



**BIRZEIT UNIVERSITY**

**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**Second Semester 2022/2023**

**ENCS313- Linux Laboratory**

**Linux Project No.2 \_PYTHON\_**

**Student Name : Noura Manassra**

**Student ID : 1212359**

**Section : 5**

**Instructor : Dr. Abdel Salam Sayyad**

**Date : Jan. 31. 2024**

## **1. Abstract**

This Python project runs in a Linux environment, executing commands directly from Python code. Its purpose is to create a user-friendly manual for selected commands. The manual includes essential details like what each command does, its version, usage examples, related commands, and recommended ones. Users can easily navigate the manual and search for specific commands. If a command is found, the project displays relevant information, making it simple for users to understand and use various commands in the Linux environment, catering to both beginners and experienced users.

## Table of Figures

Fig[1] : Generating Manual .....	5
Fig[2] : DONE Generating Manual .....	5
Fig[3] : Verfication .....	6
Fig[4] : Unsuccessful verification example .....	6
Fig[5] : Done vefication.....	7
Fig[6] : Search for a command .....	7
Fig[7] : Full manual for a command .....	8
Fig[8] : description for a command .....	8
Fig[8] : version.....	9
Fig[9] : example and it's execution for a command .....	9
Fig[10] related commands for a command .....	10
Fig[11] : search for a command .....	10
Fig[12] : Recommended commands for a command and exiting the program.....	11

## **Table of Contents**

<b>1. Abstract</b>	II
<b>Table of Figures</b>	III
<b>Table of Contents</b>	1
<b>2. Theory</b>	2
<b>2.1 Linux Environment Background</b>	2
<b>2.2 Python Background</b>	2
<b>2.3 Why Python?</b>	2
<b>3. Procedure</b>	2
<b>4. Results</b>	5

## **2. Theory**

### **2.1 Linux Environment Background**

Linux is a powerful operating system widely used for its reliability and flexibility. It's packed with various commands that users can execute to perform specific tasks. However, understanding these commands can be a bit challenging, especially for beginners. Our project aims to bridge this gap by providing a straightforward manual for selected Linux commands.

### **2.2 Python Background**

Python is a popular programming language known for its simplicity and readability. It acts as the engine behind our project, allowing us to seamlessly interact with the Linux environment. Python's ease of use makes it an ideal choice for this project, enabling us to execute commands effortlessly and create a user-friendly interface for our manual.

### **2.3 Why Python?**

#### **Simplicity and Readability:**

Python is easy to understand, even for those new to programming. Its clean syntax and readability make it a great tool for developing projects where simplicity is key. This simplicity is crucial for creating a user-friendly manual that anyone can navigate and comprehend.

#### **Versatility:**

Python's versatility allows us to integrate it with the Linux environment seamlessly. Whether it's executing commands or processing data, Python adapts well to various tasks, making it a reliable choice for our project.

#### **Community Support:**

Python boasts a vast and supportive community. This is advantageous for our project as it ensures a wealth of resources, libraries, and tools that enhance our ability to develop a robust and efficient command guide.

## **3. Procedure**

First of all, I defined the important libraries to make python interact with the operating system environment, and for some other built in functions in Python to find the matching words for example or to find the matching lines which are used to find the recommendation commands.

Secondly, I defined the command\_recommendation, which will be used to find the recommendation for each command.

The first class defined at this project is CommandManualGenerator. And in it, I defined the first function, which is the constructed function which takes the file name, (the file which contains the commands). And inside it, I defined some properties to find the recommendation for each command correctly. Also I defined [creat\_manuals\_directory] , which create a directory for manuals which will create the directory if it does not exist, and if it exists it won't make an error so I made it in exception handling. Besides that, I created the [read\_commands\_fron\_file]

function which will check if the file exists, and if yes will read the commands. And if not, it will exit the program. Furthermore, I used [generate\_manual\_for\_command], this function will get the description, version, example, related commands, and the command recommendation. And it will print it to the user, the information. But for sure the description, example, related commands and the recommendation are called from another function. Moreover, I created [get\_command\_description] function which will find the description for each command. Also, I created [get\_command\_example], which will get the example for each command, I used directories here, so when it's the time to execute a command, it will run the values for this dictionary. And I created [get\_related\_commands], which will find the related using the shell commands. Furthermore, I used [write\_manual] which will print the data as the standard output given to us in the report. Besides that I used [generate\_manuals] function, here I will create a directory for the command, and make sure that all commands are read correctly from the file, so if all of them generated correctly, it will print to the user that all of them are generated correctly.

Now for the verification part I created function called [verify\_manual], this function is also used at the same class with other functions mentioned above. First, I created a file path for a command manual based on the name of the command, and all of them are stored in CommandsManuals. And it checks if the manual file for the specific command exists, if it does, it proceeds to open the file and read its content into the variable xml\_content, else it exits the program. I initialized a variable which will check if the verification succeed or not, and compare the content including the description, examples, related to get the verification for the command. And for this function I used [verify\_manuals], so each command will be verified alone.

Furthermore, I used the [search\_command\_manual] function, here were is the program in main is executed, and it will look for the commands created using the .xml, and for the menu will be showed up here is just to show the information for each command, this will happen only if the command substring exists between the xml files. Also I used the function [display\_part], at this part it reads the xml file, and searches for the specified command, and it displays the content between the <start\_tag> and the <end\_tag> if the file exists, otherwise it prints the error message. Also I used [get\_recommendations], as I mentioned before I created a lists for the recommendations part, all of them has this style commandName\_recommendation, so here it will call the list depending on it's name. And now for the last function at this class which is [recommend\_commands], which will get the last search for the user, and will call the get\_recommendations function to get the recommended function for the last search by the user, and will print it if it exists. Otherwise it will print that the recommendation not found.

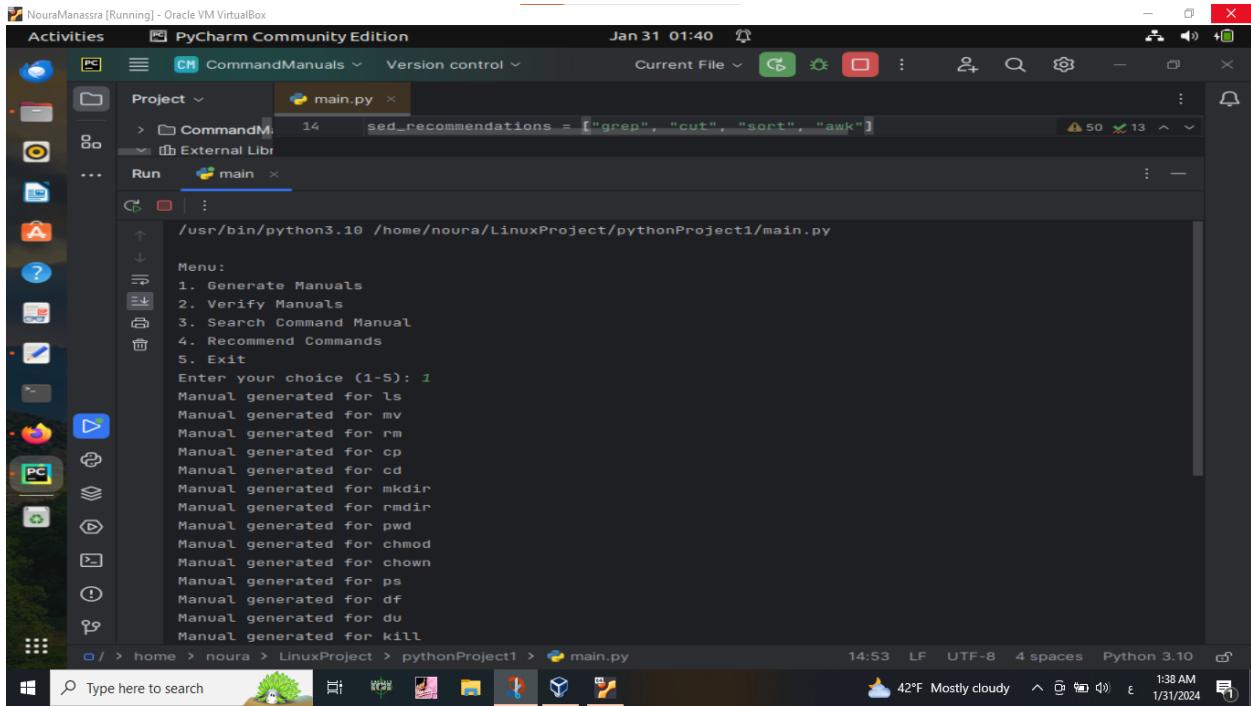
So Overall, the previous class finds each item at the manual for each command. And here is the second class, which is [CommandManual] , this class will bring the description, version, example, execute\_example and the related commands for each command. So later an objects from this class will be used in the XMLSerializer.

The last class used is the [XMLSerializer], here it uses the [CommandManual] object and convert it into XML strings, and note here that I used the tree technique, so each node will present the command name, and all it's children will act as its attributes including the description, example, version and related commands.

The last thing, which will execute the program is the main function, and the script here creates an instance of [commandManualGenerator], and here exactly where the project works. Furthermore, I put the main list which contains the options for the user to choose what he wants.

## 4. Results

Here are the results of execution the python code.

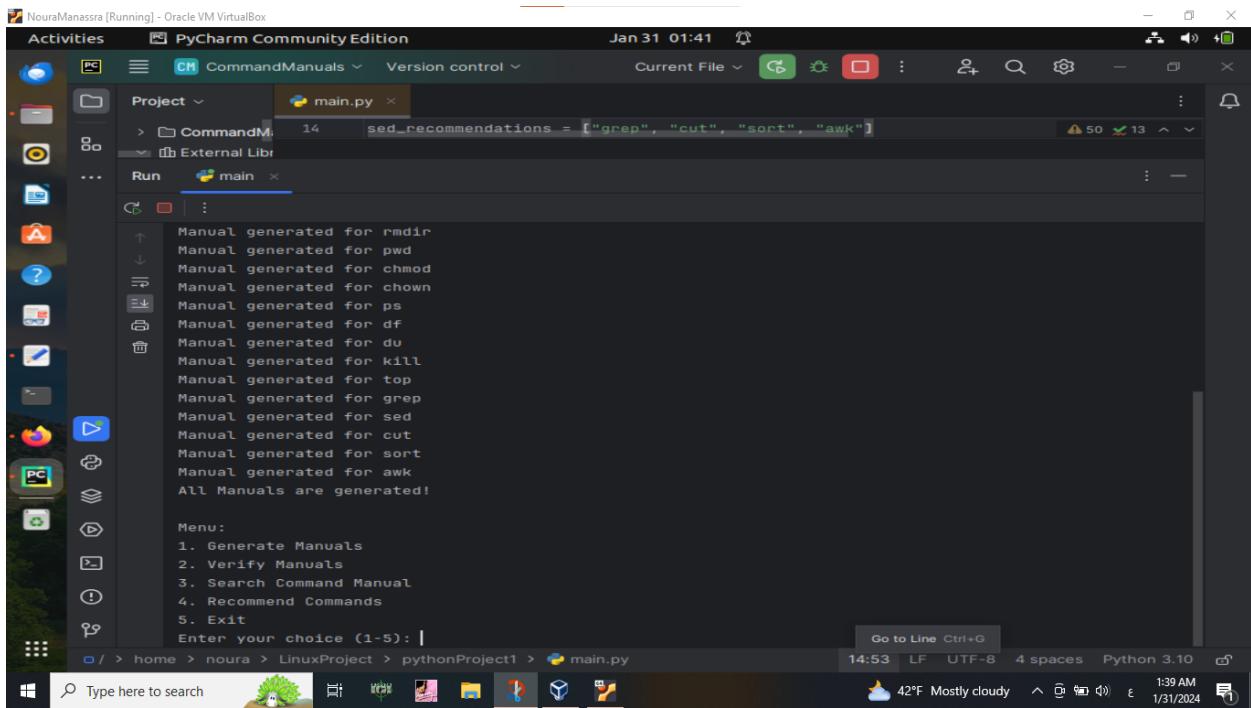


```
Activities PyCharm Community Edition Jan 31 01:40
Project main.py
> CommandM 14 Sed_recommendations = ["grep", "cut", "sort", "awk"]
External Libr
Run main

/usr/bin/python3.10 /home/noura/LinuxProject/pythonProject1/main.py

Menu:
1. Generate Manuals
2. Verify Manuals
3. Search Command Manual
4. Recommend Commands
5. Exit
Enter your choice (1-5): 1
Manual generated for ls
Manual generated for mv
Manual generated for rm
Manual generated for cp
Manual generated for cd
Manual generated for mkdir
Manual generated for rmdir
Manual generated for pwd
Manual generated for chmod
Manual generated for chown
Manual generated for ps
Manual generated for df
Manual generated for du
Manual generated for kill
14:53 LF UTF-8 4 spaces Python 3.10
Type here to search 42°F Mostly cloudy 1:38 AM 1/31/2024
```

Fig[1] : Generating Manual



```
Activities PyCharm Community Edition Jan 31 01:41
Project main.py
> CommandM 14 Sed_recommendations = ["grep", "cut", "sort", "awk"]
External Libr
Run main

Manual generated for rmdir
Manual generated for pwd
Manual generated for chmod
Manual generated for chown
Manual generated for ps
Manual generated for df
Manual generated for du
Manual generated for kill
Manual generated for top
Manual generated for grep
Manual generated for sed
Manual generated for cut
Manual generated for sort
Manual generated for awk
All Manuals are generated!

Menu:
1. Generate Manuals
2. Verify Manuals
3. Search Command Manual
4. Recommend Commands
5. Exit
Enter your choice (1-5): | 14:53 LF UTF-8 4 spaces Python 3.10
Type here to search 42°F Mostly cloudy 1:39 AM 1/31/2024
```

Fig[2] : DONE Generating Manual

The screenshot shows the PyCharm Community Edition interface. The project navigation bar at the top indicates "Activities", "PyCharm Community Edition", "Jan 31 01:42", and "Current File". The main window displays a terminal-like interface with the following output:

```
Project main.py
> CommandM 14 Sed_recommendations = ["grep", "cut", "sort", "awk"]
> External Libr
Run main

Menu:
1. Generate Manuals
2. Verify Manuals
3. Search Command Manual
4. Recommend Commands
5. Exit
Enter your choice (1-5): 2
Verification successful for ls
Verification successful for mv
Verification successful for rm
Verification successful for cp
Verification successful for cd
Verification successful for mkdir
Verification successful for rmdir
Verification successful for pwd
Verification successful for chmod
Error: The Example is not verified for chown!
Expected Example: chown: invalid user: 'username:groupname'
Actual Example: Error executing example: Command 'touch OwnerFile && chown username:groupname OwnerFile
Error: The Example is not verified for ps!
Expected Example: PID TTY      TIME CMD
1343 ?    00:00:01 systemd
1344 ?    00:00:00 (sd-pam)
1350 ?    00:00:00 pipewire
1351 ?    00:00:00 pipewire-media-
1352 ?    00:00:00 pulseaudio
1353 ?    00:00:00 snapd-desktop-i
1358 ?    00:00:00 ubuntu-report
1364 ?    00:00:01 dbus-daemon
1375 ?    00:00:00 gvfsd
1383 ?    00:00:00 gvfsd-fuse
1392 ?    00:00:00 xdg-document-po
```

The status bar at the bottom shows "14:53 LF UTF-8 4 spaces Python 3.10".

Fig[3] : Verification

Note for the unverified commands, it will print the expected and the real results.

The screenshot shows the PyCharm Community Edition interface. The project navigation bar at the top indicates "Activities", "PyCharm Community Edition", "Jan 31 01:42", and "Current File". The main window displays a terminal-like interface with the following output:

```
Project main.py
> CommandM 14 sed_recommendations = ["grep", "cut", "sort", "awk"]
> External Libr
Run main

verification successful for rm
Verification successful for cp
Verification successful for cd
Verification successful for mkdir
Verification successful for rmdir
Verification successful for pwd
Verification successful for chmod
Error: The Example is not verified for chown!
Expected Example: chown: invalid user: 'username:groupname'
Actual Example: Error executing example: Command 'touch OwnerFile && chown username:groupname OwnerFile
Error: The Example is not verified for ps!
Expected Example: PID TTY      TIME CMD
1343 ?    00:00:01 systemd
1344 ?    00:00:00 (sd-pam)
1350 ?    00:00:00 pipewire
1351 ?    00:00:00 pipewire-media-
1352 ?    00:00:00 pulseaudio
1353 ?    00:00:00 snapd-desktop-i
1358 ?    00:00:00 ubuntu-report
1364 ?    00:00:01 dbus-daemon
1375 ?    00:00:00 gvfsd
1383 ?    00:00:00 gvfsd-fuse
1392 ?    00:00:00 xdg-document-po
```

The status bar at the bottom shows "14:53 LF UTF-8 4 spaces Python 3.10".

Fig[4] : Unsuccessful verification example

The screenshot shows the PyCharm Community Edition interface. The main window displays the output of a script named 'main.py'. The terminal pane shows the following text:

```
12643 root 20 0 73196 13696 11520 S 0.0 0.2 0:00.02 cupsd
12644 root 20 0 0 0 0 I 0.0 0.0 0:00.13 kworker+
12646 root 20 0 172624 11648 10240 S 0.0 0.2 0:00.02 cups-br+
12688 root 20 0 0 0 0 I 0.0 0.0 0:00.03 kworker+
12737 root 20 0 0 0 0 I 0.0 0.0 0:00.01 kworker+
12739 root 20 0 0 0 0 I 0.0 0.0 0:00.01 kworker+
12794 noura 20 0 22712 12244 6656 S 0.0 0.2 0:00.11 python3+
13217 noura 20 0 2888 1536 1536 S 0.0 0.0 0:00.00 sh
Verification successful for grep
Verification successful for sed
Verification successful for cut
Verification successful for sort
Verification successful for awk
Verification Is Done !
```

Below the terminal, a menu is displayed:

- Menu:
- 1. Generate Manuals
- 2. Verify Manuals
- 3. Search Command Manual
- 4. Recommend Commands
- 5. Exit

The status bar at the bottom shows the path: /home/noura/LinuxProject/pythonProject1/main.py, the time: 14:53, and the Python version: Python 3.10.

Fig[5] : Done vefication

The screenshot shows the PyCharm Community Edition interface. The main window displays the output of a script named 'main.py'. The terminal pane shows the following text:

```
Enter your choice (1-5): 3
Enter the command name or topic: ls
Command manual for 'ls' exists:

Select a part to display:
1. Full Manual
2. Command Description
3. Version History
4. Example
5. Execute Example
6. Related Commands
7. Go Back to Main Menu
Enter your choice (1-7): 1
<Manuals>
  <CommandManual>
    <CommandName>ls</CommandName>
    <CommandDescription>List information about the FILEs (the current directory by default).</CommandDescription>
    <VersionHistory>ls (GNU coreutils) 8.32
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

The status bar at the bottom shows the path: /home/noura/LinuxProject/pythonProject1/main.py, the time: 14:53, and the Python version: Python 3.10.

Fig[6] : Search for a command

The screenshot shows the PyCharm Community Edition interface. The main window displays a terminal-like interface with the following text:

```
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
Written by Richard M. Stallman and David MacKenzie.</VersionHistory>  
<Example>ls</Example>  
<ExecuteExample>CommandManuals  
commands.txt  
KillMeFile  
main.py  
OwnerFile  
to</ExecuteExample>  
<RelatedCommands>else  
false  
alsabat-test  
alsa-info  
lspcmcia  
update-shells  
alsactl  
lsmod  
alsa  
lspci  
tclsh8.6  
md5sum.textutils
```

The status bar at the bottom shows the path: / home > noura > LinuxProject > pythonProject1 > main.py. The bottom right corner shows the date and time: Jan 31 01:46, 14:53 LF UTF-8 4 spaces Python 3.10, 42°F Mostly cloudy, 1:45 AM, 1/31/2024.

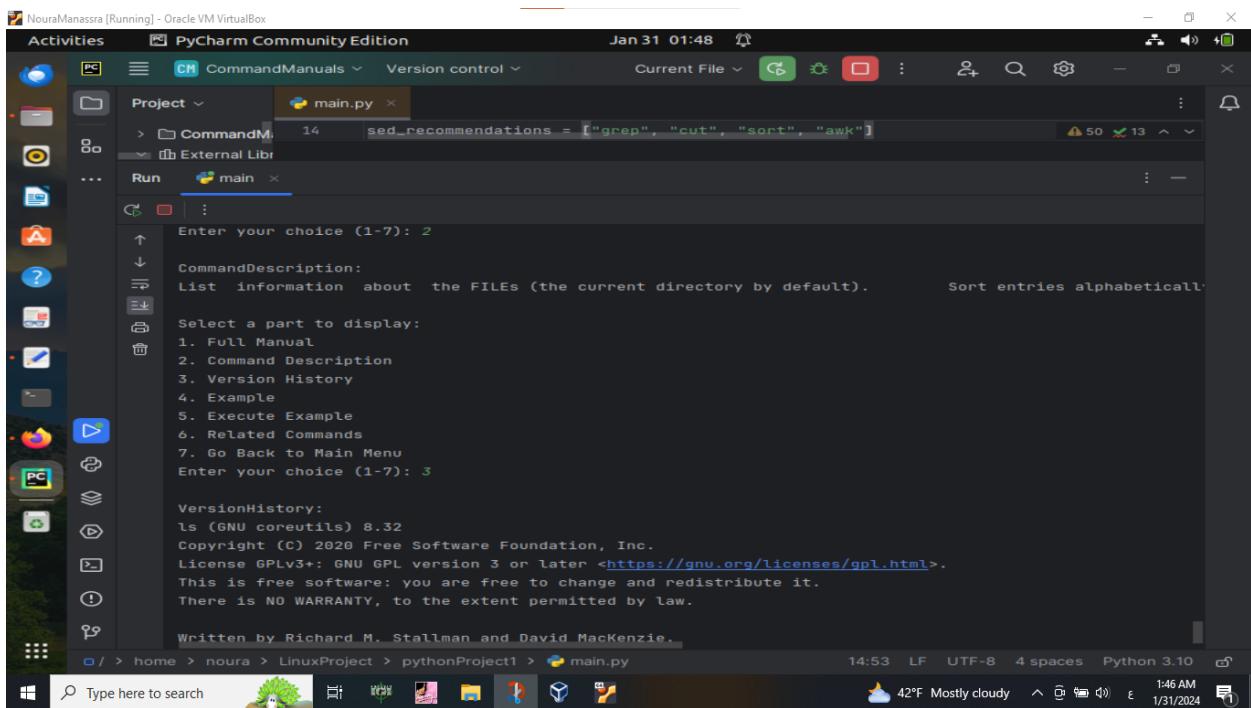
Fig[7] : Full manual for a command

The screenshot shows the PyCharm Community Edition interface. The main window displays a terminal-like interface with the following text:

```
xlsatoms  
lsns  
lscpu</RelatedCommands>  
</CommandManual>  
</Manuals>  
  
Select a part to display:  
1. Full Manual  
2. Command Description  
3. Version History  
4. Example  
5. Execute Example  
6. Related Commands  
7. Go Back to Main Menu  
Enter your choice (1-7): 2  
  
CommandDescription:  
List information about the FILES (the current directory by default). Sort entries alphabetical.  
  
Select a part to display:  
1. Full Manual  
2. Command Description
```

The status bar at the bottom shows the path: / home > noura > LinuxProject > pythonProject1 > main.py. The bottom right corner shows the date and time: Jan 31 01:47, 14:53 LF UTF-8 4 spaces Python 3.10, 42°F Mostly cloudy, 1:46 AM, 1/31/2024.

Fig[8] : description for a command



```
Activities PyCharm Community Edition Jan 31 01:48 Current File Run main.py

Project > CommandM 14 Sed_recommendations = ["grep", "cut", "sort", "awk"]
> External Libr

Run main

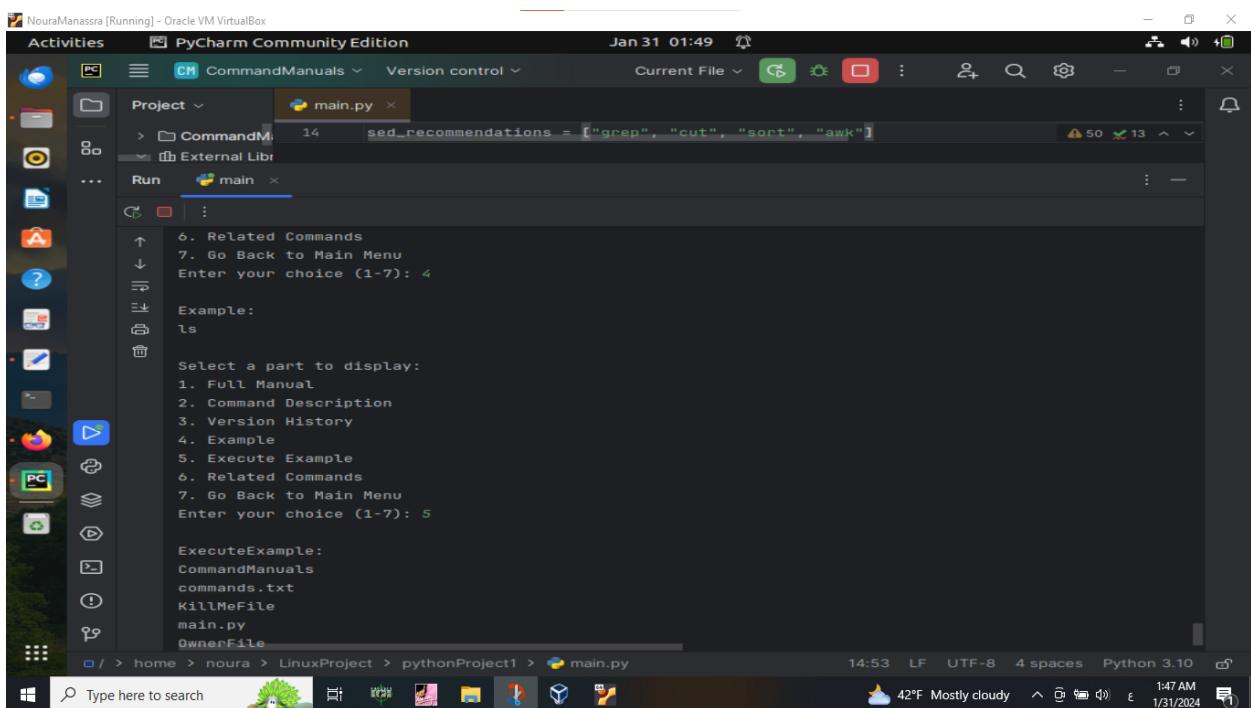
Enter your choice (1-7): 2
↑ ↓ CommandDescription: List information about the FILES (the current directory by default). Sort entries alphabetically.
Select a part to display:
1. Full Manual
2. Command Description
3. Version History
4. Example
5. Execute Example
6. Related Commands
7. Go Back to Main Menu
Enter your choice (1-7): 3

VersionHistory:
ls (GNU coreutils) 8.32
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Richard M. Stallman and David MacKenzie.

14:53 LF UTF-8 4 spaces Python 3.10 1:46 AM 1/31/2024
```

Fig[8] : version



```
Activities PyCharm Community Edition Jan 31 01:49 Current File Run main.py

Project > CommandM 14 Sed_recommendations = ["grep", "cut", "sort", "awk"]
> External Libr

Run main

6. Related Commands
7. Go Back to Main Menu
Enter your choice (1-7): 4

Select a part to display:
1. Full Manual
2. Command Description
3. Version History
4. Example
5. Execute Example
6. Related Commands
7. Go Back to Main Menu
Enter your choice (1-7): 5

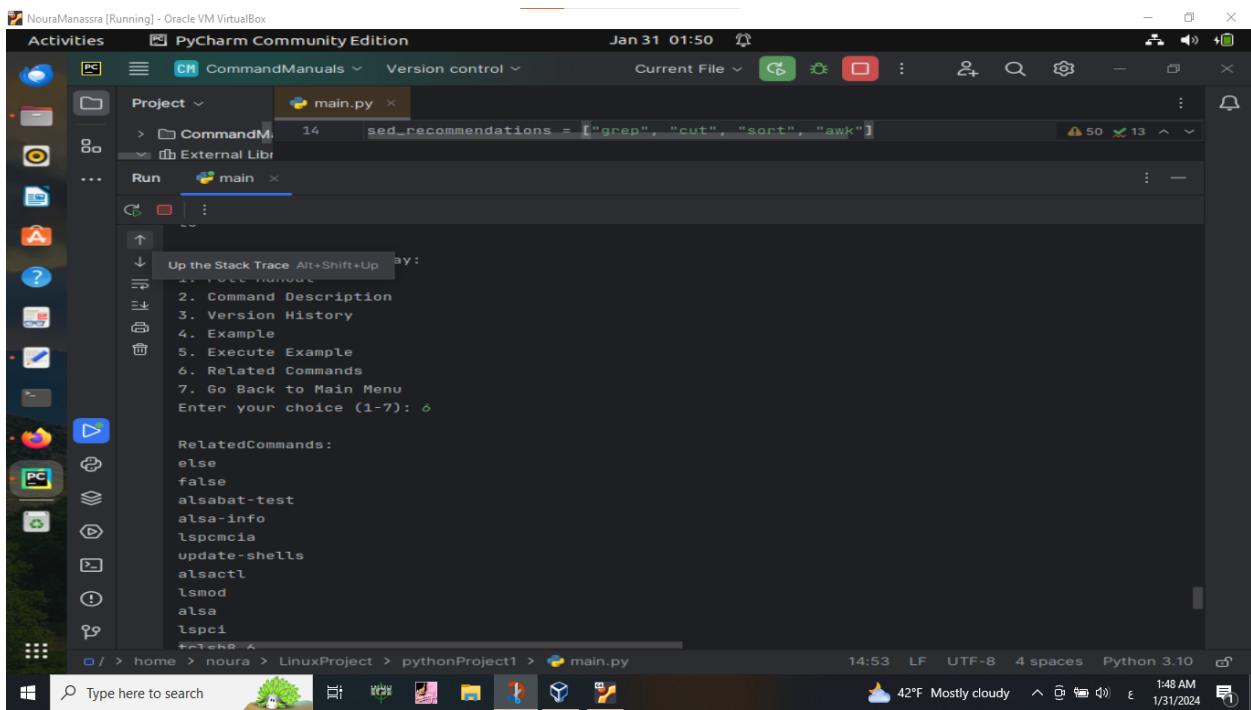
Example:
ls

Select a part to display:
1. Full Manual
2. Command Description
3. Version History
4. Example
5. Execute Example
6. Related Commands
7. Go Back to Main Menu
Enter your choice (1-7): 5

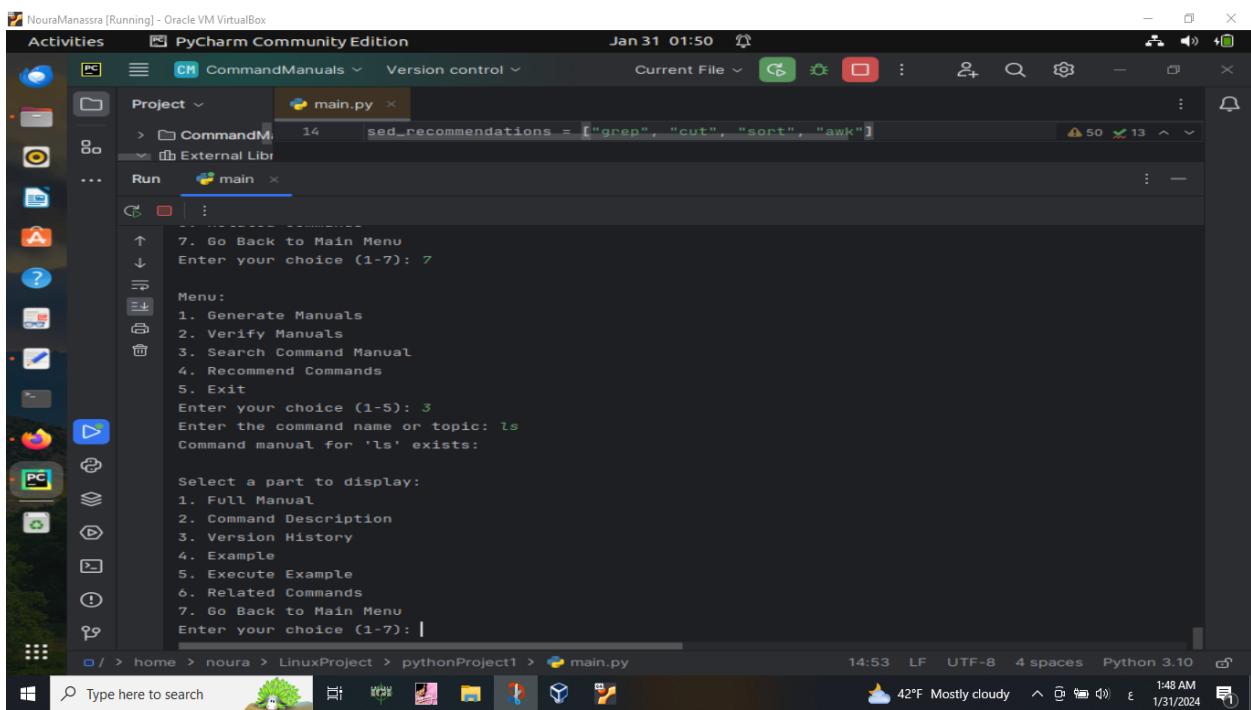
ExecuteExample:
CommandManuals
commands.txt
KillMeFile
main.py
OwnerFile

14:53 LF UTF-8 4 spaces Python 3.10 1:47 AM 1/31/2024
```

Fig[9] : example and it's execution for a command



Fig[10] related commands for a command



Fig[11] : search for a command

The screenshot shows a PyCharm Community Edition interface running on a Linux desktop. The terminal window displays a Python script named 'main.py' which provides command recommendations based on user input. The output shows:

```
Sed_recommendations = ["grep", "cut", "sort", "awk"]
5. Exit
Enter your choice (1-5): 4
Recommendations for 'ls':
- mv
- rm
- cp
- cd
- mkdir
- pwd
- rmdir

Menu:
1. Generate Manuals
2. Verify Manuals
3. Search Command Manual
4. Recommend Commands
5. Exit
Enter your choice (1-5): 5
allah ma3ak!!
```

The terminal also shows the command was completed successfully with an exit code of 0.

Fig[12] : Recommended commands  
for a command and exiting the  
program