Faculty of Engineering & Technology

Electrical & Computer Engineering Department

Linux Laboratory

ENCS3130

Project-2

Prepared by: Ahmad Ghazawneh

ID: 1201170

Instructor's Name: Dr Amr Slimi

TA: Tarek Zidan

Section: 1

Date: 28-1-2024

Abstract

This project aims to automate the creation of system manuals for shell commands using the Python programming language. generates structured XML files for each command, extracting information such as command description, version history, examples, and related commands, verification of the information of the creating files and searching on these files using Object-Oriented Programming principles to enhance modularity and encapsulation.

Table of Contents

Table of figure

1. Theory

1.1. OPP

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior [1].

1.2. XML

Extensible Markup Language (XML) is a markup language that provides rules to define any data. Unlike other programming languages, XML cannot perform computing operations by itself. Instead, any programming language or software can be implemented for structured data management [2].

1.3. Libraries used

subprocess module: The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This module intends to replace several older modules and functions [3].

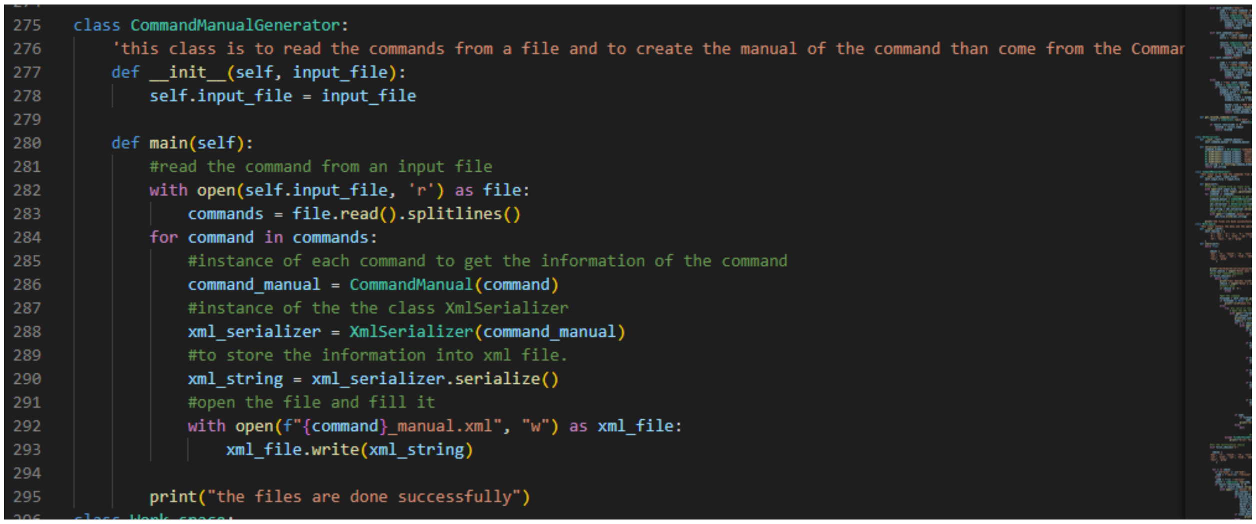
OS module: This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open() [4].

xml.etree.ElementTree module : The xml.etree.ElementTree module implements a simple and efficient API for parsing and creating XML data [5].

2. Procedure

The classes used:

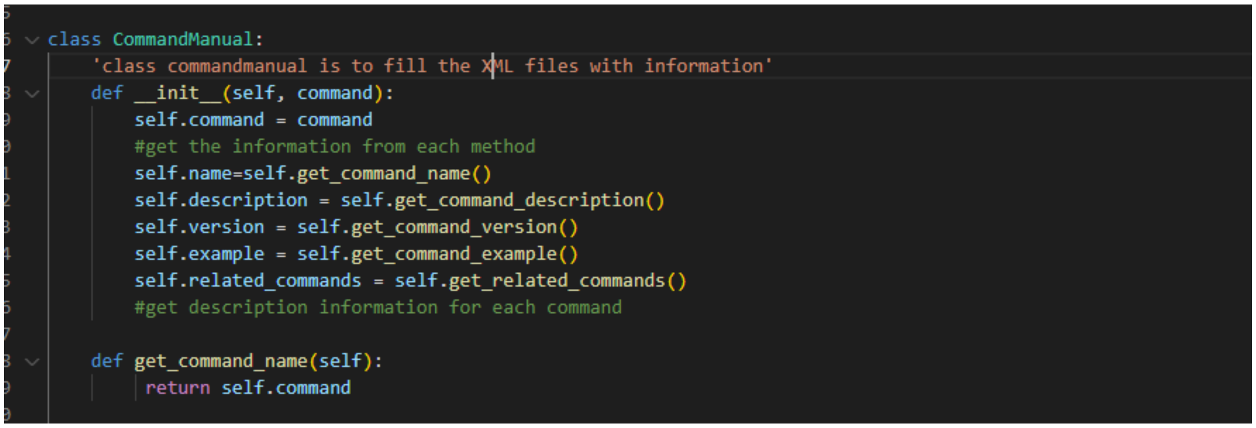
1) CommandManualGenerator Clasas: this class is to create the manual of the command than come from the CommandManual classand read the command from a file.



```
class CommandManualGenerator:
    'this class is to read the commands from a file and to create the manual of the command than come from the Commar
    def __init__(self, input_file):
        self.input_file = input_file

    def main(self):
        #read the command from an input file
        with open(self.input_file, 'r') as file:
            commands = file.read().splitlines()
        for command in commands:
            #instance of each command to get the information of the command
            command_manual = CommandManual(command)
            #instance of the the class XmlSerializer
            xml_serializer = XmlSerializer(command_manual)
            #to store the information into xml file.
            xml_string = xml_serializer.serialize()
            #open the file and fill it
            with open(f"{command}_manual.xml", "w") as xml_file:
                xml_file.write(xml_string)

        print("the files are done successfully")
```

Figure 1-commandManualGenerator class

2. 'CommandManual' Class: this class is to fill the XML files with information

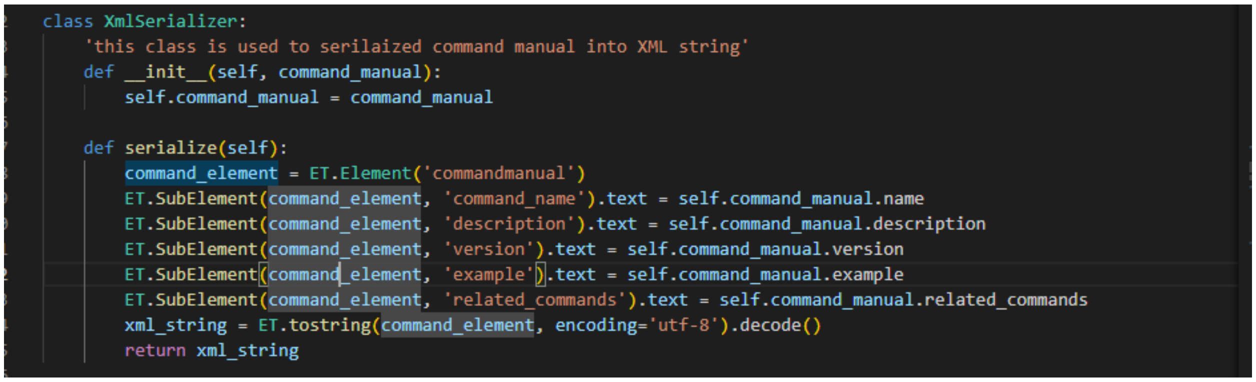


```
class CommandManual:
    'class commandmanual is to fill the XML files with information'
    def __init__(self, command):
        self.command = command
        #get the information from each method
        self.name=self.get_command_name()
        self.description = self.get_command_description()
        self.version = self.get_command_version()
        self.example = self.get_command_example()
        self.related_commands = self.get_related_commands()
        #get description information for each command

    def get_command_name(self):
        return self.command
```

Figure 2-CommandManual class

This class having 5 methods, the first is to get the name if the command, the second is to get the description of a command, the next is to get the example for the command, to get the version of the command and finally to get the version for a given command.

3. ‘XmlSerializer’ Class



```
class XmlSerializer:
    'this class is used to serilaized command manual into XML string'
    def __init__(self, command_manual):
        self.command_manual = command_manual

    def serialize(self):
        command_element = ET.Element('commandmanual')
        ET.SubElement(command_element, 'command_name').text = self.command_manual.name
        ET.SubElement(command_element, 'description').text = self.command_manual.description
        ET.SubElement(command_element, 'version').text = self.command_manual.version
        ET.SubElement(command_element, 'example').text = self.command_manual.example
        ET.SubElement(command_element, 'related_commands').text = self.command_manual.related_commands
        xml_string = ET.tostring(command_element, encoding='utf-8').decode()
        return xml_string
```

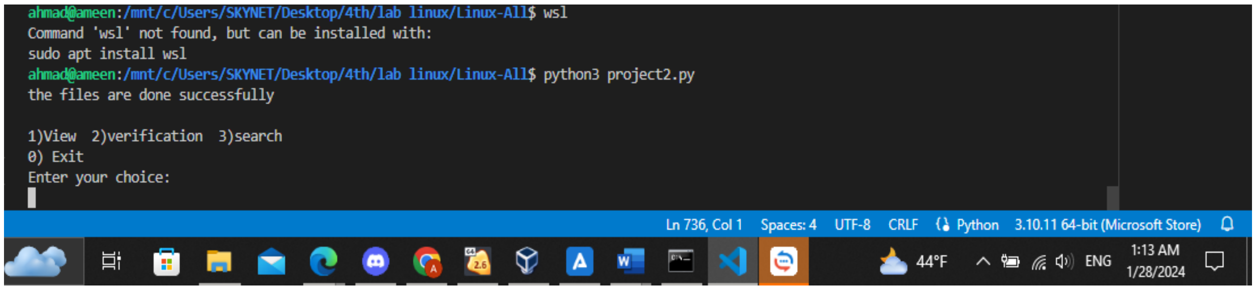
Figure 3-XMLSeriailzer Class

This class used to serialize the command manual into xml string and return the value.

4) class Workspace class: which contain all viewing, verification and search about information.

Create Files:

When run the python code, the Files created and filled successfully and wait the user to make a choice.

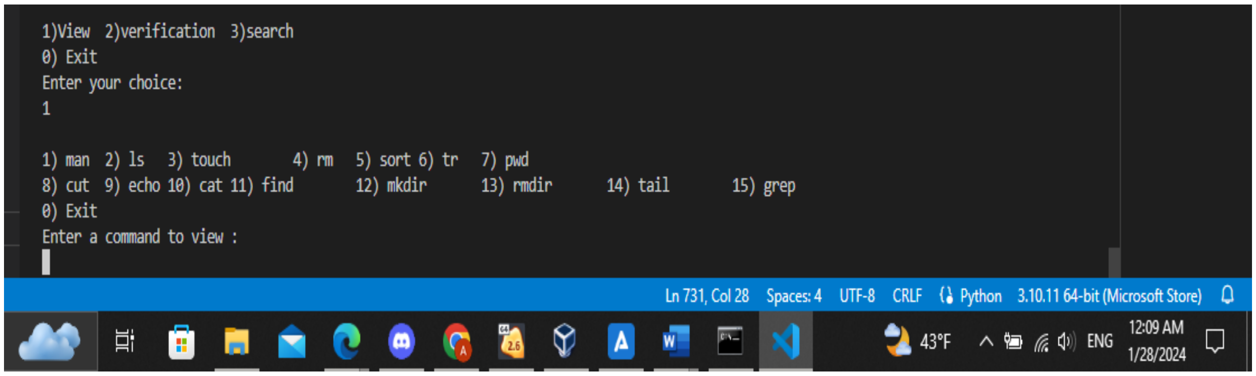


```
ahmad@ameen:/mnt/c/Users/SKYNET/Desktop/4th/lab linux/Linux-All$ wsl
Command 'wsl' not found, but can be installed with:
sudo apt install wsl
ahmad@ameen:/mnt/c/Users/SKYNET/Desktop/4th/lab linux/Linux-All$ python3 project2.py
the files are done successfully

1)View  2)verification  3)search
0) Exit
Enter your choice:
```

Figure 4-Create files output

The wsl is for let the python works as the Linux

View option:



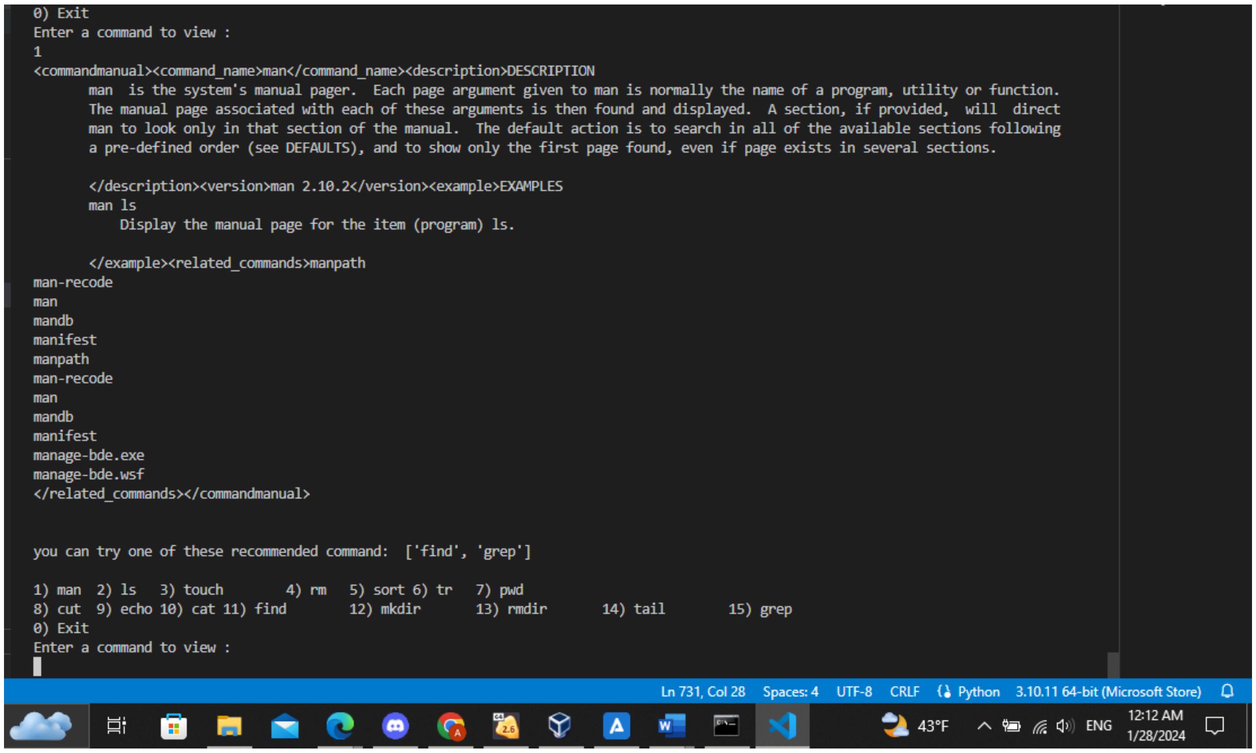
```
1)View  2)verification  3)search
0) Exit
Enter your choice:
1

1) man  2) ls   3) touch       4) rm   5) sort 6) tr   7) pwd
8) cut  9) echo 10) cat 11) find       12) mkdir       13) rmdir       14) tail        15) grep
0) Exit
Enter a command to view :
```

Figure 5-View Options output

If the User choose the View option by type 1 on the keyboard.

The names of the command file will appear in the terminal and each one has number to view it can used to show its content.

When the user types any number of the command, the commands information will execute as following figure.

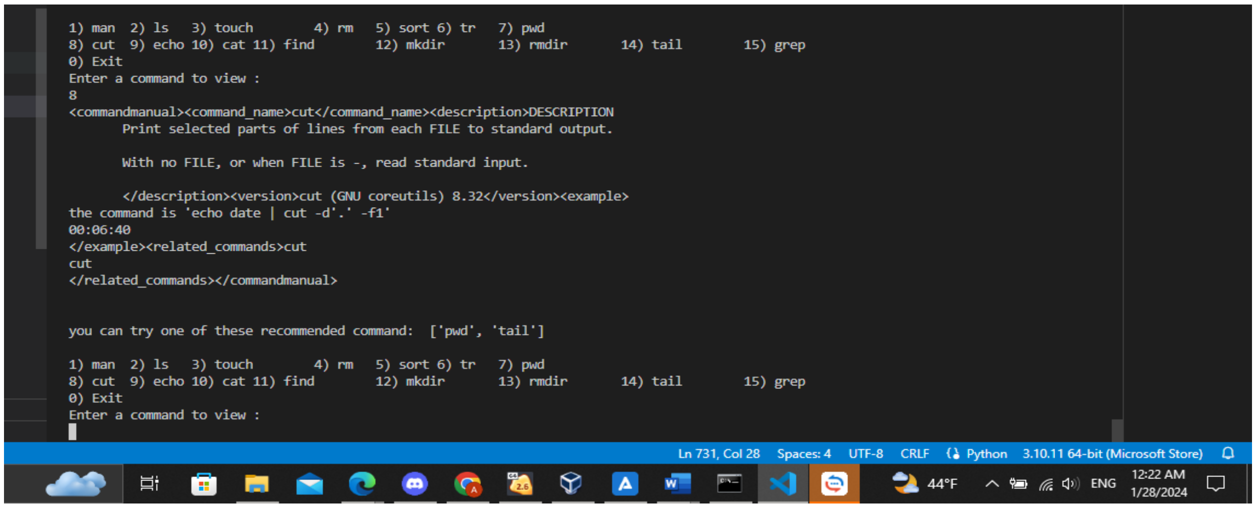


```
0) Exit
Enter a command to view :
1
<commandmanual><command_name>man</command_name><description>DESCRIPTION
       man  is the system's manual pager.  Each page argument given to man is normally the name of a program, utility or function.
       The manual page associated with each of these arguments is then found and displayed.  A section, if provided,  will  direct
       man to look only in that section of the manual.  The default action is to search in all of the available sections following
       a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.

       </description><version>man 2.10.2</version><example>EXAMPLES
       man ls
           Display the manual page for the item (program) ls.

       </example><related_commands>manpath
man-recode
man
mandb
manifest
manpath
man-recode
man
mandb
manifest
manage-bde.exe
manage-bde.wsf
</related_commands></commandmanual>


you can try one of these recommended command:  ['find', 'grep']

1) man  2) ls   3) touch        4) rm   5) sort 6) tr   7) pwd
8) cut  9) echo 10) cat 11) find        12) mkdir       13) rmdir       14) tail        15) grep
0) Exit
Enter a command to view :
```

Figure 6-view one of XML file (1)

In the previous figure the man file was printed into the terminal when user type 1, it contains the description, version, example and related command for this command in the Xml formula.

Here is the viewing of another command:

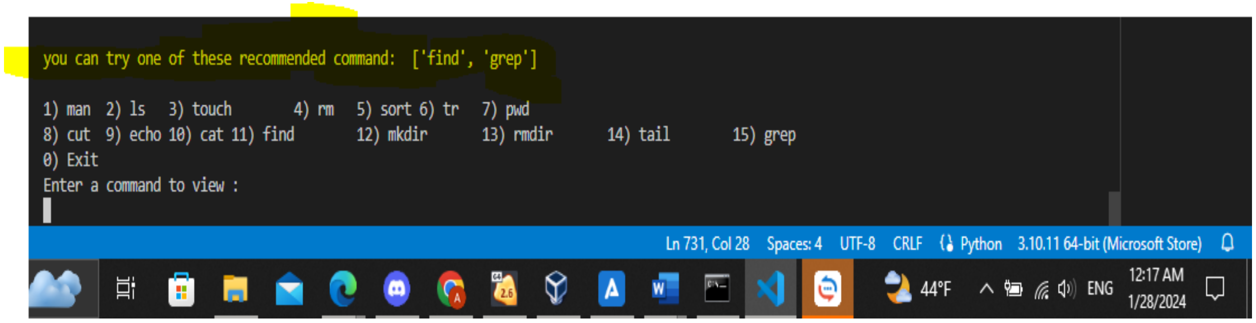


```
1) man  2) ls   3) touch        4) rm   5) sort 6) tr   7) pwd
8) cut  9) echo 10) cat 11) find        12) mkdir       13) rmdir       14) tail        15) grep
0) Exit
Enter a command to view :
8
<commandmanual><command_name>cut</command_name><description>DESCRIPTION
       Print selected parts of lines from each FILE to standard output.

       With no FILE, or when FILE is -, read standard input.

       </description><version>cut (GNU coreutils) 8.32</version><example>
the command is 'echo date | cut -d'.' -f1'
00:06:40
</example><related_commands>cut
cut
</related_commands></commandmanual>


you can try one of these recommended command:  ['pwd', 'tail']

1) man  2) ls   3) touch        4) rm   5) sort 6) tr   7) pwd
8) cut  9) echo 10) cat 11) find        12) mkdir       13) rmdir       14) tail        15) grep
0) Exit
Enter a command to view :
```

Figure 7-vie an XML file (2)

Recommended command:

After the information printed into the terminal, the executed command maybe has some recommended command that work the same as previous command or have a some similarly to it that printed inside the list in yellow



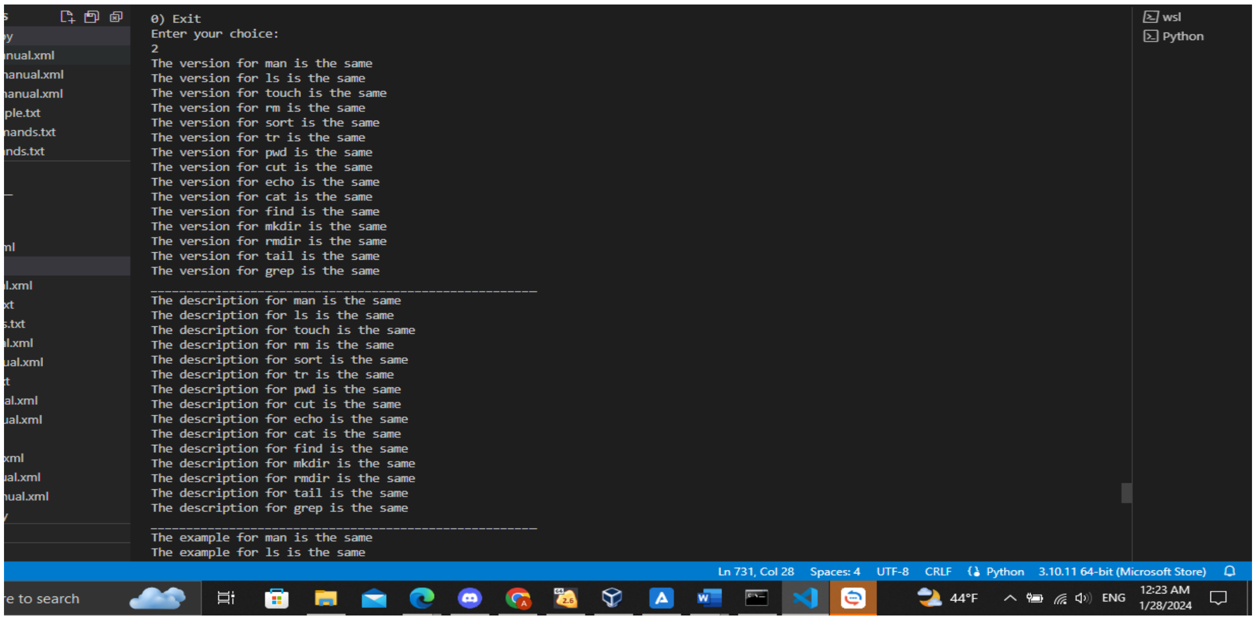
```
you can try one of these recommended command:  ['find', 'grep']

1) man  2) ls   3) touch       4) rm   5) sort 6) tr   7) pwd
8) cut  9) echo 10) cat 11) find       12) mkdir       13) rmdir       14) tail        15) grep
0) Exit
Enter a command to view :
```

Ln 731, Col 28   Spaces: 4   UTF-8   CRLF   Python   3.10.11 64-bit (Microsoft Store)
44°F
ENG
12:17 AM
1/28/2024

Figure 8-Recommended command output

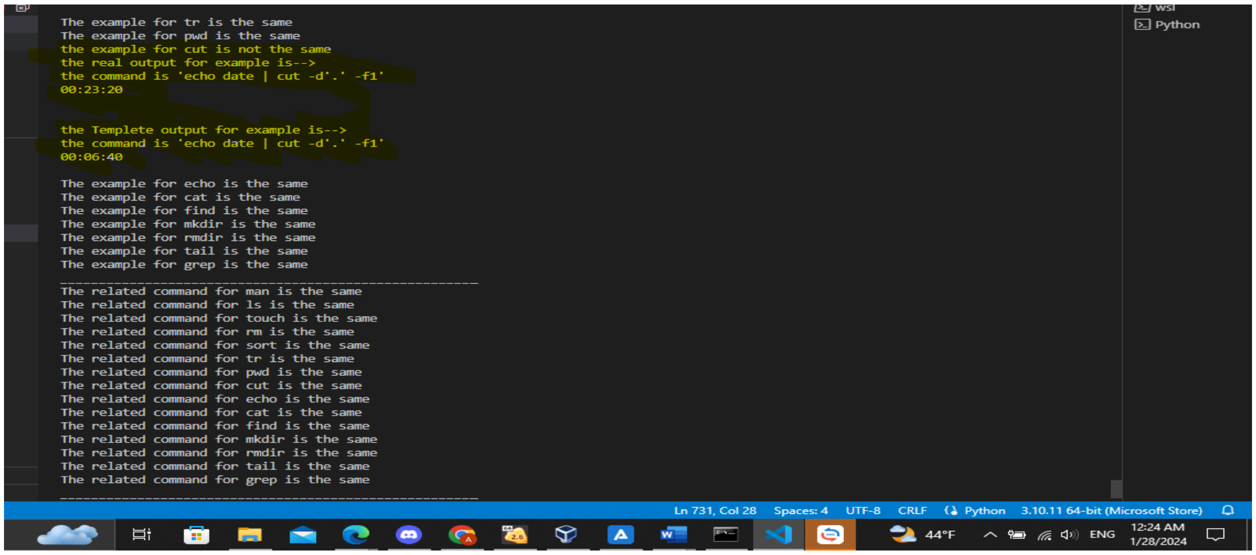
Verification of the Output of the command:

when user type 2 instead of 1. It will check al command file if it has same output with the real one(now).



```
0) Exit
Enter your choice:
2
The version for man is the same
The version for ls is the same
The version for touch is the same
The version for rm is the same
The version for sort is the same
The version for tr is the same
The version for pwd is the same
The version for cut is the same
The version for echo is the same
The version for cat is the same
The version for find is the same
The version for mkdir is the same
The version for rmdir is the same
The version for tail is the same
The version for grep is the same
-------------------------------------------------------
The description for man is the same
The description for ls is the same
The description for touch is the same
The description for rm is the same
The description for sort is the same
The description for tr is the same
The description for pwd is the same
The description for cut is the same
The description for echo is the same
The description for cat is the same
The description for find is the same
The description for mkdir is the same
The description for rmdir is the same
The description for tail is the same
The description for grep is the same
-------------------------------------------------------
The example for man is the same
The example for ls is the same
```

Figure 9-verification output(1)



```
The example for tr is the same
The example for pwd is the same
the example for cut is not the same
the real output for example is-->
the command is 'echo date | cut -d'.' -f1'
00:23:20


the Templete output for example is-->
the command is 'echo date | cut -d'.' -f1'
00:06:40

The example for echo is the same
The example for cat is the same
The example for find is the same
The example for mkdir is the same
The example for rmdir is the same
The example for tail is the same
The example for grep is the same
-------------------------------------------------------
The related command for man is the same
The related command for ls is the same
The related command for touch is the same
The related command for rm is the same
The related command for sort is the same
The related command for tr is the same
The related command for pwd is the same
The related command for cut is the same
The related command for echo is the same
The related command for cat is the same
The related command for find is the same
The related command for mkdir is the same
The related command for rmdir is the same
The related command for tail is the same
The related command for grep is the same
-------------------------------------------------------
```

wsl
Python
Ln 731, Col 28 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (Microsoft Store)
44°F ENG 12:24 AM 1/28/2024

Figure 10-verification output (2)

The previous two figure shown the verification of the Output of each file.

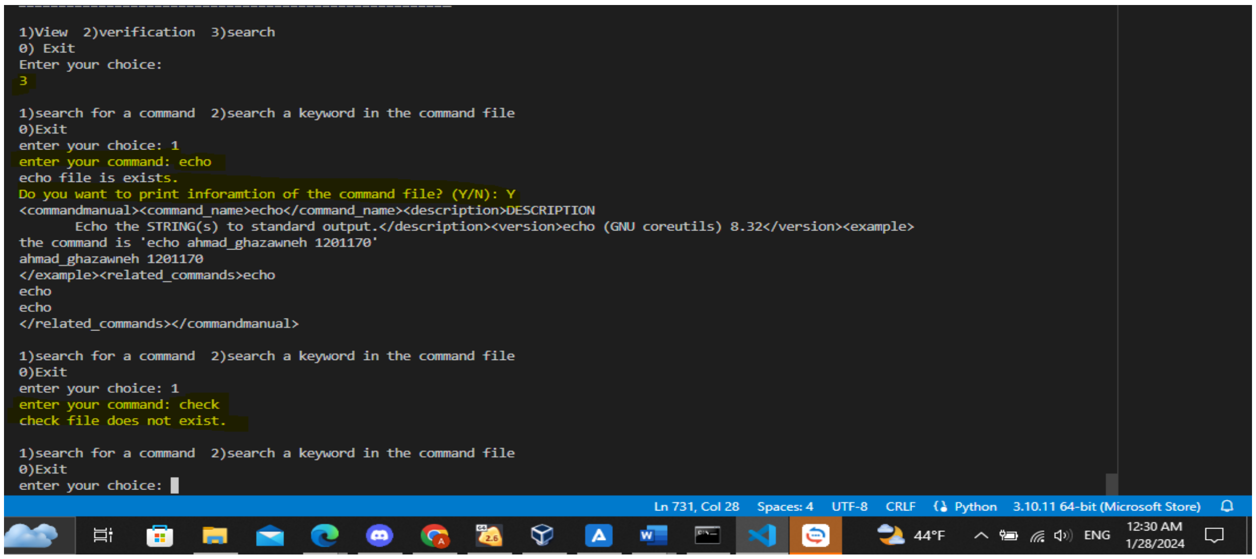
From the output, the verification was done (not all cases are done successfully)

Noticed in the second figure, there is a command does not have the same output as in the real one

Noticed that the information which not the same as the real one, both of them were print to noticed where is the different.

Search:

The user can search for a command file or search for a keyword inside any file from the created files (XML) by enter the choice number 3 which for search, and enter search for a command or search to a keyword.



```
-------------------------------------------------------
1)View  2)verification  3)search
0) Exit
Enter your choice:
3

1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 1
enter your command: echo
echo file is exists.
Do you want to print inforamtion of the command file? (Y/N): Y
<commandmanual><command_name>echo</command_name><description>DESCRIPTION
       Echo the STRING(s) to standard output.</description><version>echo (GNU coreutils) 8.32</version><example>
the command is 'echo ahmad_ghazawneh 1201170'
ahmad_ghazawneh 1201170
</example><related_commands>echo
echo
echo
</related_commands></commandmanual>

1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 1
enter your command: check
check file does not exist.

1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 
```

Ln 731, Col 28 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (Microsoft Store)
44°F ENG 12:30 AM 1/28/2024

Figure 11-search for a command output

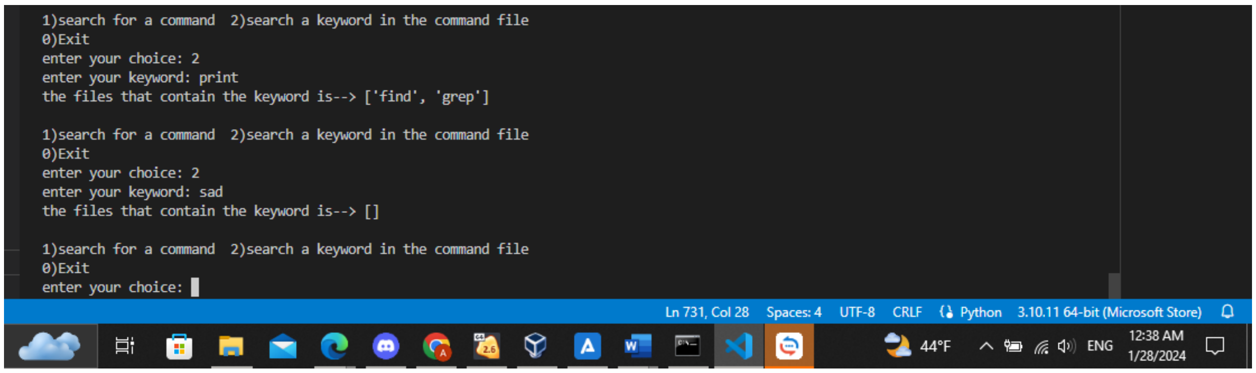
In the previous figure the user search for a command and enter "echo" as a command.
the program show names of file with that name exactly. And print it, and ask user if want to print the information of this file or not.

when the user search again to a command and type anything not included in this code

the output "name of the user enter does not exist". Since it is not from the creating files.

In the below figure the user choose to search for a keyword inside all of commands files.

When the user search for the "search" keyword, the files that contain this word was shown, on the other hand when the user search for a keyword not exist in any files, nothing appear to the user. The output be in the list.



```
1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 2
enter your keyword: print
the files that contain the keyword is--> ['find', 'grep']

1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 2
enter your keyword: sad
the files that contain the keyword is--> []

1)search for a command  2)search a keyword in the command file
0)Exit
enter your choice: 
```

Figure 12-search for a keyword output

3. Conclusion

In conclusion, this project gave a solution of creating automatic XML files using for shell command using python, by applying Object-Oriented Programming(classes) to make it easier to maintain and extend.

Using classes (CommandManual, CommandManualGenerator, and XmlSerializer) help to create the manual for each command to let the user work easy in the class name workspace to view, verification and search and finally to show the results of the code.

4. Reference

[1]. https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP [Accessed 30 Jan 2024 at 7:46]

[2].https://aws.amazon.com/whatis/xml/#:~:text=An%20Extensible%20Markup%20Language%20(XML,like%20Note [Accessed 30 Jan 2024 at 7:48]

[3]. https://docs.python.org/3/library/subprocess.html [Accessed 30 Jan 2024 at 8:44]

[4] https://docs.python.org/3/library/os.html [Accessed 30 Jan 2024 at 8:46]

[5] https://docs.python.org/3/library/xml.etree.elementtree.html [Accessed 30 Jan 2024 at 8:50]