

¹⁰Virtual machine: https://en.wikipedia.org/wiki/Full_virtualization

¹¹Sandbox: [https://en.wikipedia.org/wiki/Sandbox_\(software_development\)](https://en.wikipedia.org/wiki/Sandbox_(software_development))

¹²VirtualBox download: <https://www.virtualbox.org/wiki/Downloads>

¹³The download link for our VM image: <https://kitten.cosy.sbg.ac.at/index.php/s/Hp5YAY47izLKkaT>

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.

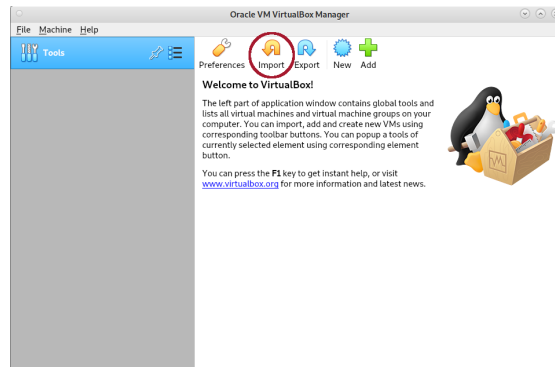


Figure 1: VirtualBox starting screen.

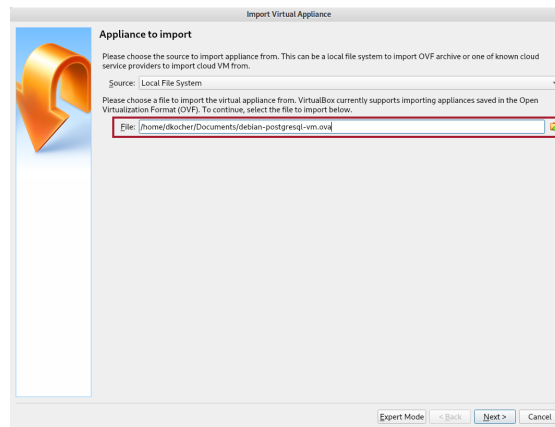


Figure 2: Selecting the image to be imported into VirtualBox.

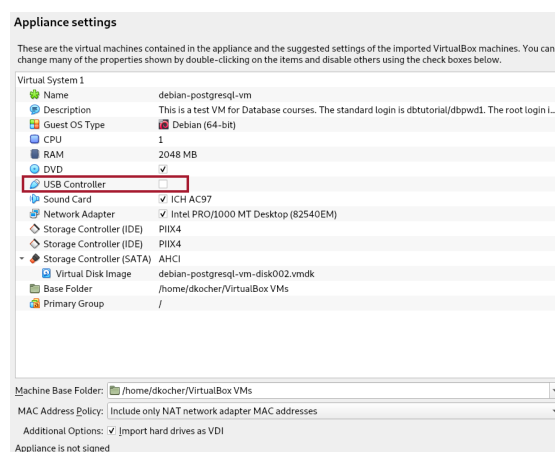


Figure 3: VirtualBox appliance settings.

2.3 Your 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 a Python3 application (i.e., code).

Start 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 as follows:

Username: dbtutorial **Password:** dbpwd1

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, 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 your convenience, the Desktop already contains a shortcut to the terminal. 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 on 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 see the files within our current directory, or to see the subdirectories 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 into other directories.

pwd – Print Working Directory. 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 this command; Figure 4 shows the result. As a response, we get `/home/dbtutorial`, which is the path to the current directory. There are two directories *levels* that are separated by `/`. The first

¹⁴The `pwd` command: <https://en.wikipedia.org/wiki/Pwd>

¹⁵The `ls` command: <https://en.wikipedia.org/wiki/Ls>

¹⁶The `cd` command: [https://en.wikipedia.org/wiki/Cd_\(command\)](https://en.wikipedia.org/wiki/Cd_(command))

Listing 3: terminal – Show all files and directories.

```
1  ls -l
```

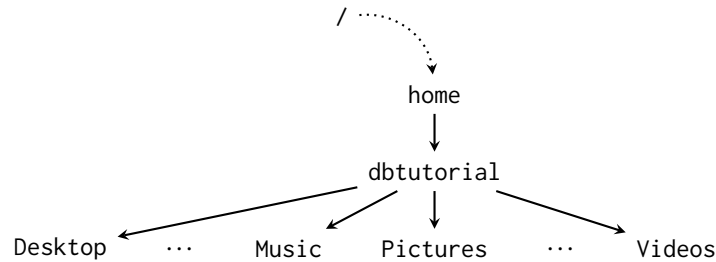
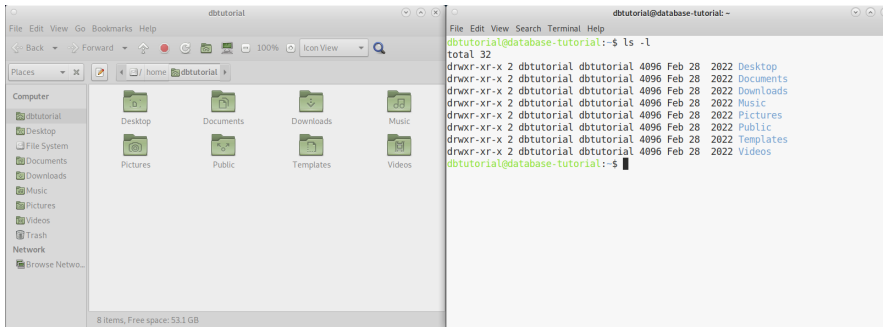
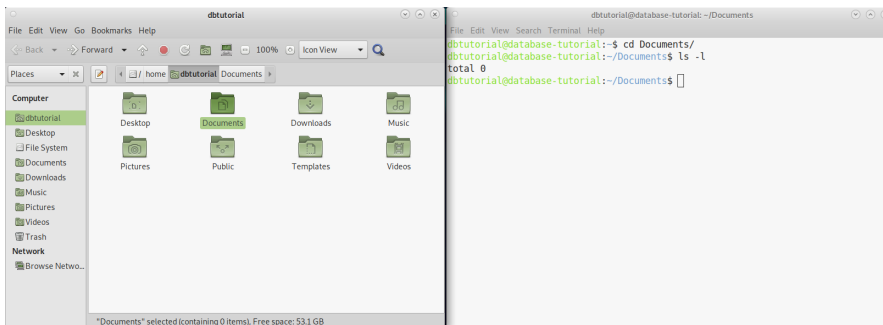


Figure 5: The ls command; showing all files and directories.

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.

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

Listing 6: terminal – Move to the parent directory.

```
1 cd ..
2 ls -l
```

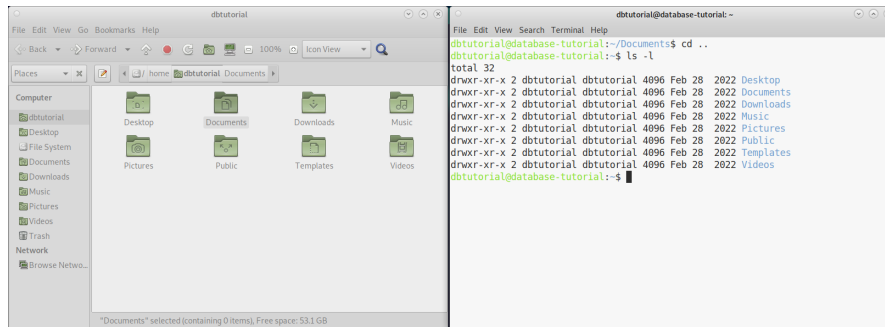


Figure 7: The `..` shortcut; moving to the parent directory.

want to create. Importantly, this subdirectory will be created relative to our current directory, i.e., within the directory we are currently in. 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).

For the sake of completeness, Figure 10 shows a screenshot of the graphical 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-template.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 before executing it. 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. For your convenience, there is a directory named “assignment-00” (in `/home/dbtutorial/Desktop`), which already contains the Python3 code. As part of this assignment, find out how to navigate into this specific directory.

Listing 7: terminal – Create a new directory.

```
1 mkdir myDirectory
2 ls -l
```

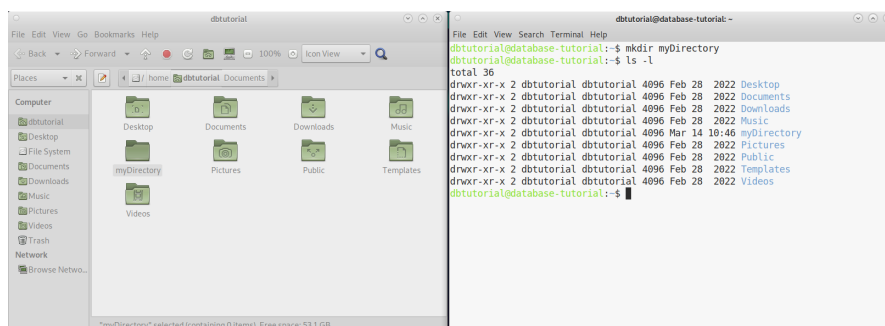


Figure 8: The `mkdir` command; creating a new directory.

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

```
1 cd myDirectory
```

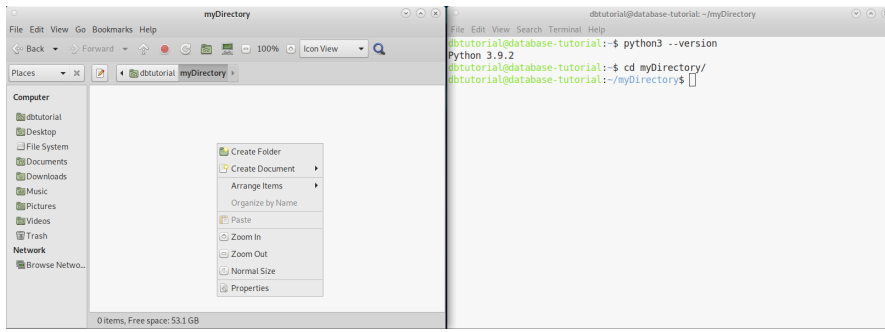


Figure 9: Moving to the newly created directory /home/dbtutorial/myDirectory.

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

```
1 ls -l
```

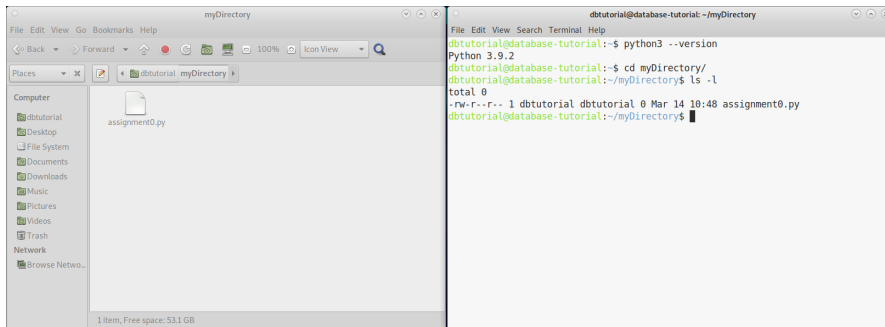


Figure 10: Showing the content of the new directory, i.e., assignment0-template.py.

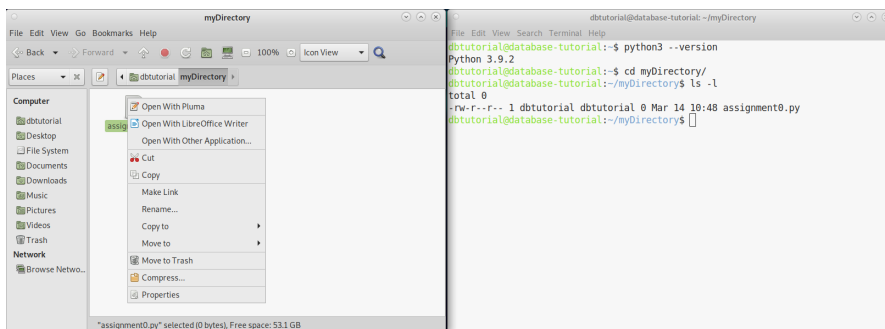
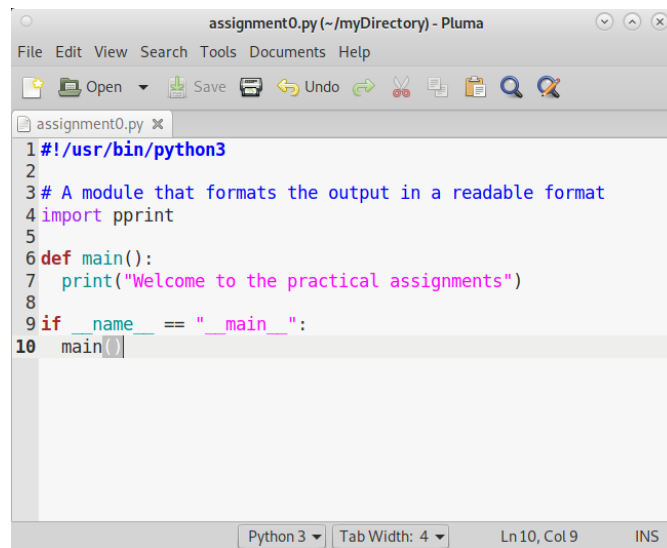


Figure 11: Editing a file with Pluma.

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.

The image shows a screenshot of a Pluma text editor window titled "assignment0.py (~myDirectory) - Pluma". The window contains the following Python code:

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

The status bar at the bottom indicates "Python 3", "Tab Width: 4", "Ln 10, Col 9", and "INS".

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

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:

`-rw-r--r--`

Each color-coded triple, `rw-`, `r-`, and `r-`, denotes the permissions for a specific category 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-template.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` command, which is an abbreviation for *change file mode bits*. Since we want to allow (+) our `user` to execute our Python3 code located in the file `assignment0-template.py`, we call the `chmod` command with `u+x` and the file we want to change it for, i.e., `assignment0-template.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 `python` command in combination with the file that we want to execute, i.e., `assignment0-template.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:

Welcome to the practical assignments

Listing 10: terminal – Allow Python code to execute.

```
1 chmod u+x assignment0-template.py
```

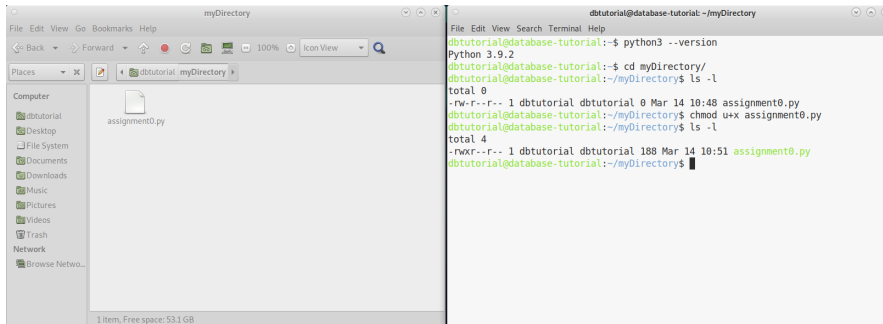


Figure 13: Modifying the permissions to execute our Python3 code.

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.

```
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()
```

Listing 11: terminal – Execute the Python code.

```
1 python3 assignment0-template.py
```

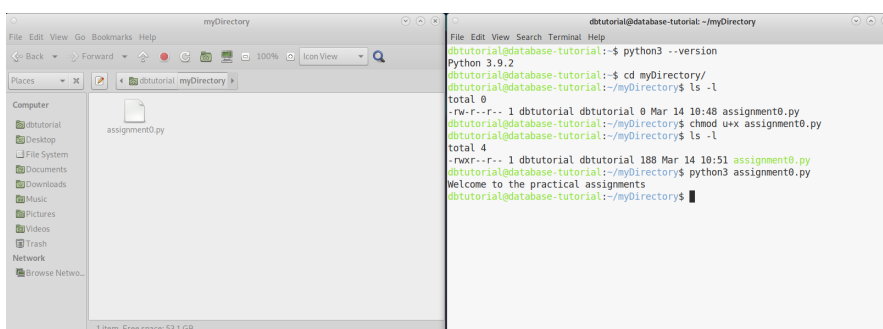


Figure 14: Executing our Python3 code.

Alternatively, you can try to execute the Python3 code using the method shown in Listing 12. Before, we explicitly specified that we want to execute the code with Python3. Contrarily, the system now knows implicitly to use Python3 due to the very first line in our code, which is a special type of comment that tells the systems to use Python3.

Listing 12: terminal – Execute the Python code (alternative).

```
1 ./assignment0-template.py
```

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 20 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 <i>Welcome</i> is correctly printed to the command line (in order).
2	The word <i>to</i> is correctly printed to the command line (in order).
2	The word <i>the</i> is correctly printed to the command line (in order).
2	The word <i>practical</i> is correctly printed to the command line (in order).
2	The word <i>assignments</i> is correctly printed to the command line (in order).
10	

Questionnaire The questionnaire contributes at most 10 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.
2	Answer one question during after-assignment meeting correctly.
10	