University Of Karachi

DEPARTMENT OF COMPUTER SCIENCE (UBIT)

MCS FINAL

Evening Program

Project Report (FYP)

For

Notepedia | Word Processor

Prepared By:

Muhammad Fahmeed Qayyum (EP19101045)

Bushra Shaukat (EP19101011)

SUBMITTED TO:

SIR KHALID JAMAL

ACKNOWLEDGMENT

My profound gratitude to Almighty Allah for the knowledge, wisdom and understanding that he bestowed on me to carry out this project.

My grateful thanks also go to the Entire Computer science Department of the University of Karachi, **Sir Khalid Jamal** and all lecturers who prepared me from the base of computer science. The co-operation is much indeed appreciated.

Special thanks also to my parents; who encouraged supported and helped me financially, prayerfully and morally throughout this project.

May the Almighty Allah bless you all and keep you all to enjoy the fruit of your labor Insha'Allah.

PROJECT PROFILE:

PROJECT TITLE:

- Notepedia | Word Processor

HARDWARE AND SOFTWARE USED:

SOFTWARE REQUIREMENTS:

- Operating system: Linux (Debian based), Windows 7,8,10
- Front-end: Python, PyQt5
- Back-end: Python, speech_recognition, sys

HARDWARE REQUIREMENTS:

- RAM: 200MB
- Hard Disc: 10MB

DOCUMENTATION TOOLS:

- Microsoft Word 2013
- Microsoft PowerPoint 2013
- Notepedia | Word Processor

PROJECT DURATION:

- 8 weeks

INTRODUCTION:

This project will give us the information about the Notepedia | Word Processor. In this project I and my group member has implemented an interactive text editor that supports advance editing functions that are not find in your normal text editors plus infinite-level undo and multiple windows on the same file. We used Python and the PyQt5 (for GUI) for the implementation. The goal of this project is to implement a text editor that can handle text files with different fonts and styles. It is based on the simple editor presented in the course but with much useful and rich features. Since this word processor is written in Python the natural choice for the implementation language of our editor is Python as well, but we can also use C# or C++ instead.

It will be able to read and display a text file. Long lines are not wrapped but simply clipped at the right margin.

- The user will be able to type and delete characters in the text.
- There will be a caret and a selection. Words should also be selectable with a double click.
- The user should be able to use Artificial Intelligence and Deep Learning Algorithms to recognize speech directly from microphone and convert it into text.
- The user will be able to scroll.
- The user will be able to change font sizes and different fonts from variety of fonts available.
- The user will be able to make multiple instances of the same file and work

on multiple files.

- The user will be able to open an .npd file which is the extension for our software.
- The user will be able to save a file to use it later.
- The user will be able to cut, copy, paste a text and perform undo and redo operations.
- The user will be able to Find words and replace them with choice of particular words.
- The user will be able to set the current date and time with multiple format available.
- The user will be able to see the total words and symbols available live.
- The user should be able to insert tables with spacing and padding.
- The user should be able to insert nested tables within the table.
- The user should be able to insert images to make a document look more appealing.
- The user should be able to insert bulleted and numbered list.
- The user should be able to align the text for however he or she wants.
- The user should be able to set the superscripted and subscripted text formats.
- The user should be able to indent or dedent a line.
- The user should be able to print a document, set document measurements and set the type format the document will be printed on.
- The user should be able to print either a grayscale or a colored document (Depends on if the printer actually support it).
- The user should be able to preview the document before printing.

GATHERING MATERIALS:

The materials for our Word Processor are ideas and experiences that we have learned from YouTube Playlist. We have read books to better implement PyQt5 with Python into making own GUI based applications, Finally we implemented our software using Visual Studio Code and Python 3.6.9 and used icons from IconFinder.com, PyQt5 libraries, speech_recognition libraries, QtPrintSupport to support printing functionalities. All these experiences we have got and finally implemented them into our own unique software comprises of useful rich features.

The Materials used:

- Icons from IconFinder.
- PyQt5 Book (Creating GUI Applications with Python Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python).
- YouTube Playlist (PyQt with Python GUI Programming Tutorial).
- Visual Studio Code.
- Ideas from Microsoft Word/Notepad.
- How our word processor can be easily accessible to person with visual impairments.

FEATURES

Here are the features that are implemented in our project:

1. BASIC DISPLAY AND SCROLLING:

The editor displays the contents of a file in a window along with horizontal and vertical scrollbars. The editor works correctly when the window is resized. The size of the thumbs in the scrollbars properly reflect the current height and of the window relative to the total document height and width; this changes when the document is modified or the window is resized.

2. RICH TEXT:

All text will be displayed in an advance text format (like a program supporting rtf file type or html document displaying web pages, for example). The editor supports rich text features such as multiple fonts, boldface, italic, tabs, or adjustable indents.

3. LINES:

Lines in the document are terminated by newline characters ("\n"). Each line of the document should appear as a single line in the display (i.e., do not wrap long lines; the user can scroll horizontally instead). On Windows platforms, files sometimes use a two-character sequence "\r\n" to terminate lines; when reading in files, you should replace "\r\n" with just "\n", and when writing out files, you should always use "\n" as the line terminator.

4. DYNAMIC WIDTH FONT:

We used a dynamic width font for the editor, so that columns line up. I recommend using the Monospace font in a 12-point size

5. READING AND WRITING FILES:

The editor should require a single command-line argument giving the name of the file to edit. If the user types **Control-s**, the editor should save the current contents of the document back to the file.

6. INSERTION CURSOR:

When the user clicks in the window, the editor displays a blinking vertical bar just to the left of the closest character to the click. The bar blinks on and off: on for 0.5 seconds, then off for 0.5 seconds, and so on. If the mouse is dragged with the button down, the insertion cursor must follow the mouse. If a window loses input focus (i.e., you click in a different window) it will stop displaying the cursor; when the window regains focus it displays the cursor again.

4. TYPING:

When the user type's printable characters such as "A" or "%", the characters will be inserted at the cursor location and the cursor will advance to remain just after the character that was typed.

5. NEWLINES:

If the user types the **Enter** or **Return** key (which corresponds to the **Control-r** character, or **"\r"**), you should insert a newline character ("\n") in the document and display a new line in the window.

6. BACKSPACE AND DELETE:

If the user types the **Backspace** key, the editor deletes the character immediately before the cursor (unless there is a selection; see below). The **Delete** key should delete the character at the cursor location.

7. SELECTION:

If the user presses the mouse button and drags the mouse either forward or backward, the editor creates a selection corresponding to the range of characters between the character where the mouse was pressed and the current mouse location. The new selection replaces any previous selection. The selection must be displayed using some form of highlighting, such as a colored background or reverse video. Once a selection has been made, if the user presses the mouse button while holding down the **Shift key**, then the selection will be modified as if the mouse button had never been released (i.e. the current mouse position specifies one end of the selection, and the other end of the selection is determined by the location where the mouse was clicked without the **Shift key down**). If the mouse button is pressed and released without dragging, then any existing selection is canceled. You do not need to implement word selection for double-clicks.

8. SELECTION DELETION:

If a printable character or newline is typed at a time when there exists a selection, then the new character replaces the selection.

If **Backspace** or **Delete** is typed when there exists a selection, then the editor skips the normal behavior for these keys and simply delete the selected characters.

9. COPY AND PASTE:

If the user types **Control-c,** the editor copies the selected characters to the clipboard, where they can be pasted by other applications. If the user

types **Control-v**, the editor reads the contents of the clipboard and copy it to the cursor location (deleting any selected characters first). You must use Swing/AWT facilities for copying and pasting, so that you can select in this window and paste in other applications, and vice versa.

10. ARROW KEYS:

The four arrow keys moves the insertion cursor up, down, left, and right by one character. When moving up and down, the editor tries to maintain the same x-coordinate. For example, suppose the cursor is currently before the 10th character on line 13, and then the up arrow is typed twice. If line 13 only contains 5 characters, then the first up arrow will position the cursor just before the newline on that line. However, if line 12 contains 20 characters, then the second up arrow will position the cursor just before character 10 on that line. If the right arrow is typed at the end of a line, the cursor will be moved to the first character in the next line. If the left arrow is typed at the beginning of a line, the cursor will be moved to just before the newline in the preceding line.

11. INFINITE-LEVEL UNDO:

The editor records all modifications to the document. If the user types **Control-z**, the editor will undo the most recent modification. If the user types **Control-z** repeatedly, the editor will undo successively older changes, all the way back to the original version of the document loaded when the editor was first invoked. After undoing one or more modifications, the user may type Control-Alt-Z to redo the modifications in the reverse order they were undone. If the document is modified at a time when there exist redo-able operations, then all of the redo-able operations are discarded; future undo's and redo's will behave as if those

operations never occurred.

12. UNDO AND THE SELECTION AND CURSOR:

The undo/redo mechanism also records information about the selection and insertion cursor. Undo operations will restore the selection and cursor to the state they were in just before the original execution of the modification. Redo operations will restore the selection and cursor to the same state they had just after the original execution of the operation.

13. UNDO GROUPING:

If several characters are typed consecutively without repositioning the cursor, they well be treated as a unit for undo and redo: a single undo must undo all of the characters. If several characters are deleted with **Backspace** or **Delete**, all of these deletions must also be treated as a unit for undo and redo.

14. ALL MODIFICATIONS VISIBLE:

If a modification is made to a portion of the document that is off-screen, the document must be scrolled so that the modified text is visible. This can happen during undo and redo operations, and also during normal text inserts, if the window is scrolled to move the cursor off-screen. If multiple windows are open on the document, then only the window in which the modification was invoked will be scrolled: other windows will be repainted if needed, but they should not be scrolled.

15. NEW FILE:

We can make multiple instances of our document by clicking on the new icon from the toolbar or from the menu bar. We can also use **Ctrl-N** shortcut key to create an instance of our already running document and work on multiple files at

the same time. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Create a new document from Scratch”.**

16. OPEN FILE:

We can open multiple instances of new already saved documents by clicking on the open icon from the toolbar or from the menu bar. We can also use **Ctrl-O** shortcut key to open an already save document work on multiple files at the same time. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Open Existing Document”.**

17. SAVE FILE:

We can make save our document by clicking on the save icon from the toolbar or from the menu bar. We can also use **Ctrl-S** shortcut key to save our already running document. If the file name is not given already our document when we trigger the save icon will open a getSaveFileNameDialog to tell where the document should be saved. It opens a file explorer to save our document and if the name of a file is already decided then the file is just saved and overrited. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Save document”.**

18. PRINT A FILE:

We can print our document by clicking on the print icon from the toolbar or from the menu bar. We can also use **Ctrl-P** shortcut key to print our document. If the printer is connected then it will get the information regarding the printer, the model and version of a printer and shows in the recognized hardware and a button is provided inside a dialog so that print action can be performed. An Options button is also provided and when clicked it opens a new dialog which is used to select a directory for converting text to pdf and number of copies to print and the ranges and order of printing multiple pages in a document and hence contains 2 tabs (Copies) and (Options) tab and we can print a document in a colored manner or in a grayscale manner. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Print document".**

19. TEXT TO PDF CONVERTER:

We can print our document by clicking on the print icon from the toolbar or from the menu bar. We can also use **Ctrl-P** shortcut key to print our document. If the printer is connected then it will get the information regarding the printer, the model and version of a printer and shows in the recognized hardware and a button is provided inside a dialog so that print action can be performed. An Options button is also provided and when clicked it opens a new dialog which is used to select a directory for converting text to pdf. We can convert our text to pdf directly from Notepedia software and hence use it to read on any kind of

device that is working in your area. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about. We have to select a path where the converted document will be saved and then hit the print button to print a document in a pdf readable format.

20. PREVIEW PAGES BEFORE PRINTING:

We can preview our document prior to printing by clicking on the preview icon from the toolbar or from the menu bar. We can also use **Ctrl-Shift-P** shortcut key to preview our document before printing. A separate dialog will appear containing the information of how the document can be viewed (A landscape style) or (A portrait style) and Zoom in and Out a document and different pages that can be viewed directly inside a preview document dialog and how the pages can be viewed (Either they can be viewed as a single page or multiple pages) and the measurements of how the document should be printed and what will be the measurements while printing (Either in inches, points, millimeters, centimeters). And it also provides an icon to print directly from the preview dialog and then the print dialog will appear and then we can print the document directly from here. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about. We have to select a path where the converted document will be saved and then hit the print button to print a document in a pdf readable format. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a

status bar **“Preview document before printing”.**

21. FIND AND REPLACE:

WE can find the particular text if our document is very large and we can replace the text as well by clicking on the Find and Replace icon from the toolbar or from the menu bar. We can also use **Ctrl-F** shortcut key to open find and replace dialog so that we can find particular word inside our document or words and we can replace them either manually or by using a replace function inside find and replace dialog. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Find and replace words in your document”.**

22. INSERTING DATE/TIME:

WE can Insert current date and time by clicking on the date time icon from the toolbar or from the menu bar. We can also use **Ctrl-D** shortcut key to insert current date and time and when the dialog is opened we can choose multiple formats of displaying either date or time or both in multiple fashions. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Insert current date/time”.**

23. WORD/SYMBOLS COUNTER:

WE can see total words that have been inserted inside our document and total symbols making those words in our document by clicking on the Word/Symbol icon from the toolbar or from the menu bar. We can also use **Ctrl-W** shortcut key to open a dialog to see the live count of words and symbols that have been placed inside our document or the ones that are being placed live. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"See Word/Symbol Count".**

24. INSERTING TABLES:

WE can create a table inside our document by clicking on the new icon from the toolbar or from the menu bar. We can also use **Ctrl-T** shortcut key to create a table inside our already running document. When we trigger the table function inside our document a dialog appears comprises of information related to inserting a table such as (Number of rows to insert, Number of columns to insert, Cell padding in points and Cell spacing). The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Insert a table in a document".**

25. INSERTING IMAGES:

WE can insert images or multiple images inside our document by clicking on the Insert Image icon from the toolbar or from the menu bar. We can also use **Ctrl-I** shortcut key to insert an image. A file open dialog will appear and we can explore the images to insert. An image can be of types (png, jpeg, gif, bump, etc.). The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Insert Image".**

26. INSERTING BULLETED LISTS:

WE can insert bullet points list in our document by clicking on the Bulleted list icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-B** shortcut key to insert a bullet points list. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Insert Bulleted list".**

27. INSERTING NUMBERED LISTS:

WE can insert numbered list in our document by clicking on the Numbered list icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-N** shortcut key to insert a numbered list. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Insert Numbered list".**

28. SPEECH NOTES

This is a highlighting feature of this software (Notepedia | Word Processor) because it uses Artificial Intelligence. We use google api cloud project and PyAudio to access the microphone connected to our system and use speech_recognition module to convert the audio spoken from the microphone and use google API to recognize the audio being spoken and convert it into text by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-M** shortcut key to insert speech notes. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Speak something in your microphone".**

29. CHOOSING FONT FAMILY:

We can choose number of built-in fonts installed. The fonts will be individual for every text selected unlike notepad which has one single font for a complete document. Font Combo Box is used to select a font from.

30. CHOOSING FONT SIZES:

We can choose font sizes to make our text appear bigger or smaller. The font size will be individual for every text selected unlike notepad which has one single font size for a complete document. Font Spin Size is used to increase or decrease the font size.

31. CHANGE TEXT FOREGROUND COLOR:

We can change the Text Foreground Color in our document by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-F** shortcut

key to open a color picker dialog. The Color will be individual for every text selected. When Font Foreground color function is triggered, the dialog for choosing a color or a color picker dialog will appear and user can select either from the built-in colors or user can make custom colors as well and the swatches will be saved as well. The customized color picked will also show the hex code as well. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Change text Foreground Color".**

32. CHANGE TEXT BACKGROUND COLOR:

We can change the Text Background Color in our document by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-B** shortcut key to open a color picker dialog. The Color will be individual for every text selected. When Font Foreground color function is triggered, the dialog for choosing a color or a color picker dialog will appear and user can select either from the built-in colors or user can make custom colors as well and the swatches will be saved as well. The customized color picked will also show the hex code as well. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Change text Background Color".**

33. MAKE A TEXT APPEAR BOLD:

We can make a text appear bold by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-B** shortcut key to make a text appear bold.

The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear bold”.**

34. MAKE A TEXT APPEAR ITALICIZED:

We can make a text appear italicized by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-I** shortcut key to make a text appear italicized. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear italicized”.**

35. MAKE A TEXT APPEAR UNDERLINED:

We can make a text appear underlined by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-U** shortcut key to make a text appear underlined. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear underlined”.**

36. MAKE A TEXT APPEAR STRIKED-OUT:

We can make a text appear striked-out by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-S** shortcut key to make a text appear striked-out. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a

text is displayed inside a status bar **“Make a text appear striked-out”.**

37. MAKE A TEXT SUPERSCRIPTED:

We can make a text appear superscripted by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-PgUp** shortcut key to make a text appear superscripted. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear superscripted”.**

38. MAKE A TEXT SUBSCRIPTED:

We can make a text appear subscripted by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-PgDn** shortcut key to make a text appear subscripted. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear subscripted”.**

39. ALIGN TEXT:

We can make align a text to the left, to the right, to the center or align justify by clicking on the icons from the toolbar or from the menu bar. We can also use the **Ctrl-L** shortcut key for left aligning the text, **Ctrl-R** shortcut key for right aligning the text, **Ctrl-E** shortcut key for center aligning the text, **Ctrl-J shortcut key** for justify the text aligning. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the

shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a texts are displayed inside a status bar like **“Align a text to the left”** for left aligning **“Align a text to the right”** for right aligning, **“Align a text to the center”** for center aligning and **“Align a text justified”** for justify aligning a text**.**

40. INDENT SELECTED LINES OF TEXTS:

We can make our selected text indented in a document by clicking on the indent icon available in the format bar for in the menu bar or we can also use the shortcut **Ctrl-Tab** to make a selected text indented. If we have the selection enabled then it will count how many lines we have selected and then according to these lines it will place a tab on start of each line. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text indented”.**

41. DEDENT SELECTED LINES OF TEXT:

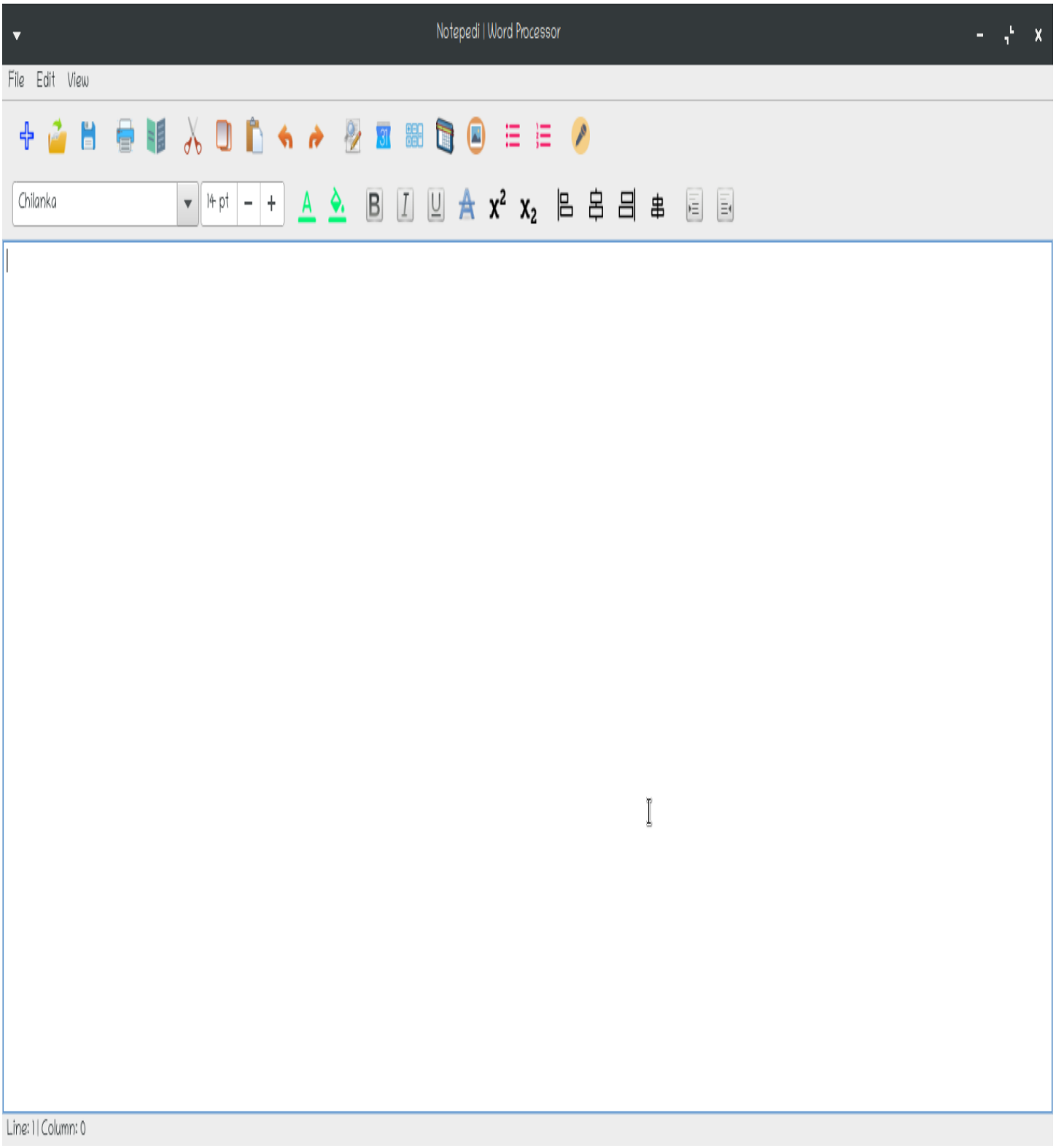
We can make our selected text dedented in a document by clicking on the dedent icon available in the format bar for in the menu bar or we can also use the shortcut **Ctrl-Shift-Tab** to make a selected text dedented. If we have the selection enabled then it will count how many lines we have selected and then according to these lines it will remove a tab on start of each line. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text dedented”.**

42. DRAG AND DROP UI ELEMENTS

A Special functionality that is not available in most of the word processing or text processing software is the drag and drop UI elements we can make use of these drag and drop UI elements to drop the format bar to whatever position we want it to be placed and we can even place the the format bar, menu bar or toolbar outside of our software anywhere we want and that is the power of drag and drop UI elements and if the icons are not showing properly due to space limitations then it will show an arrow that we can click and show rest of the icons. A highlighter will appear whenever we want to drop the UI element inside our software these kind of functionalities are normally available in professional-grade softwares like Photoshop or Adobe Illustrator where we are given full use case on how well we can customize the software and now this functionality is available in Notepedia word processor. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut.

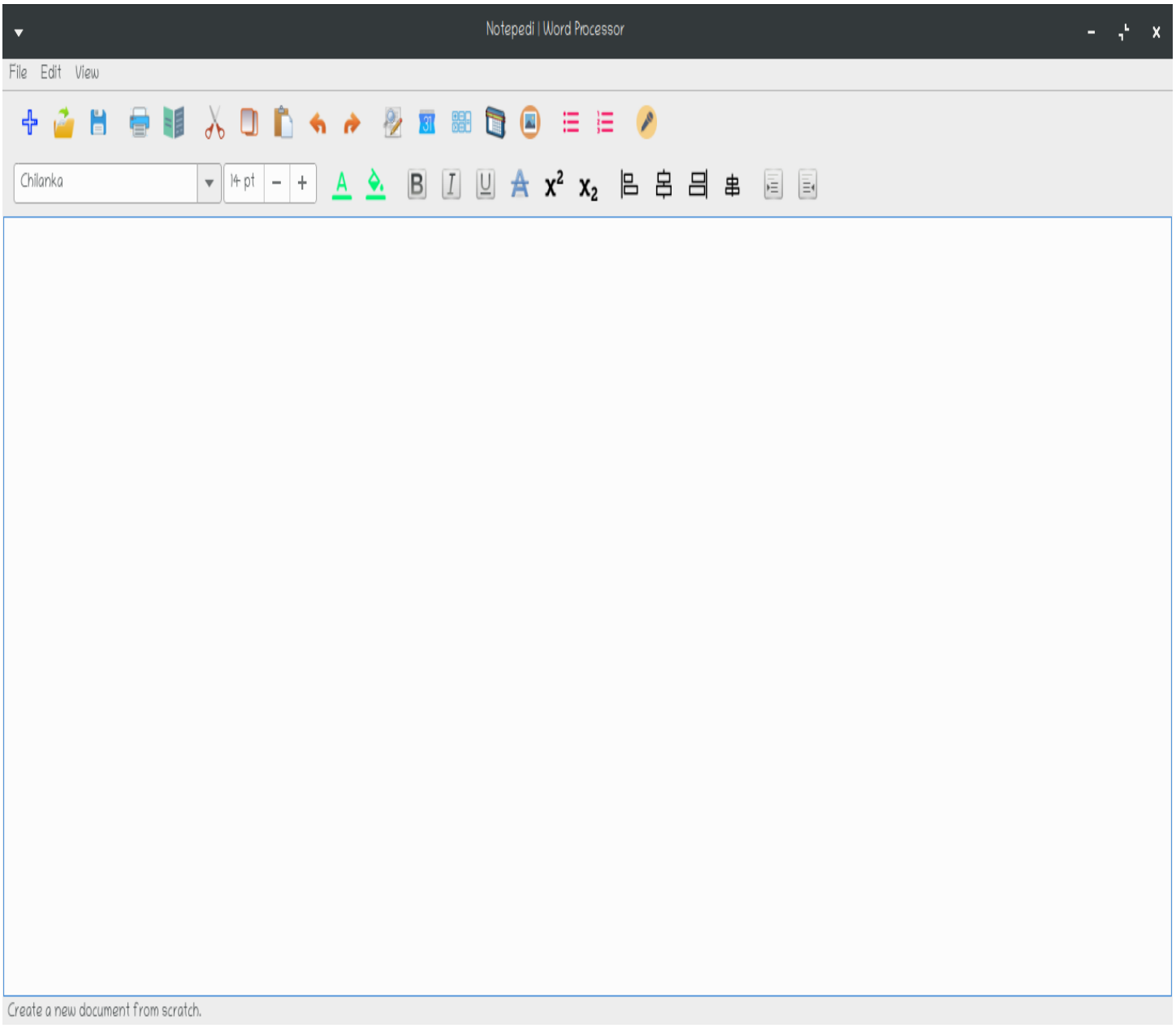
IMPLEMENTATION

We implemented these features inside our software and the working and implementation of these features are described below:



1. NEW FILE:

We can make multiple instances of our document by clicking on the new icon from the toolbar or from the menu bar. We can also use **Ctrl-N** shortcut key to create an instance of our already running document and work on multiple files at the same time. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Create a new document from Scratch”.**

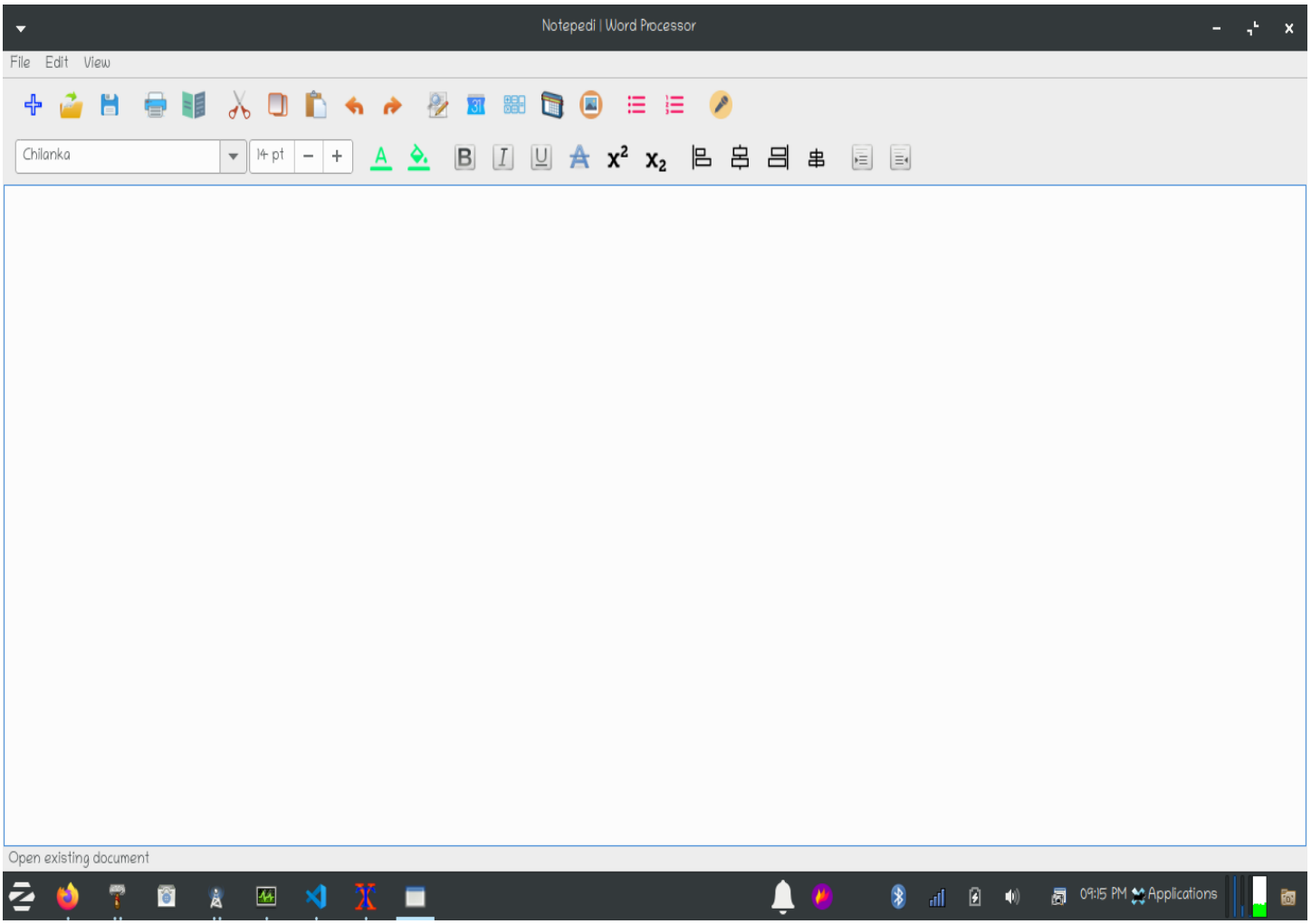


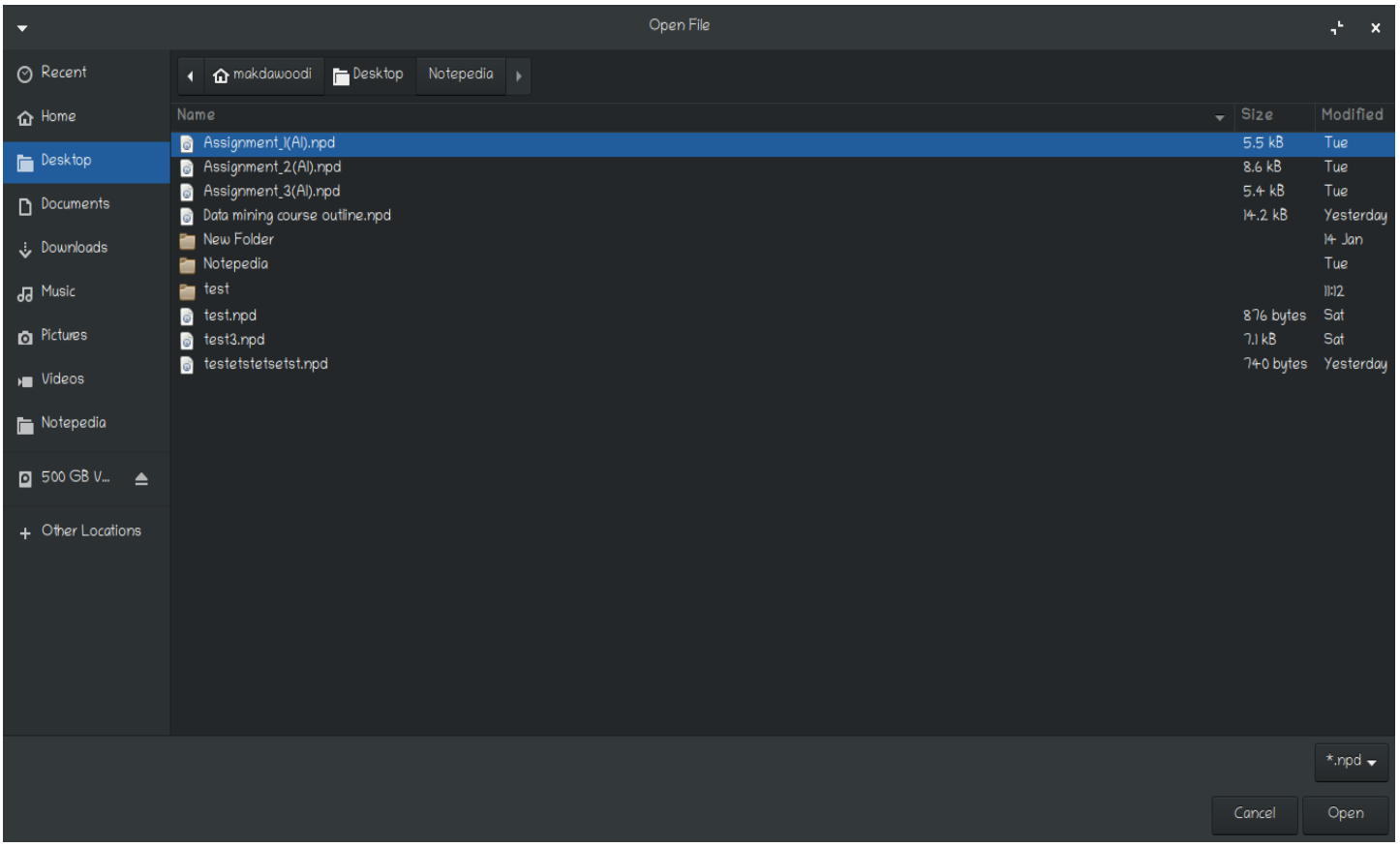
```
def new(self):
    spawn = Main()
    spawn.show()
```

2. OPEN FILE:

We can open multiple instances of new already saved documents by clicking on the open icon from the toolbar or from the menu bar. We can also use **Ctrl-O** shortcut key to open an already save document work on multiple files at the same time. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Open Existing Document”.**

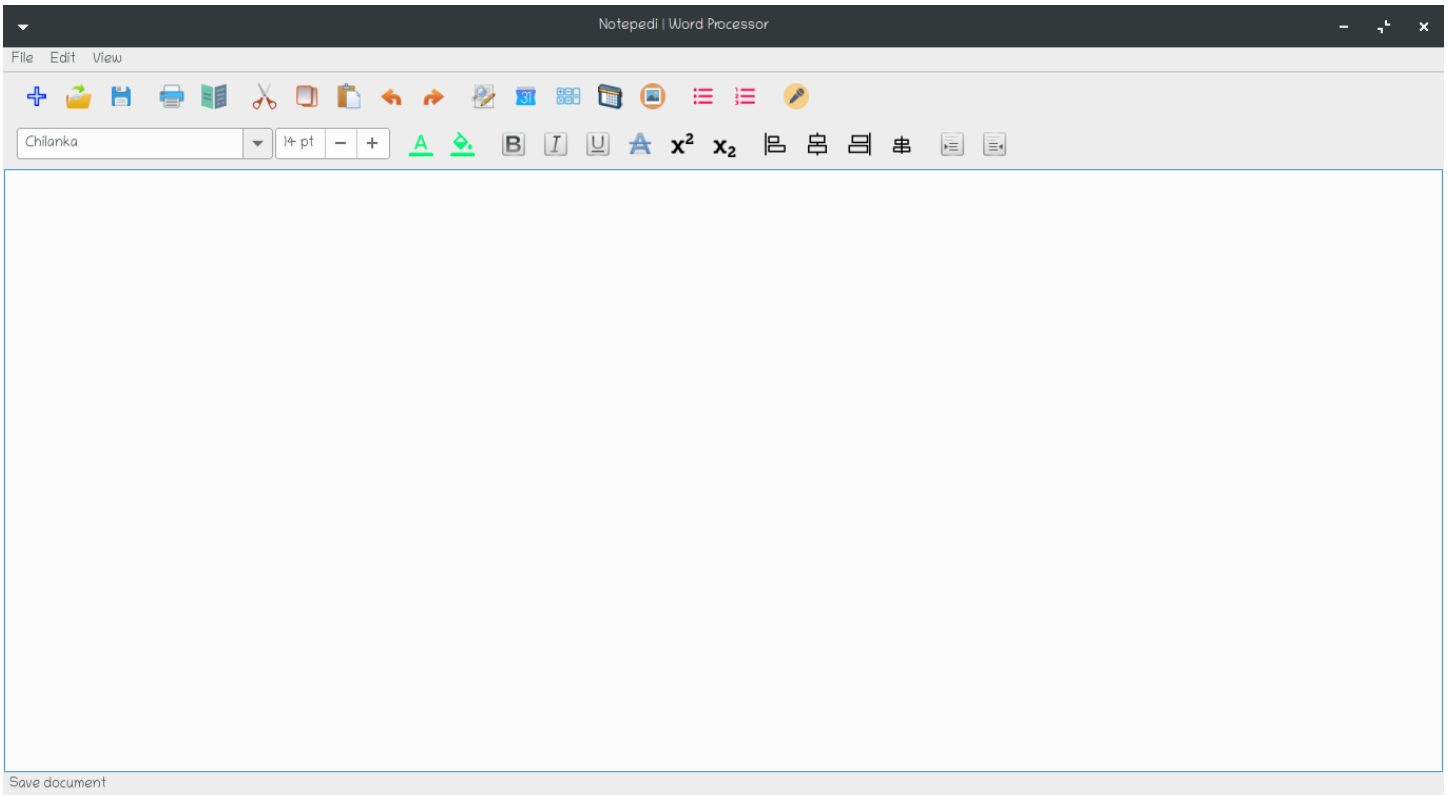
```
def open(self):

    # Get filename and show only .npd files
    self.filename = QtWidgets.QFileDialog.getOpenFileName(self, 'Open File',".","(*.npd)")[0]

    if self.filename:
        with open(self.filename,"rt") as file:
            self.text.setText(file.read())
```

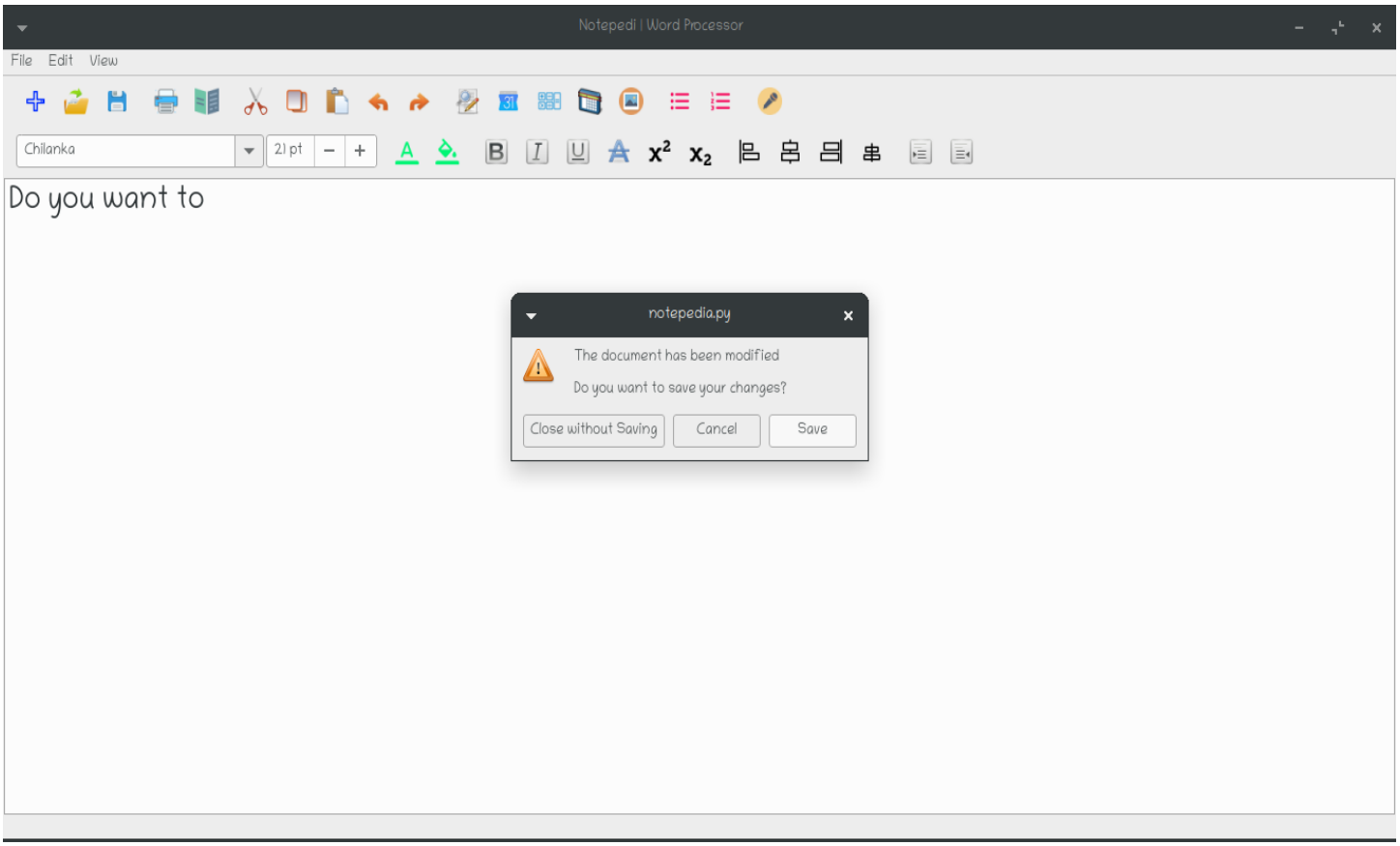

Notepedi | Word Processor
File Edit View
Chilanka
14 pt
Open existing document
09:15 PM Applications


Open File
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Notepedia
500 GB V...
Other Locations
makdawoodi
Desktop
Notepedia
Name
Size
Modified
Assignment_1(AI).npd 5.5 kB Tue
Assignment_2(AI).npd 8.6 kB Tue
Assignment_3(AI).npd 5.4 kB Tue
Data mining course outline.npd 14.2 kB Yesterday
New Folder 14 Jan
Notepedia Tue
test 11:12
test.npd 876 bytes Sat
test3.npd 7.1 kB Sat
testetstetsetst.npd 740 bytes Yesterday
*.npd
Cancel
Open

3. SAVE FILE:

We can make save our document by clicking on the save icon from the toolbar or from the menu bar. We can also use **Ctrl-S** shortcut key to save our already running document. If the file name is not given already our document when we trigger the save icon will open a getSaveFileNameDialog to tell where the document should be saved. It opens a file explorer to save our document and if the name of a file is already decided then the file is just saved and overrited. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Save document”.**

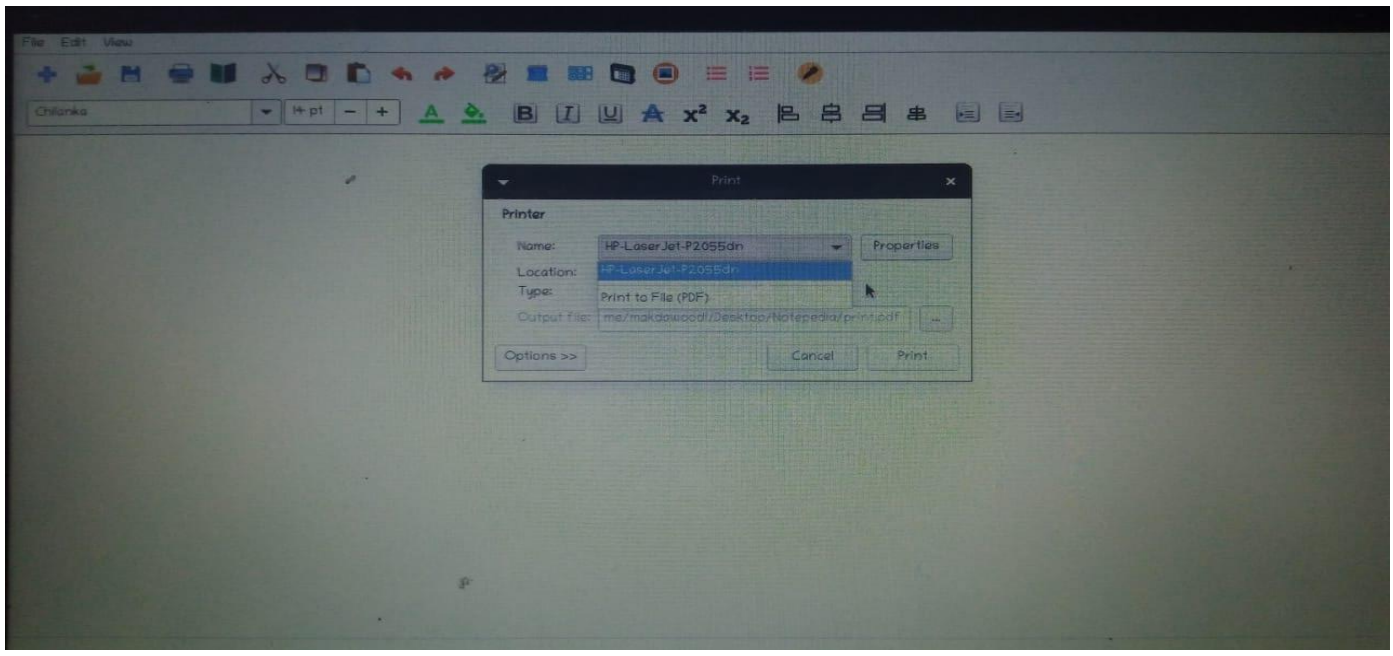


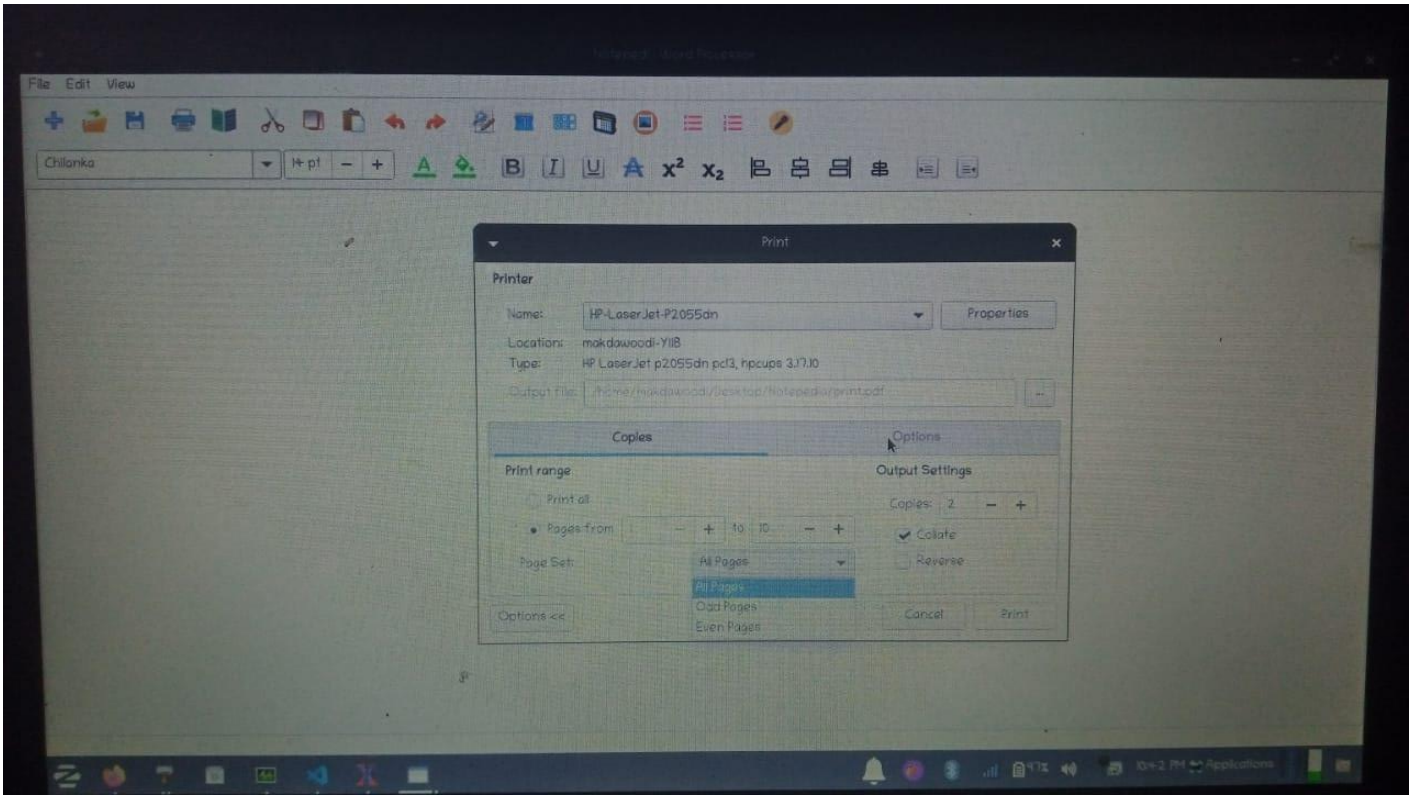


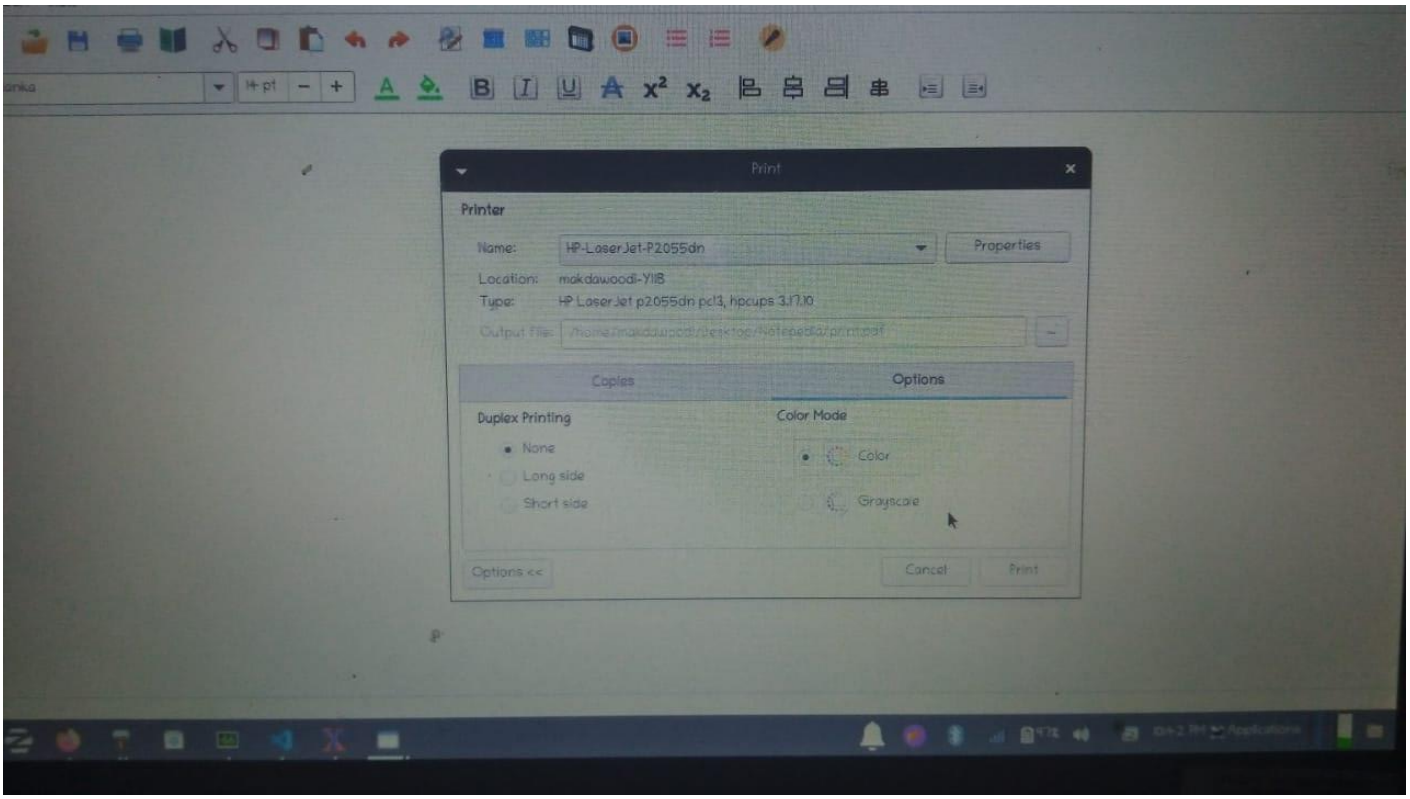

```
def save(self):

    # Only open dialog if there is no filename yet
    if not self.filename:
        self.filename = QtWidgets.QFileDialog.getSaveFileName(self, 'Save File')[0]

    if self.filename:

        # Append extension if not there yet
        if not self.filename.endswith(".npd"):
            self.filename += ".npd"

        # We just store the contents of the text file along with the
        # format in html, which Qt does in a very nice way for us
        with open(self.filename,"wt") as file:
            file.write(self.text.toHtml())

        self.changesSaved = True
```

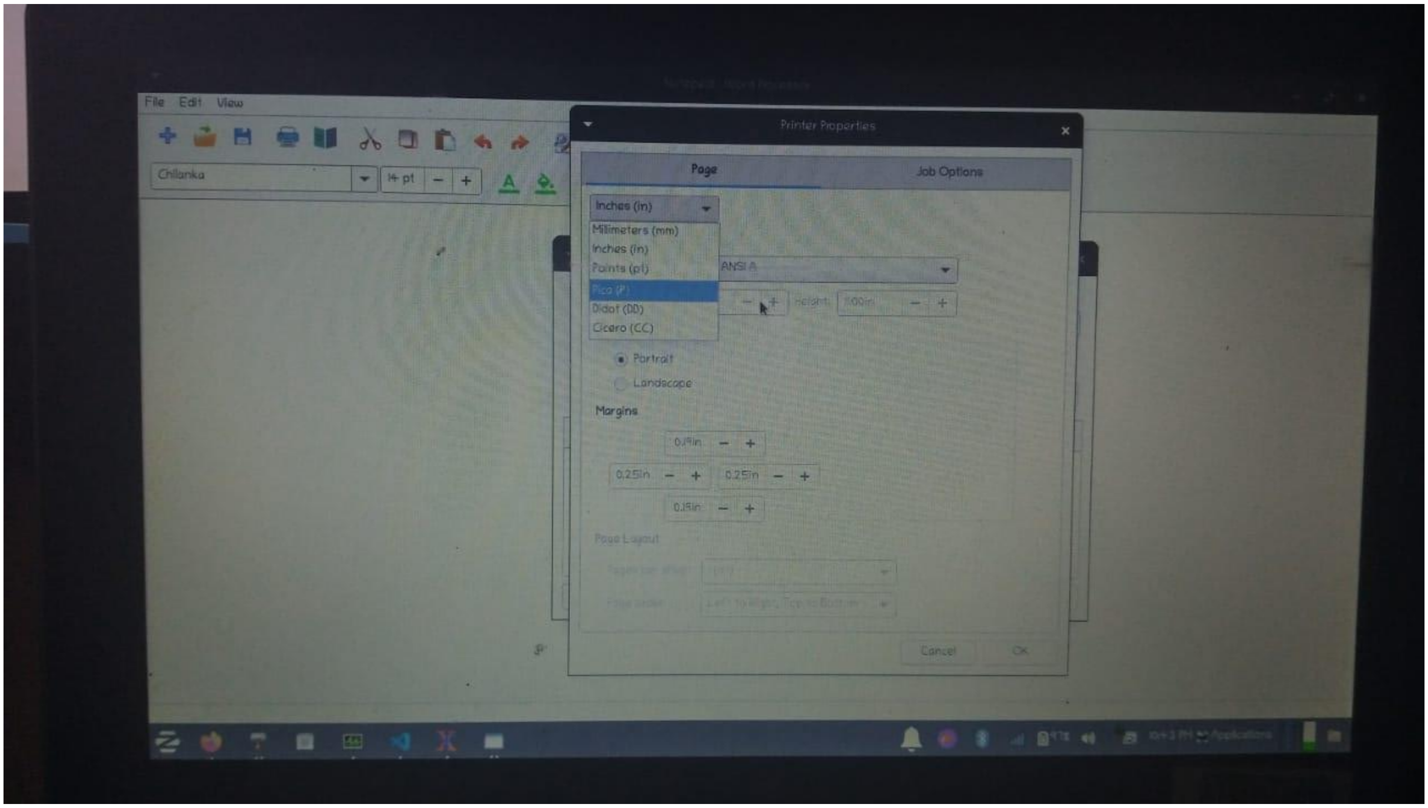
4. PRINT A FILE:

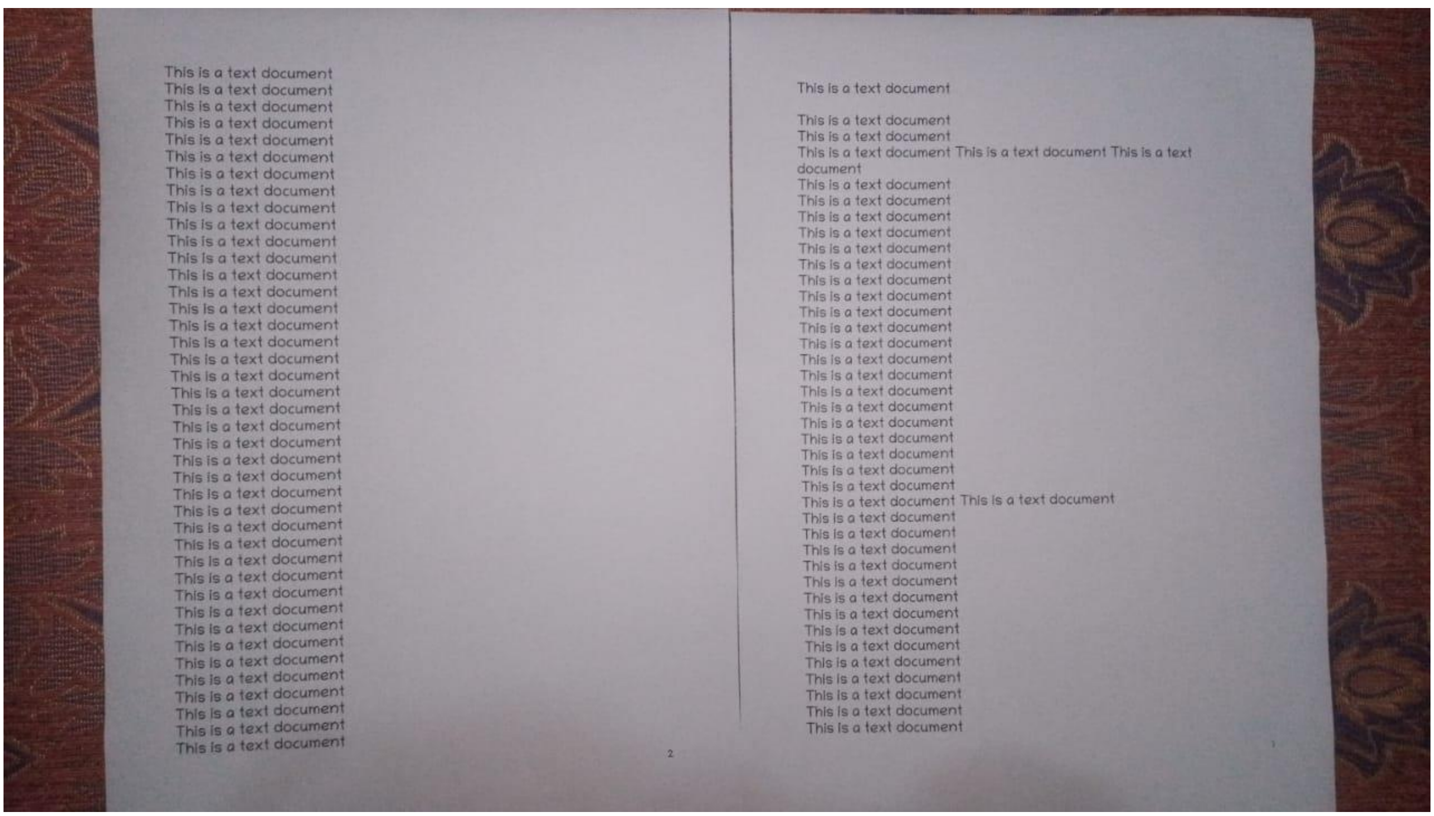
We can print our document by clicking on the print icon from the toolbar or from the menu bar. We can also use **Ctrl-P** shortcut key to print our document. If the printer is connected then it will get the information regarding the printer, the model and version of a printer and shows in the recognized hardware and a button is provided inside a dialog so that print action can be performed. An Options button is also provided and when clicked it opens a new dialog which is used to select a directory for converting text to pdf and number of copies to print and the ranges and order of printing multiple pages in a document and hence contains 2 tabs (Copies) and (Options) tab and we can print a document in a colored manner or in a grayscale manner. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Print document".**




File Edit View
Chilanka
14 pt
Print
Printer
Name: HP-LaserJet-P2055dn
Properties
Location: makdawoodi-YII8
Type: HP LaserJet p2055dn pcl3, hpcups 3.17.10
Output File: /home/makdawoodi/Desktop/Notepadqq/print.pdf
Copies
Options
Print range
Print all
Pages from 1 to 10
Page Set: All Pages
All Pages
Odd Pages
Even Pages
Output Settings
Copies: 2
Collate
Reverse
Options <<
Cancel
Print


Print
Printer
Name: HP-LaserJet-P2055dn
Properties
Location: makdawoodi-YII8
Type: HP LaserJet p2055dn pcl3, hpcups 3.17.10
Output File: /home/makdawoodi/Desktop/Notepadqq/print.pdf
Copies
Options
Duplex Printing
None
Long side
Short side
Color Mode
Color
Grayscale
Options <<
Cancel
Print




This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document

2

This is a text document

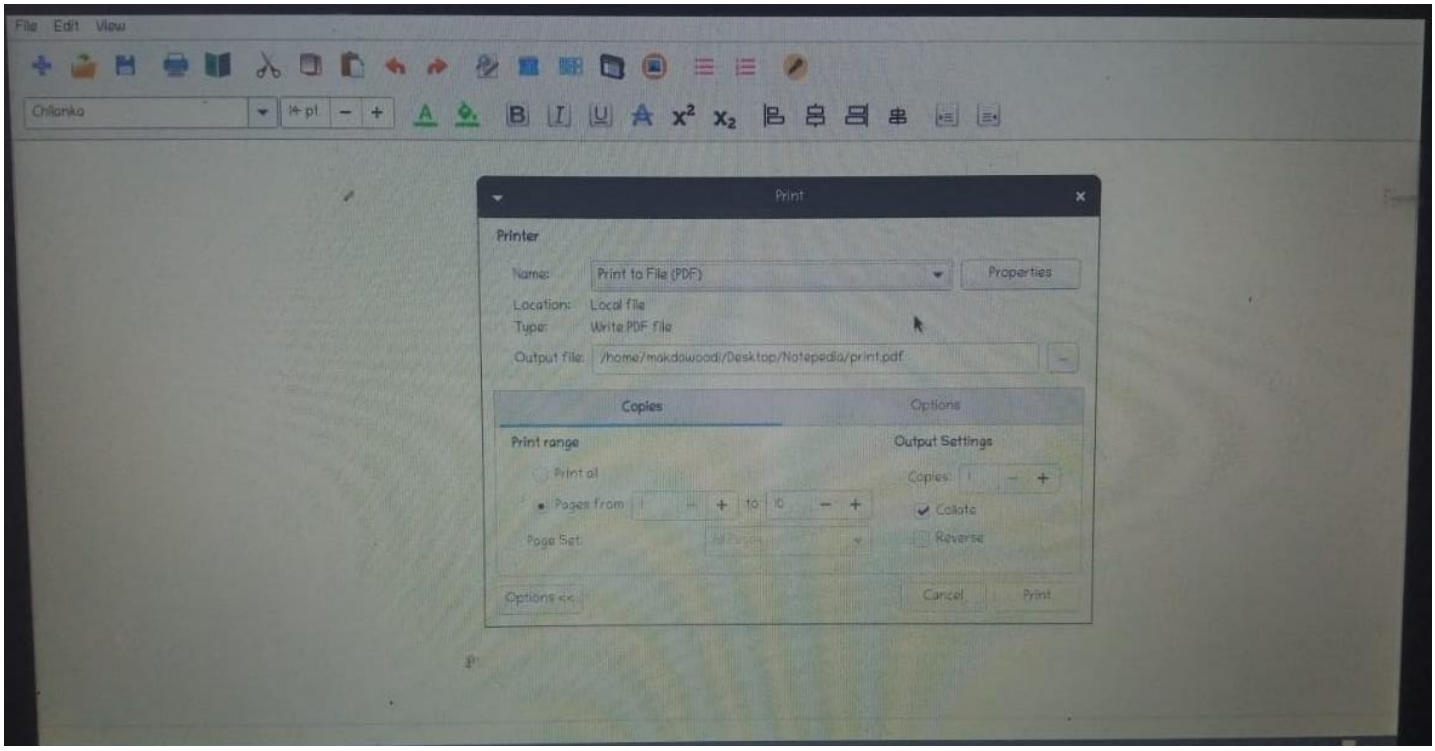
This is a text document
This is a text document
This is a text document This is a text document This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document
This is a text document

1

```
def printHandler(self):

    # Open printing dialog
    dialog = QtPrintSupport.QPrintDialog()

    if dialog.exec_() == QtWidgets.QDialog.Accepted:
        self.text.document().print_(dialog.printer())
```

5. TEXT TO PDF CONVERTER:

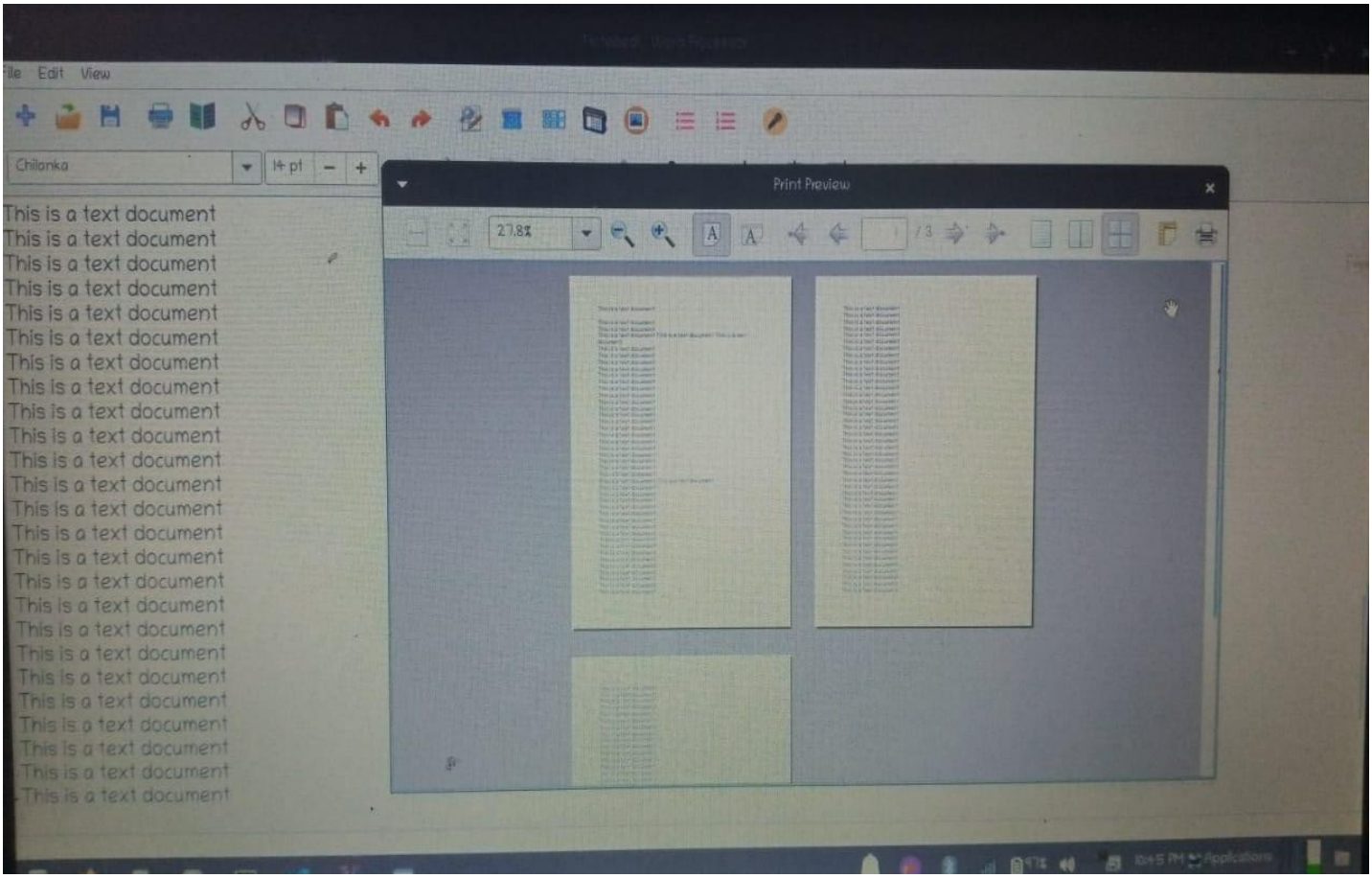
We can print our document by clicking on the print icon from the toolbar or from the menu bar. We can also use **Ctrl-P** shortcut key to print our document. If the printer is connected then it will get the information regarding the printer, the model and version of a printer and shows in the recognized hardware and a button is provided inside a dialog so that print action can be performed. An Options button is also provided and when clicked it opens a new dialog which is used to select a directory for converting text to pdf. We can convert our text to pdf directly from Notepedia software and hence use it to read on any kind of device that is working in your area. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about. We have to select a path where the converted document will be saved and then hit the print button to print a document in a pdf readable format.




```python
def preview(self):

    # Open preview dialog
    preview = QtPrintSupport.QPrintPreviewDialog()

    # If a print is requested, open print dialog
    preview.paintRequested.connect(lambda p: self.text.print_(p))

    preview.exec_()

def printHandler(self):

    # Open printing dialog
    dialog = QtPrintSupport.QPrintDialog()

    if dialog.exec_() == QtWidgets.QDialog.Accepted:
        self.text.document().print_(dialog.printer())
```

6. PREVIEW PAGES BEFORE PRINTING:

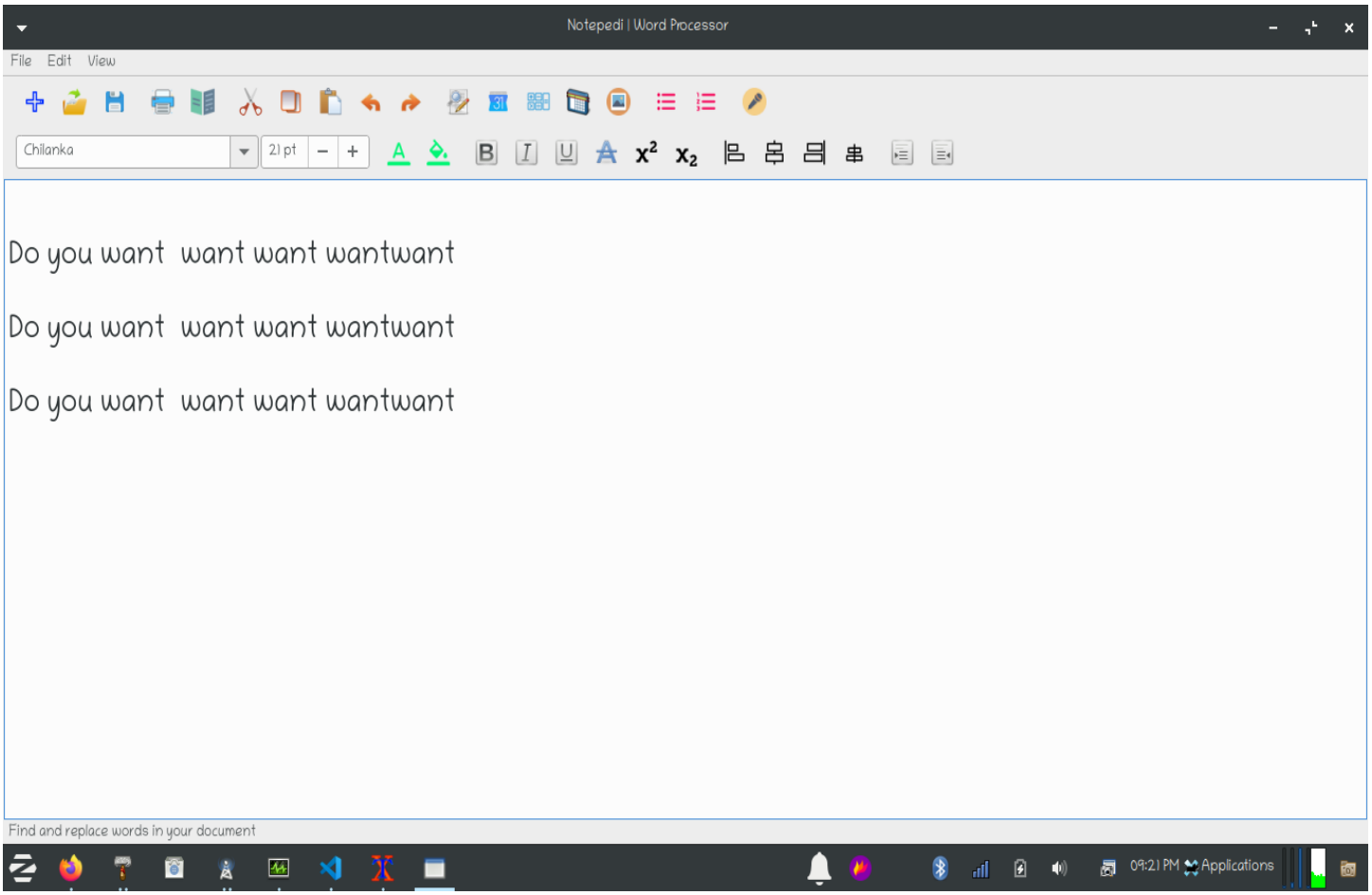
We can preview our document prior to printing by clicking on the preview icon from the toolbar or from the menu bar. We can also use **Ctrl-Shift-P** shortcut key to preview our document before printing. A separate dialog will appear containing the information of how the document can be viewed (A landscape style) or (A portrait style) and Zoom in and Out a document and different pages that can be viewed directly inside a preview document dialog and how the pages can be viewed (Either they can be viewed as a single page or multiple pages) and the measurements of how the document should be printed and what will be the measurements while printing (Either in inches, points, millimeters, centimeters). And it also provides an icon to print direcly from the preview dialog and then the print dialog will appear and then we can print the document directly from here. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about. We have to select a path where the converted document will be saved and then hit the print button to print a document in a pdf readable format. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Preview document before printing”.**

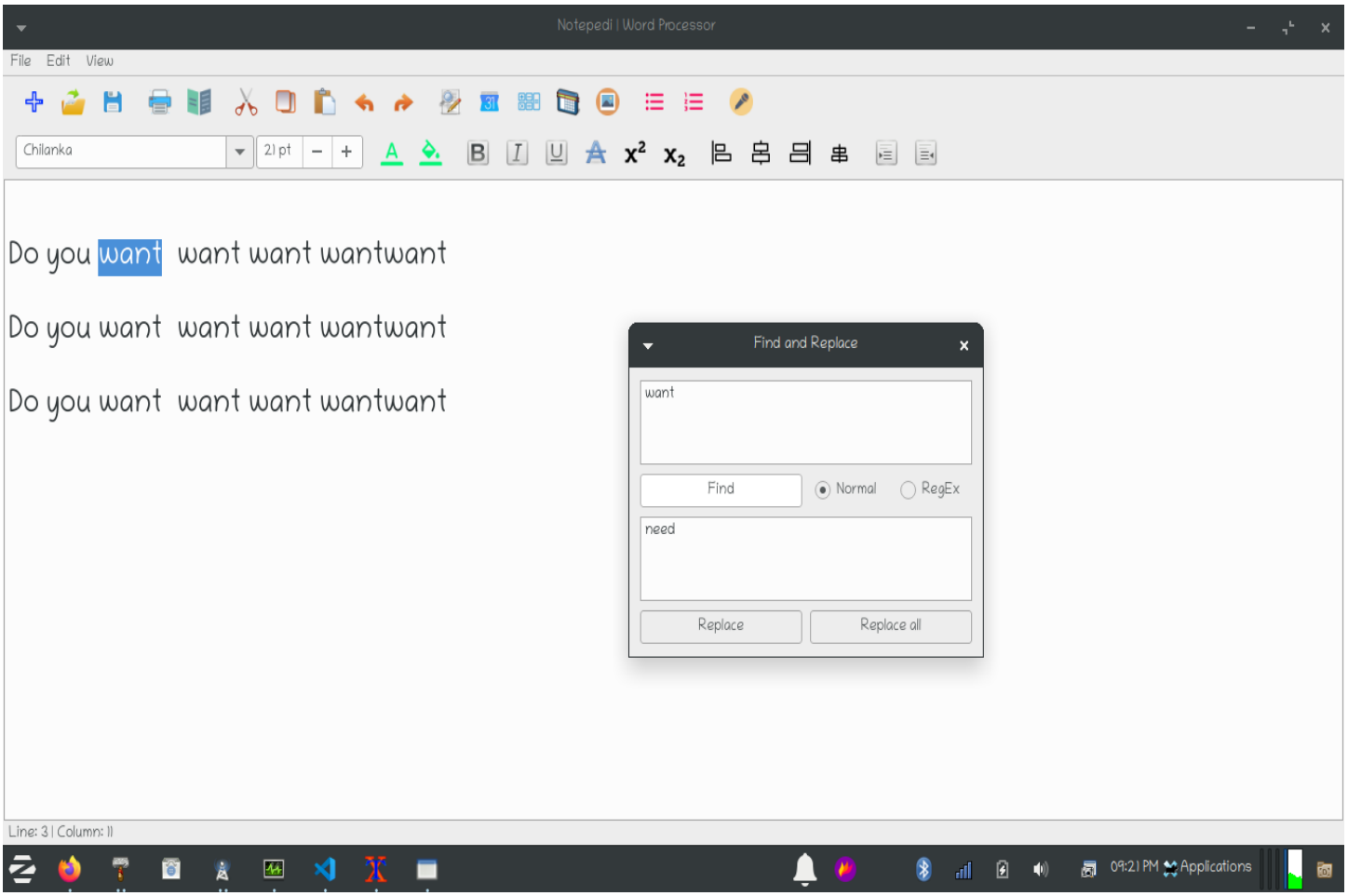


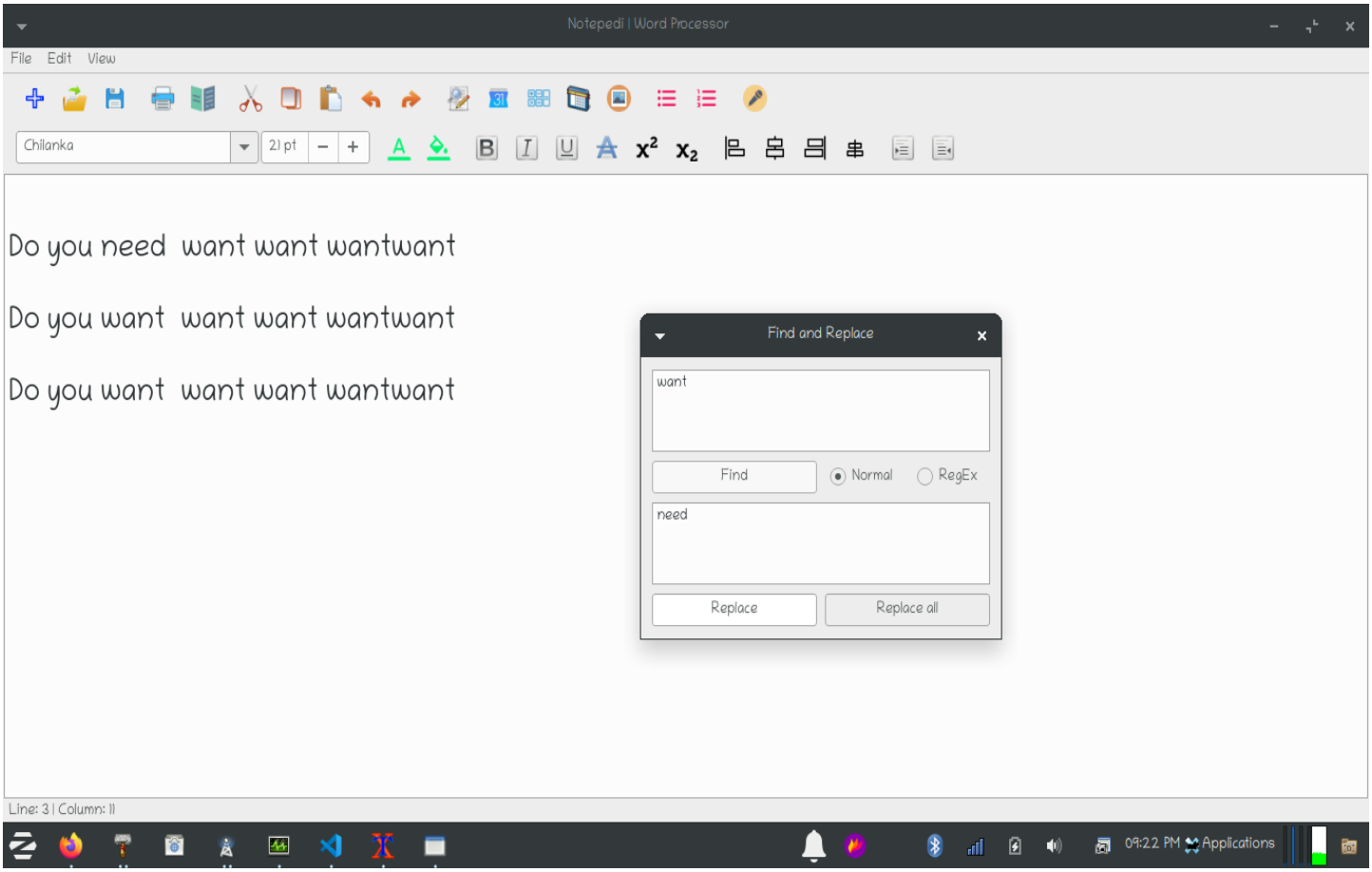

```
def preview(self):

    # Open preview dialog
    preview = QtPrintSupport.QPrintPreviewDialog()

    # If a print is requested, open print dialog
    preview.paintRequested.connect(lambda p: self.text.print_(p))

    preview.exec_()

def printHandler(self):

    # Open printing dialog
    dialog = QtPrintSupport.QPrintDialog()

    if dialog.exec_() == QtWidgets.QDialog.Accepted:
        self.text.document().print_(dialog.printer())
```

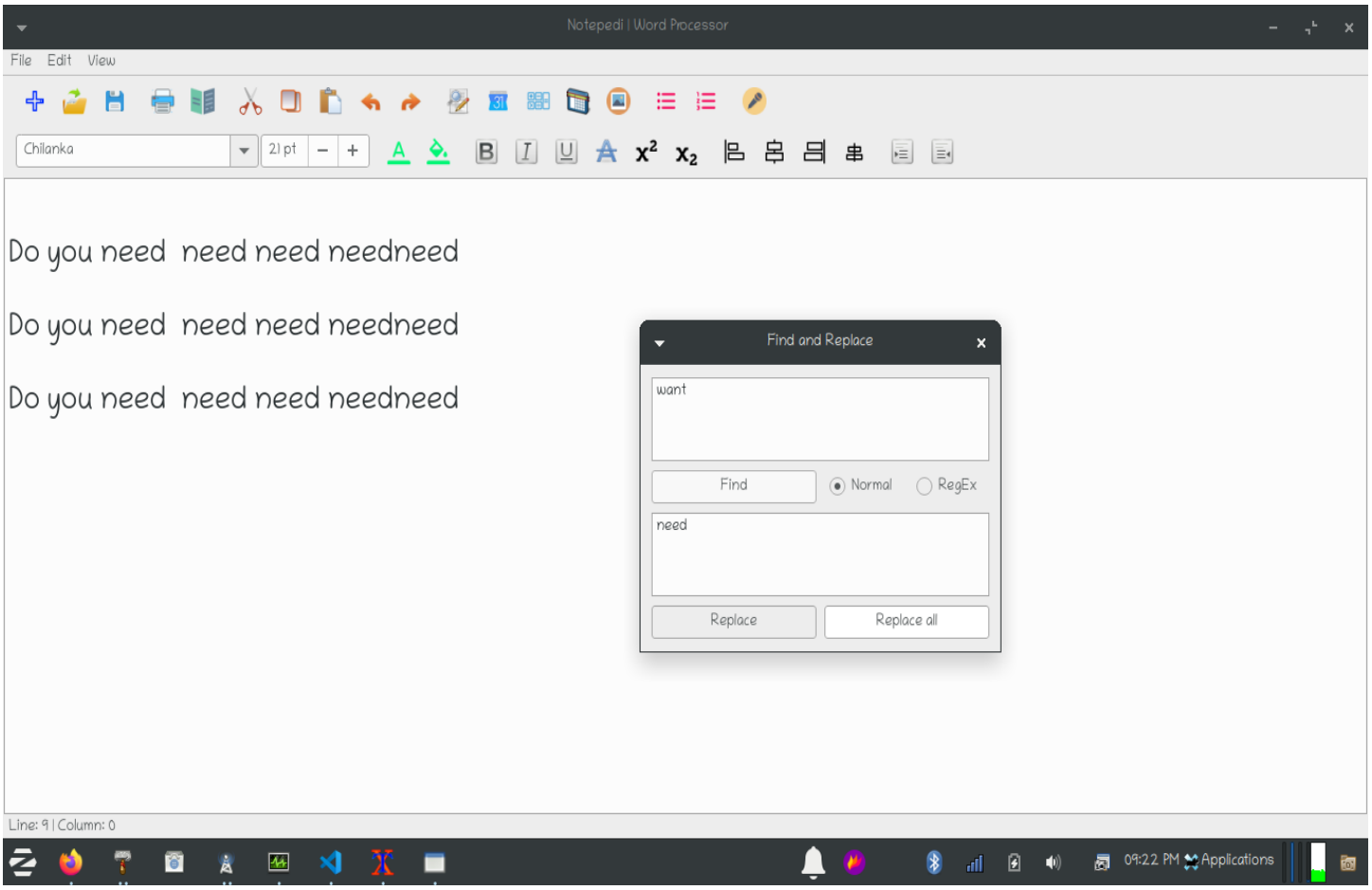
7. FIND AND REPLACE:

WE can find the particular text if our document is very large and we can replace the text as well by clicking on the Find and Replace icon from the toolbar or from the menu bar. We can also use **Ctrl-F** shortcut key to open find and replace dialog so that we can find particular word inside our document or words and we can replace them either manually or by using a replace function inside find and replace dialog. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Find and replace words in your document”.**




Notepedi | Word Processor
File Edit View
Chilanka
21 pt
Do you want want want wantwant
Do you want want want wantwant
Do you want want want wantwant
Find and Replace
want
Find
Normal
RegEx
need
Replace
Replace all
Line: 3 | Column: 11
09:21 PM
Applications


Notepedi | Word Processor
File Edit View
Chilanka
21 pt
Do you need want want wantwant
Do you want want want wantwant
Do you want want want wantwant
Find and Replace
want
Find
Normal
RegEx
need
Replace
Replace all
Line: 3 | Column: 11
09:22 PM
Applications




```
def find(self):

    # Grab the parent's text
    text = self.parent.text.toPlainText()

    # And the text to find
    query = self.findField.toPlainText()

    if self.normalRadio.isChecked():

        # Use normal string search to find the query from the
        # last starting position
        self.lastStart = text.find(query,self.lastStart + 1)

        # If the find() method didn't return -1 (not found)
        if self.lastStart >= 0:

            end = self.lastStart + len(query)
```

```
def replace(self):

    # Grab the text cursor
    cursor = self.parent.text.textCursor()

    # Security
    if cursor.hasSelection():

        # We insert the new text, which will override the selected
        # text
        cursor.insertText(self.replaceField.toPlainText())

        # And set the new cursor
        self.parent.text.setTextCursor(cursor)

def replaceAll(self):

    self.lastStart = 0

    self.find()

    # Replace and find until self.lastStart is 0 again
    while self.lastStart:
        self.replace()
        self.find()

def moveCursor(self,start,end):

    # We retrieve the QTextCursor object from the parent's QTextEdit
```

8. INSERTING DATE/TIME:

WE can Insert current date and time by clicking on the date time icon from the toolbar or from the menu bar. We can also use **Ctrl-D** shortcut key to insert current date and time and when the dialog is opened we can choose multiple formats of displaying either date or time or both in multiple fashions. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Insert current date/time”.**

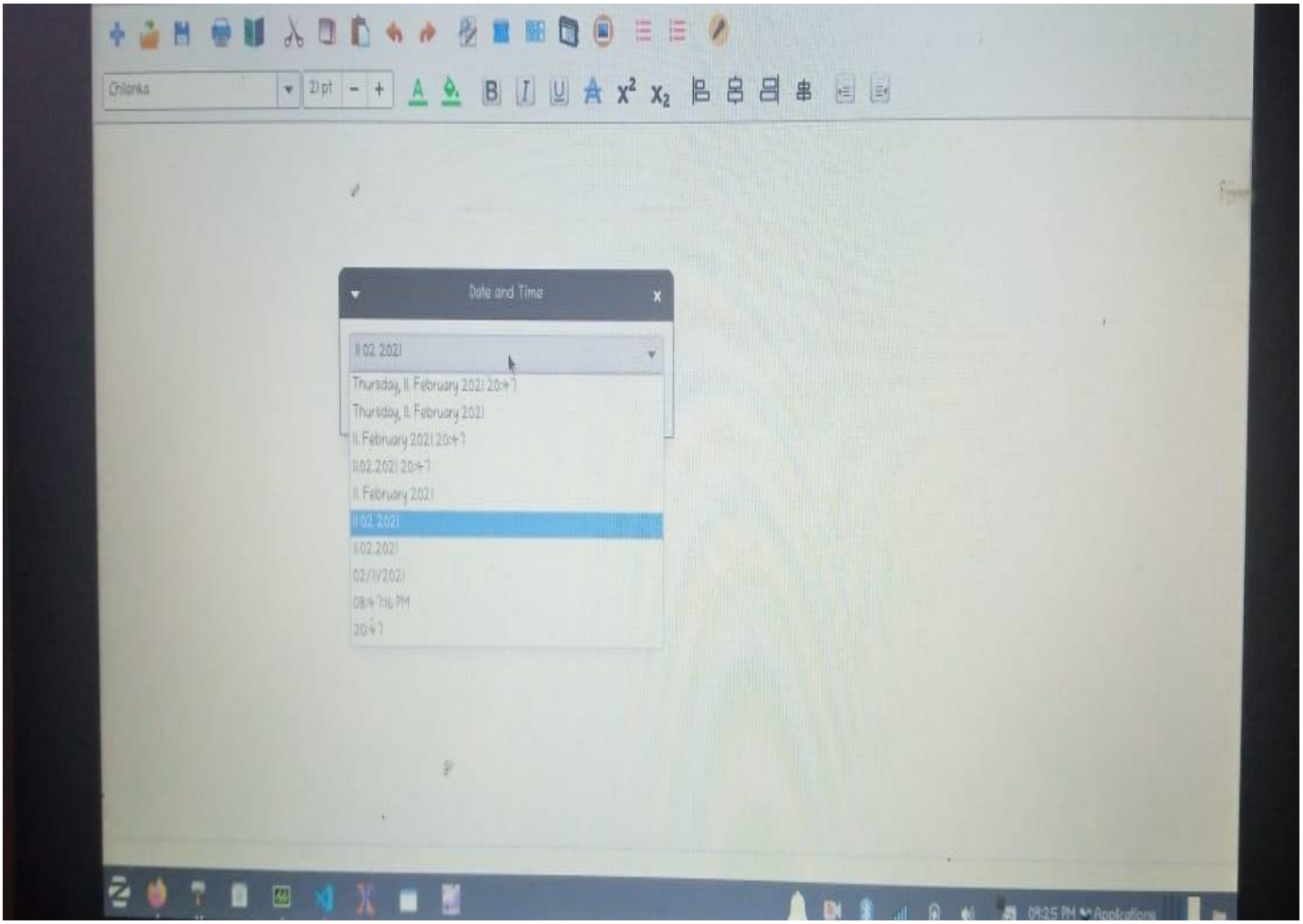
Notepadi | Word Processor

File Edit View

Chilanka 21 pt

Insert current date/time

09:23 PM Applications

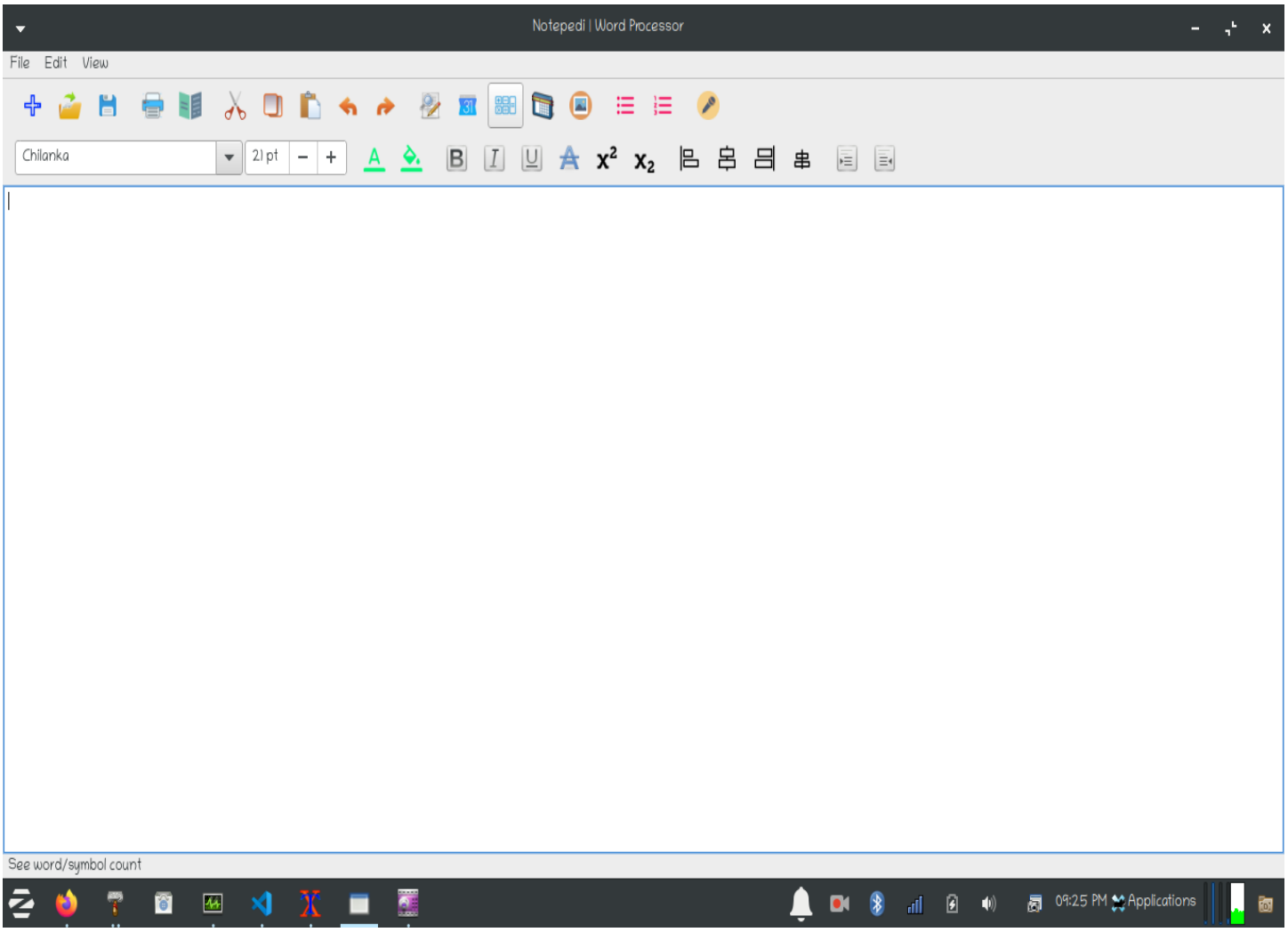


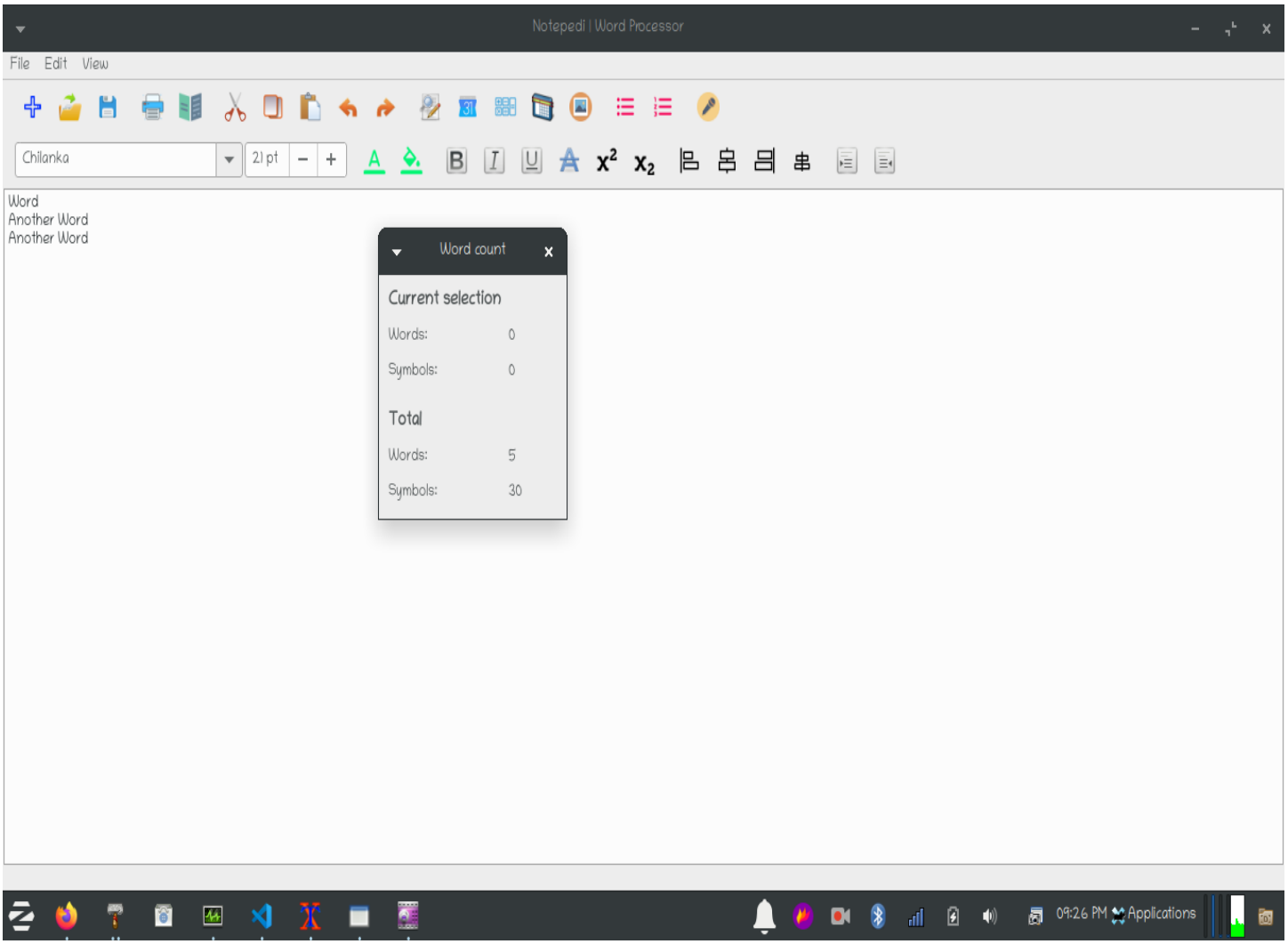
```
#PYQT5 PyQt4's QtGui module has been split into PyQt5's QtGui, QtPrintSupport and QtWidgets modules

from PyQt5 import QtWidgets
#PYQT5 QDialog, QPushButton, QGridLayout, QComboBox

from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt

from time import strftime

class DateTime(QtWidgets.QDialog):
    def __init__(self,parent = None):
        QtWidgets.QDialog.__init__(self, parent)

        self.parent = parent

        self.formats = ["%A, %d. %B %Y %H:%M",
                        "%A, %d. %B %Y",
                        "%d. %B %Y %H:%M",
                        "%d.%m.%Y %H:%M",
                        "%d. %B %Y",
                        "%d %m %Y",
                        "%d.%m.%Y",
                        "%x",
                        "%X",
                        "%H:%M"]

        self.initUI()
```

9. WORD/SYMBOLS COUNTER:

WE can see total words that have been inserted inside our document and total symbols making those words in our document by clicking on the Word/Symbol icon from the toolbar or from the menu bar. We can also use **Ctrl-W** shortcut key to open a dialog to see the live count of words and symbols that have been placed inside our document or the ones that are being placed live. The related Icon is also used to indicate a user of what action will be performed when icon is

triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“See Word/Symbol Count”.**

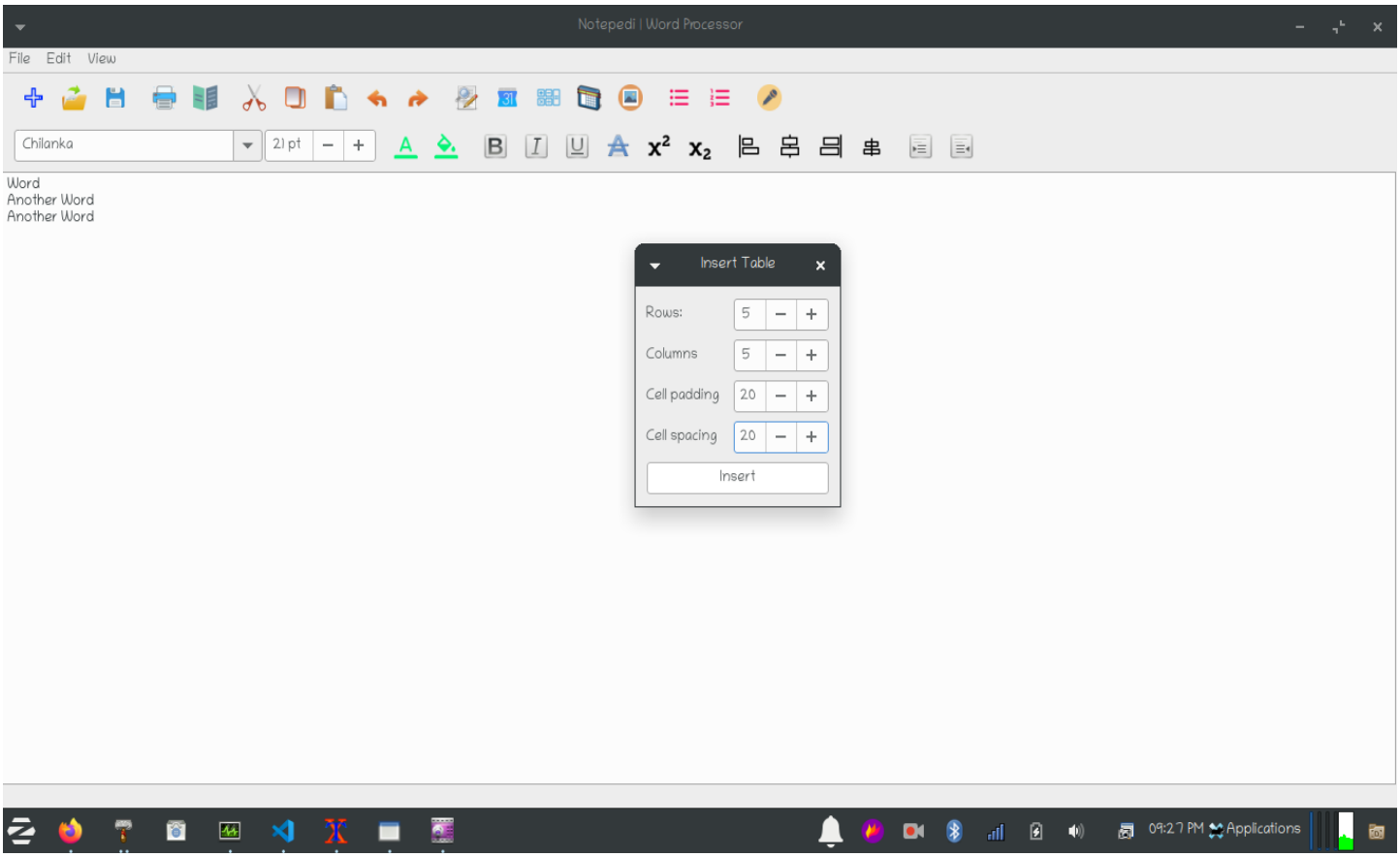


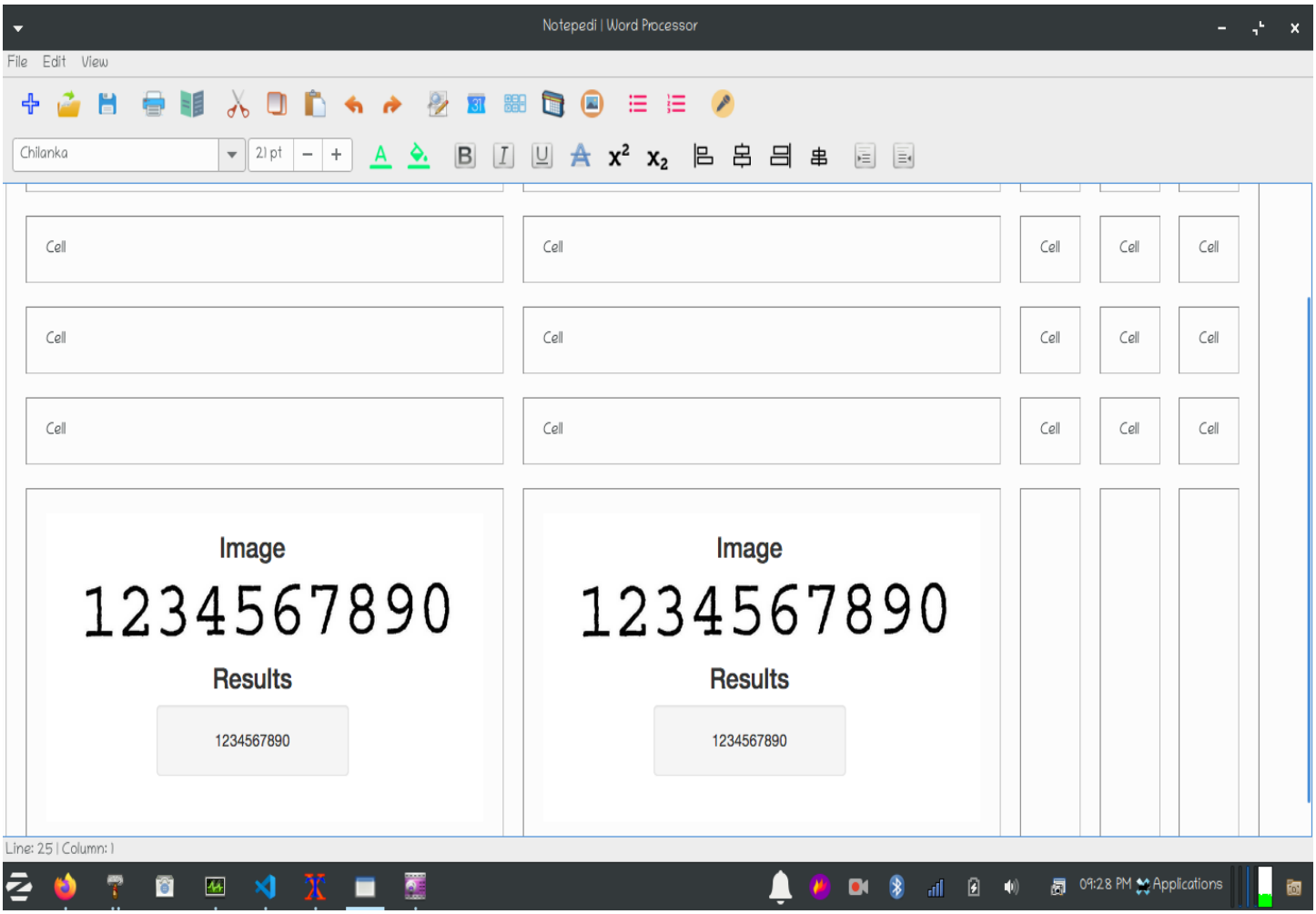



```
    def getText(self):

        # Get the text currently in selection
        text = self.parent.text.textCursor().selectedText()

        # Split the text to get the word count
        words = str(len(text.split()))

        # And just get the length of the text for the symbols
        # count
        symbols = str(len(text))

        self.currentWords.setText(words)
        self.currentSymbols.setText(symbols)

        # For the total count, same thing as above but for the
        # total text

        text = self.parent.text.toPlainText()

        words = str(len(text.split()))
        symbols = str(len(text))

        self.totalWords.setText(words)
        self.totalSymbols.setText(symbols)
```

10. INSERTING TABLES:

WE can create a table inside our document by clicking on the new icon from the toolbar or from the menu bar. We can also use **Ctrl-T** shortcut key to create a table inside our already running document. When we trigger the table function inside our document a dialog appears comprises of information realted to inserting a table such as (Number of rows to insert, Number of columns to insert, Cell padding in points and Cell spacing). The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Insert a table in a document".**

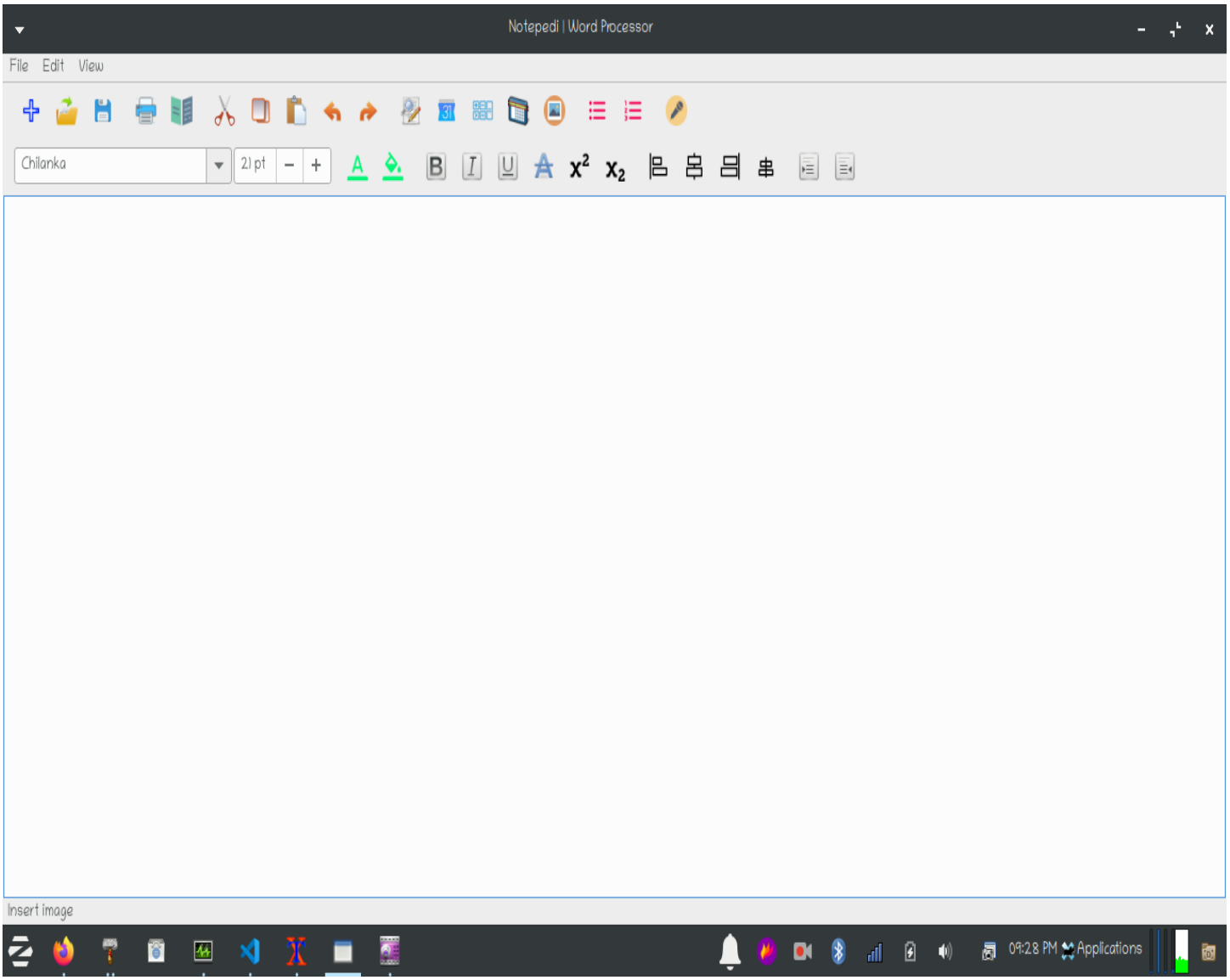


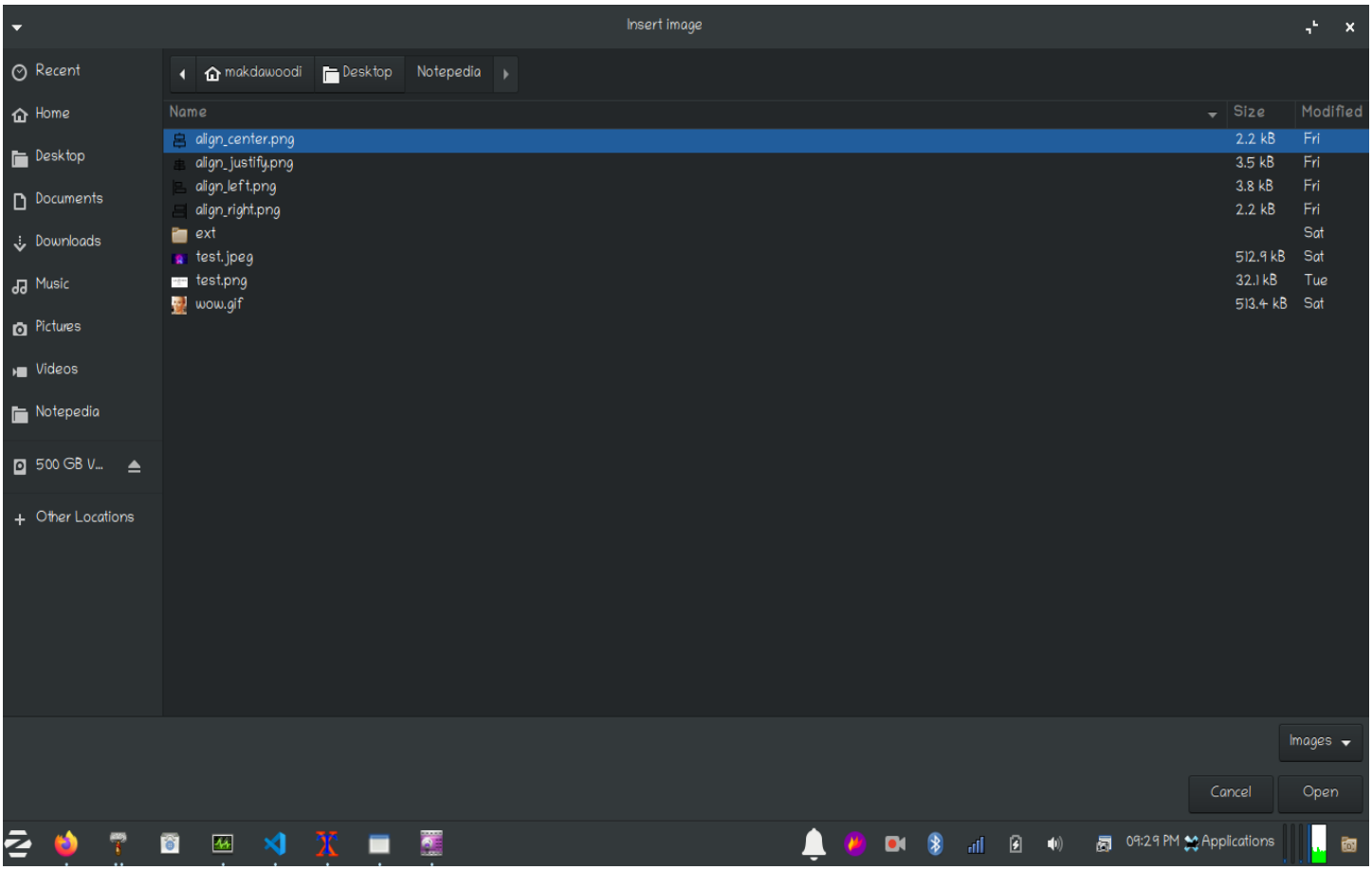


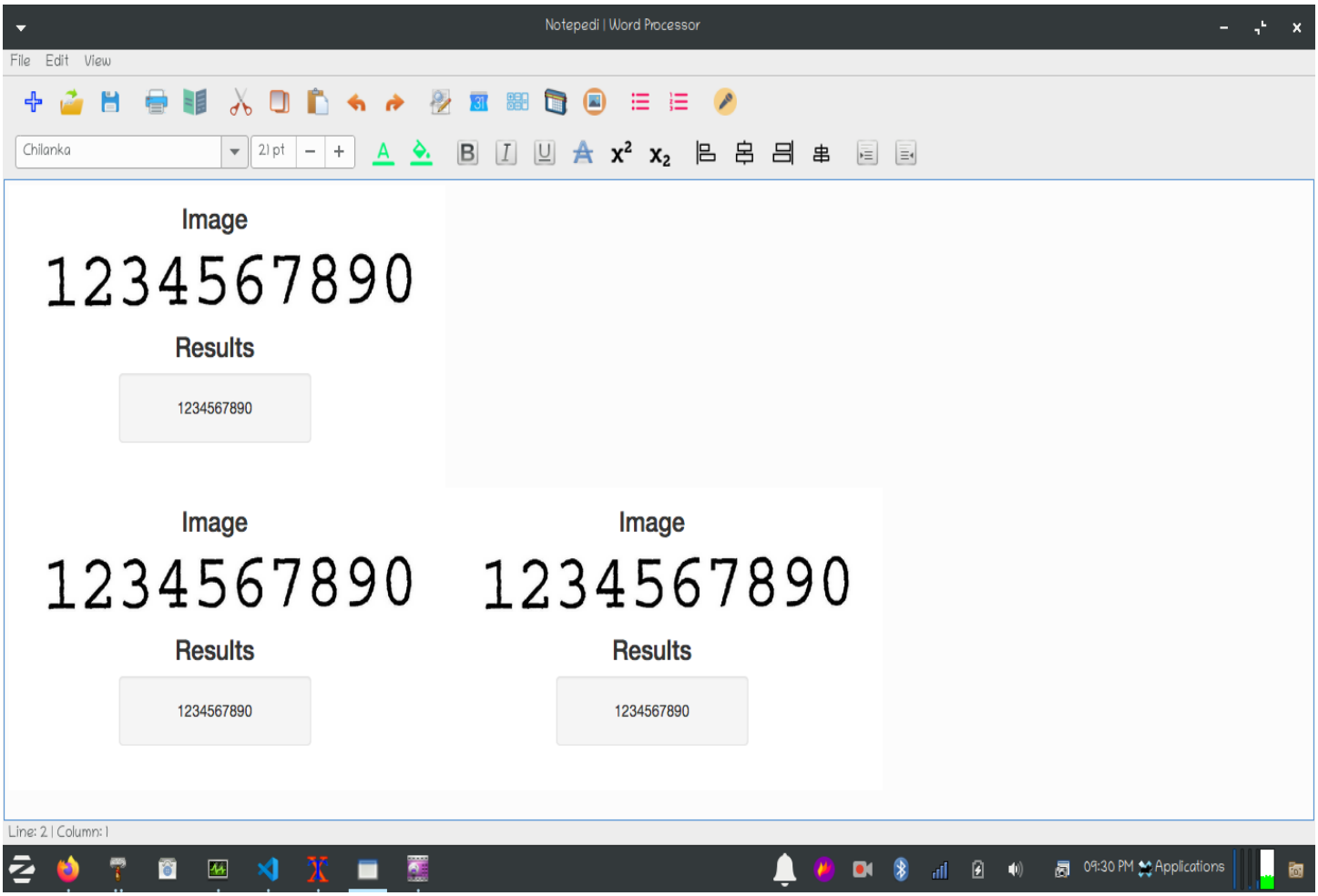

```
def insert(self):

    cursor = self.parent.text.textCursor()

    # Get the configurations
    rows = self.rows.value()

    cols = self.cols.value()

    if not rows or not cols:

        popup = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Warning,
                                      "Parameter error",
                                      "Row and column numbers may not be zero!",
                                      QtWidgets.QMessageBox.Ok,
                                      self)
        popup.show()

    else:

        padding = self.pad.value()

        space = self.space.value()

        # Set the padding and spacing
```

11. INSERTING IMAGES:

WE can insert images or multiple images inside our document by clicking on the Insert Image icon from the toolbar or from the menu bar. We can also use **Ctrl-I** shortcut key to insert an image. A file open dialog will appear and we can explore the images to insert. An image can be of types (png, jpeg, gif, bump, etc). The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Insert Image”.**



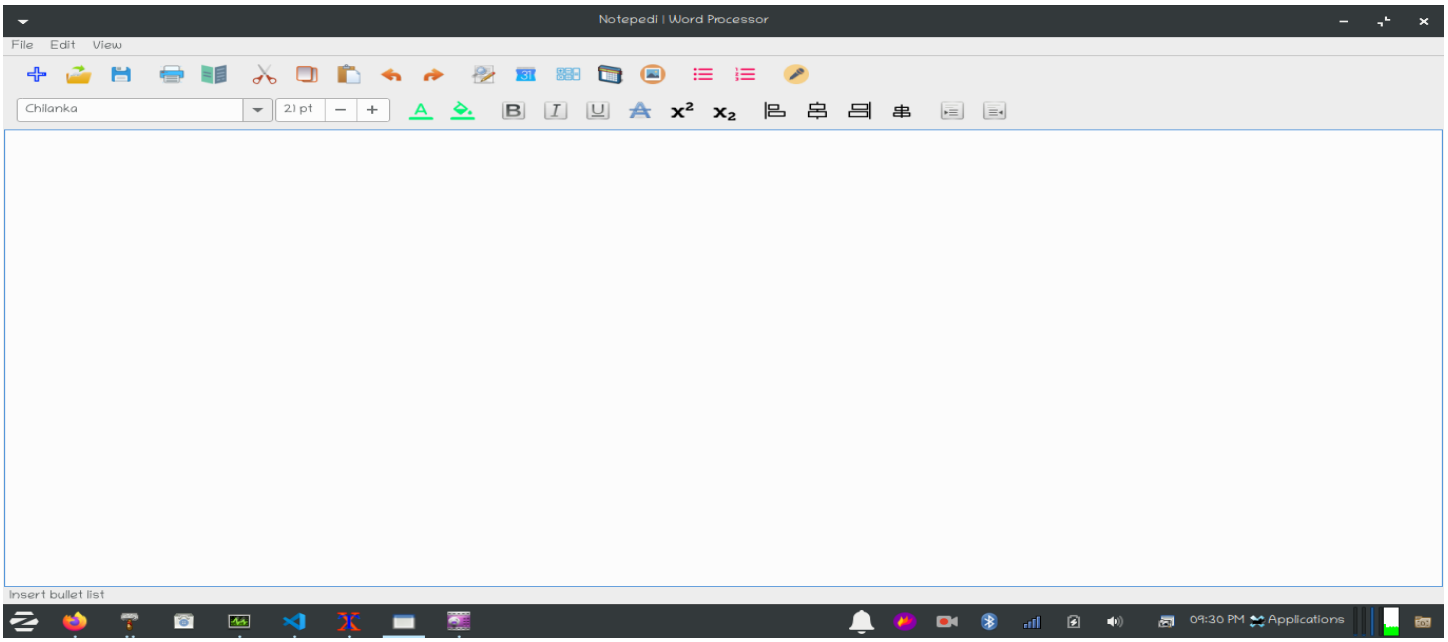

Insert Image
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Notepedia
500 GB V...
Other Locations
makdawoodi
Desktop
Notepedia
Name
Size
Modified
align_center.png 2.2 kB Fri
align_justify.png 3.5 kB Fri
align_left.png 3.8 kB Fri
align_right.png 2.2 kB Fri
ext Sat
test.jpeg 512.9 kB Sat
test.png 32.1 kB Tue
wow.gif 513.4 kB Sat
Images
Cancel
Open
09:29 PM
Applications

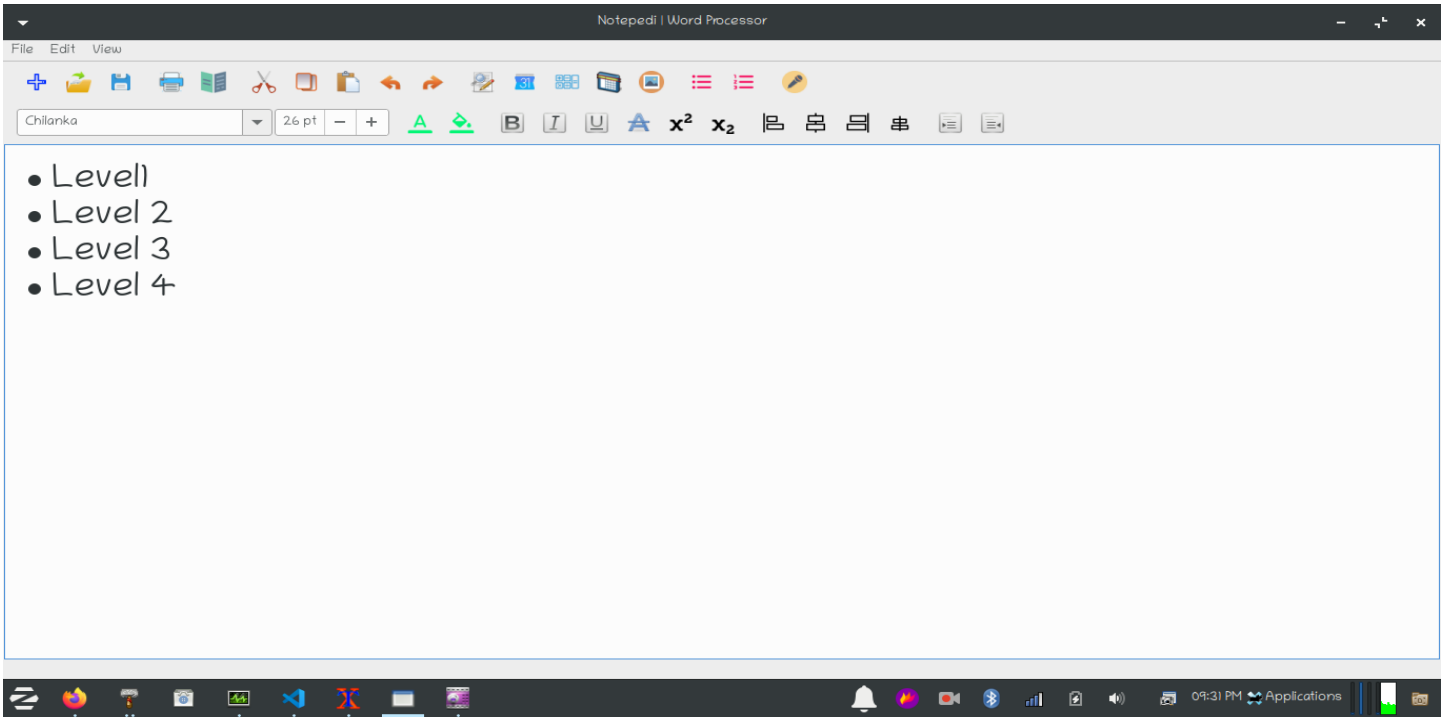

Notepedi | Word Processor
File Edit View
Chilanka
21 pt
Image
1234567890
Results
1234567890
Image
1234567890
Results
1234567890
Image
1234567890
Results
1234567890
Line: 2 | Column: 1
09:30 PM
Applications

```
def insertImage(self):

    # Get image file name
    #PYQT5 Returns a tuple in PyQt5
    filename = QtWidgets.QFileDialog.getOpenFileName(self, 'Insert image',".","Images (*.png *.xpm *.jpg *.jpeg *.bmp *.gif)")[0]

    if filename:

        # Create image object
        image = QtGui.QImage(filename)
        # Error if unloadable
        if image.isNull():

            popup = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Critical,
                                          "Image load error",
                                          "Could not load image file!",
                                          QtWidgets.QMessageBox.Ok,
                                          self)
            popup.show()

        else:

            cursor = self.text.textCursor()

            cursor.insertImage(image, filename)
            cursor.image.scaledToHeight(100)
            cursor.image.scaledToWidth(100)
```

12. INSERTING BULLETED LISTS:

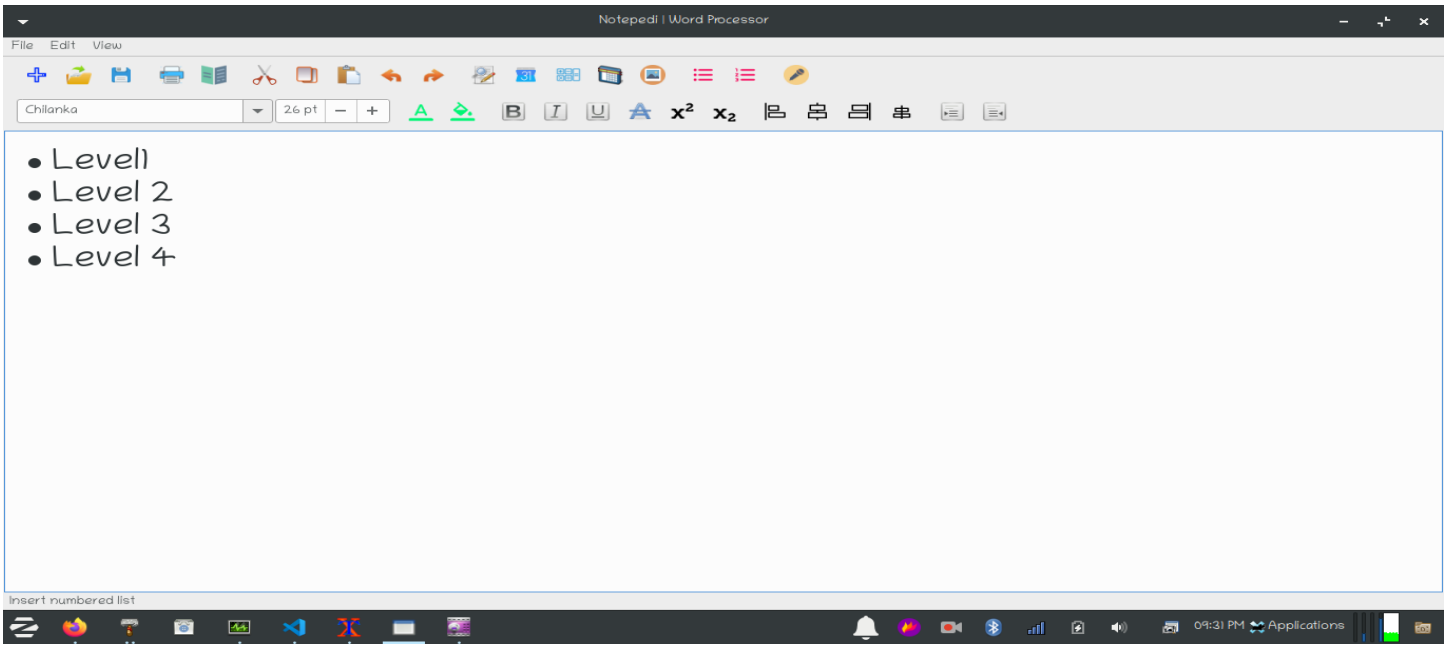
WE can insert bullet points list in our document by clicking on the Bulleted list icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-B** shortcut key to insert a bullet points list. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Insert Bulleted list”.**

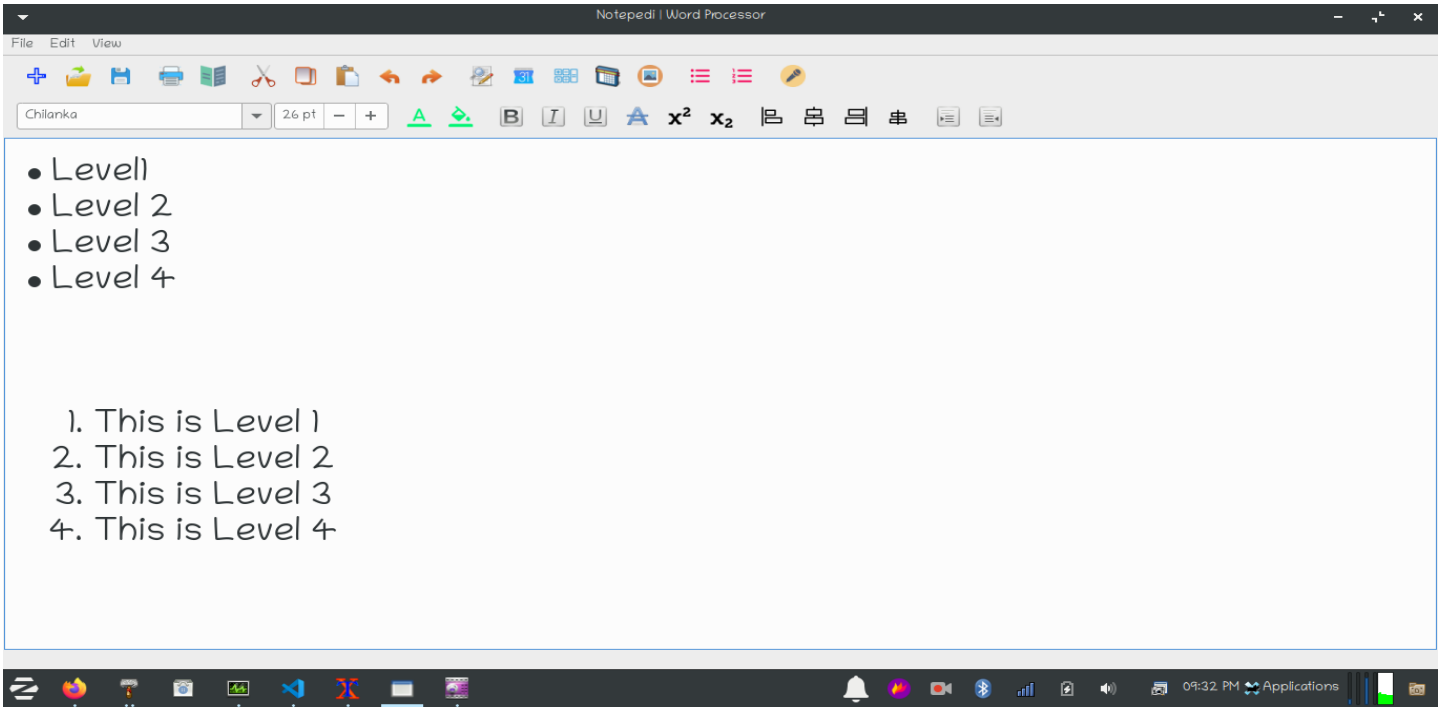




13. INSERTING NUMBERED LISTS:

WE can insert numbered list in our document by clicking on the Numbered list icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-N** shortcut key to insert a numbered list. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Insert Numbered list”.**

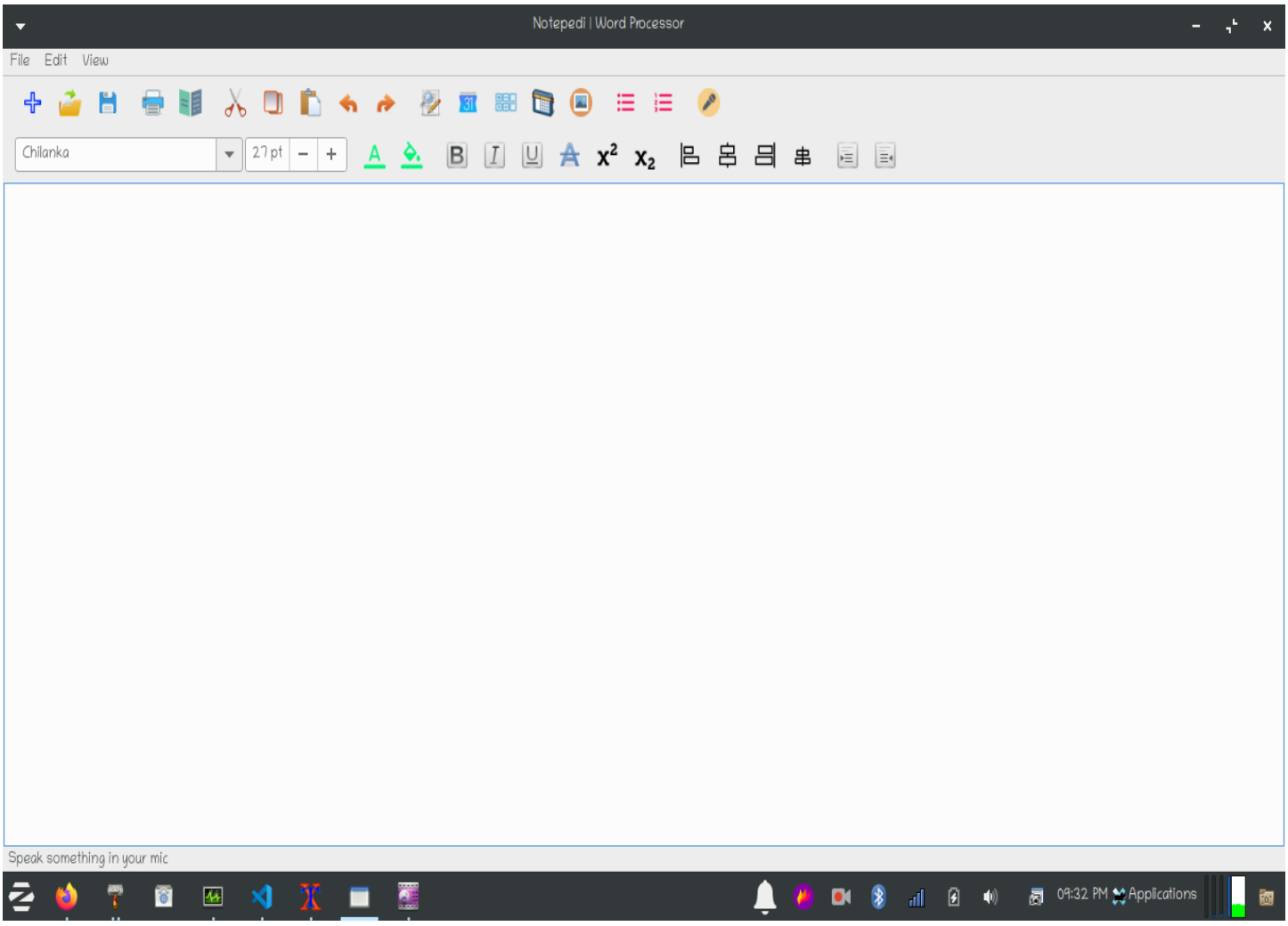


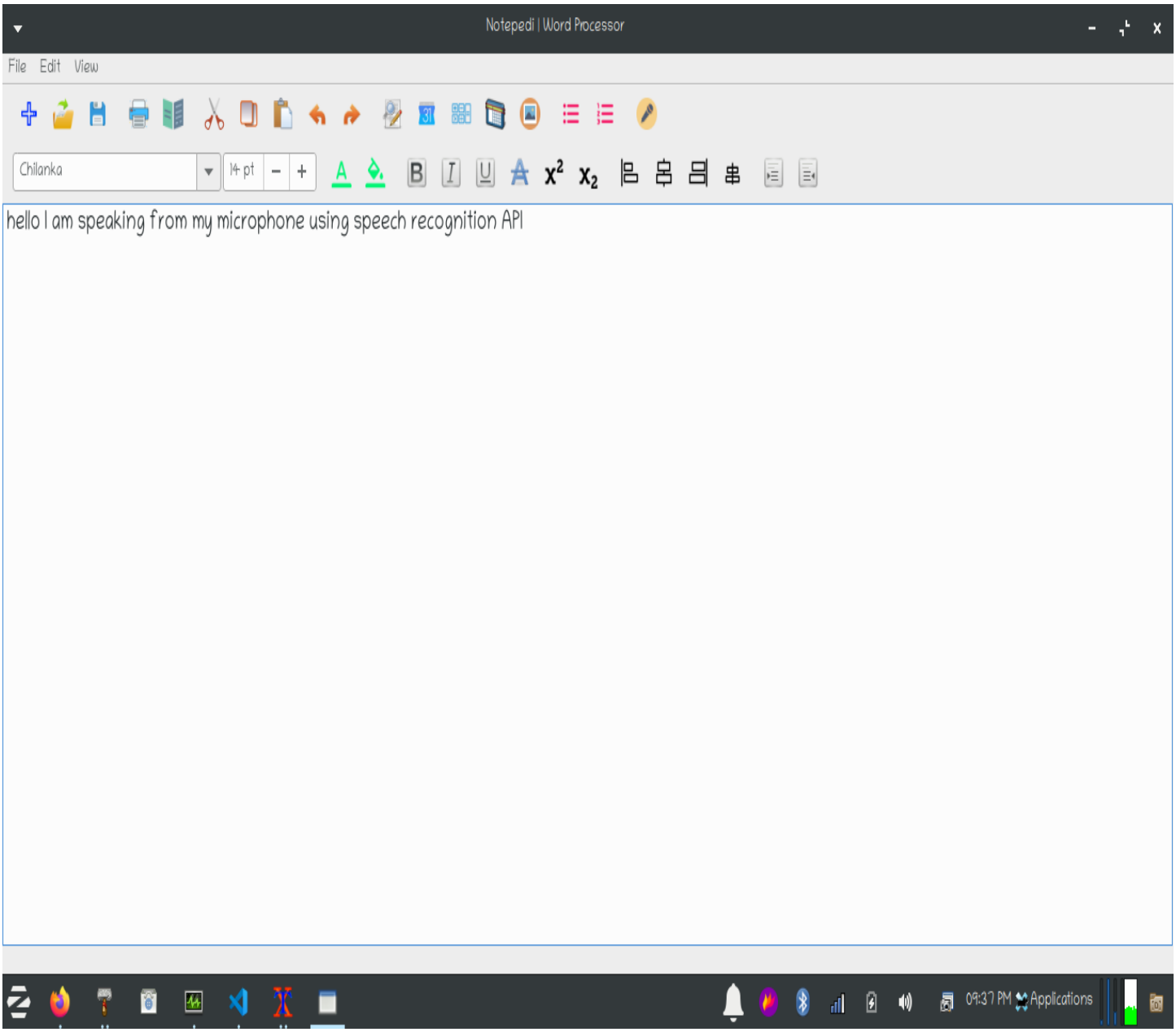



```python
def bulletList(self):

    cursor = self.text.textCursor()

    # Insert bulleted list
    cursor.insertList(QtGui.QTextListFormat.ListDisc)

def numberList(self):

    cursor = self.text.textCursor()

    # Insert list with numbers
    cursor.insertList(QtGui.QTextListFormat.ListDecimal)
```

14. SPEECH NOTES

This is a highlighting feature of this software (Notedia | Word Processor) because it uses Artificial Intelligence. We use google api cloud project and PyAudio to access the microphone connected to our system and use speech_recognition module to convert the audio spoken from the microphone and use google api to recognize the audio being spoken and convert it into text by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-M** shortcut key to insert speech notes. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Speak something in your microphone”.**

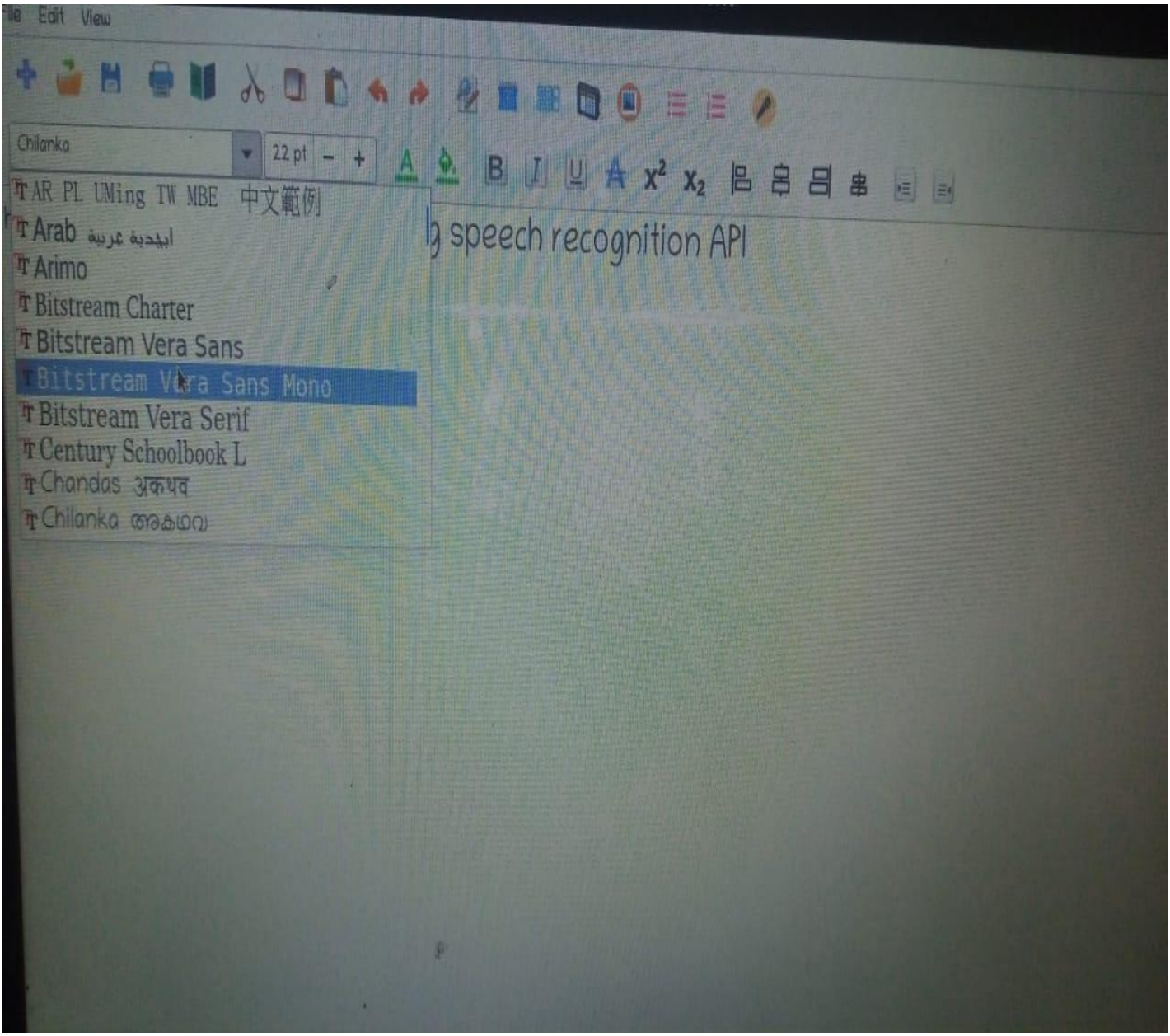





```
def speechRecog(self):
    r = sr.Recognizer()
    cursor = self.text.textCursor()

    with sr.Microphone() as source:
        audio = r.listen(source)
    try:
        text = r.recognize_google(audio)
        cursor.insertText(text)
    except:
        print('Sorry could not recognize your voice!')
```

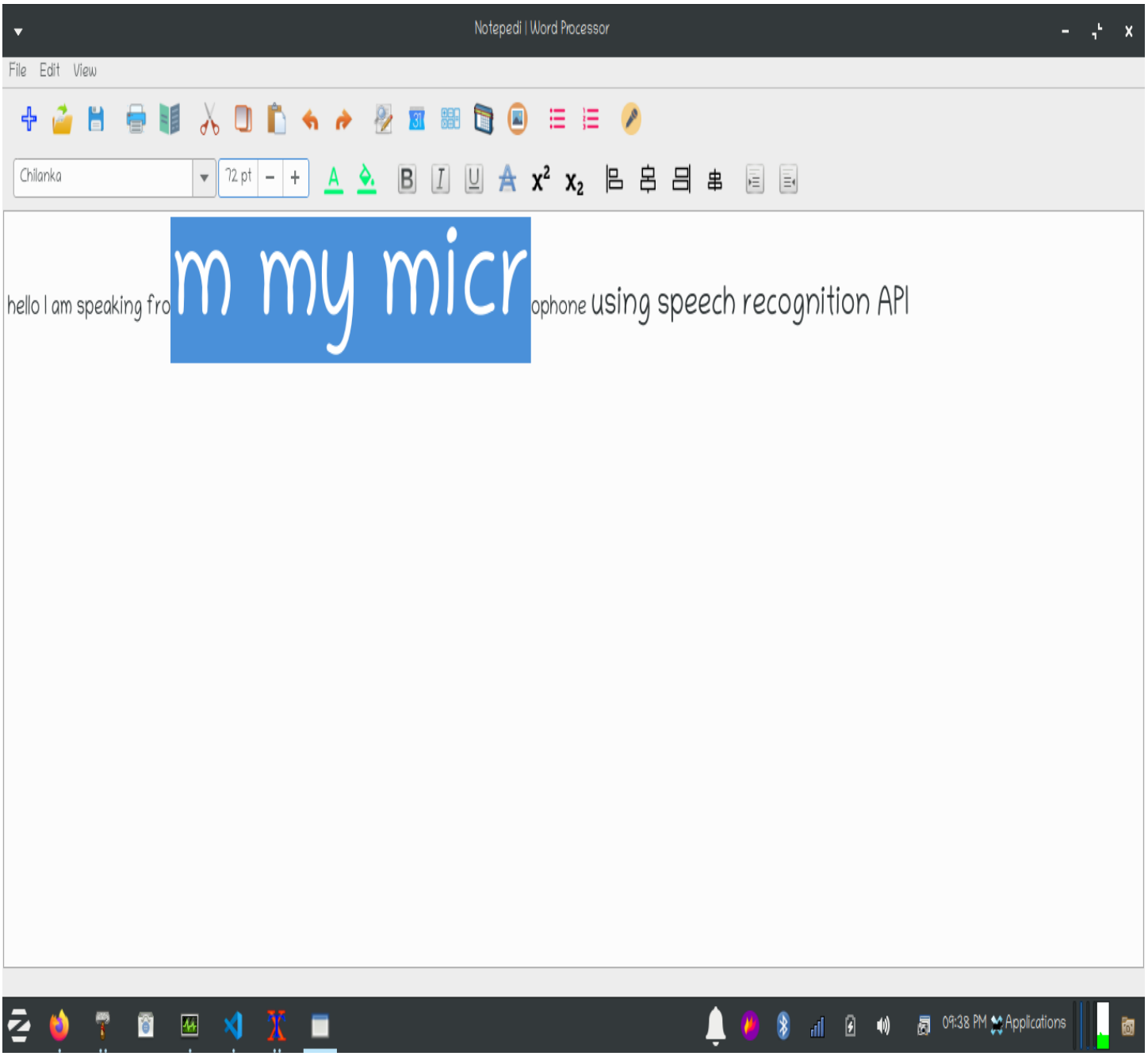
15. CHOOSING FONT FAMILY:

We can choose number of built-in fonts installed. The fonts will be individual for every text selected unlike notepad which has one single font for a complete document. Font Combo Box is used to select a font from.



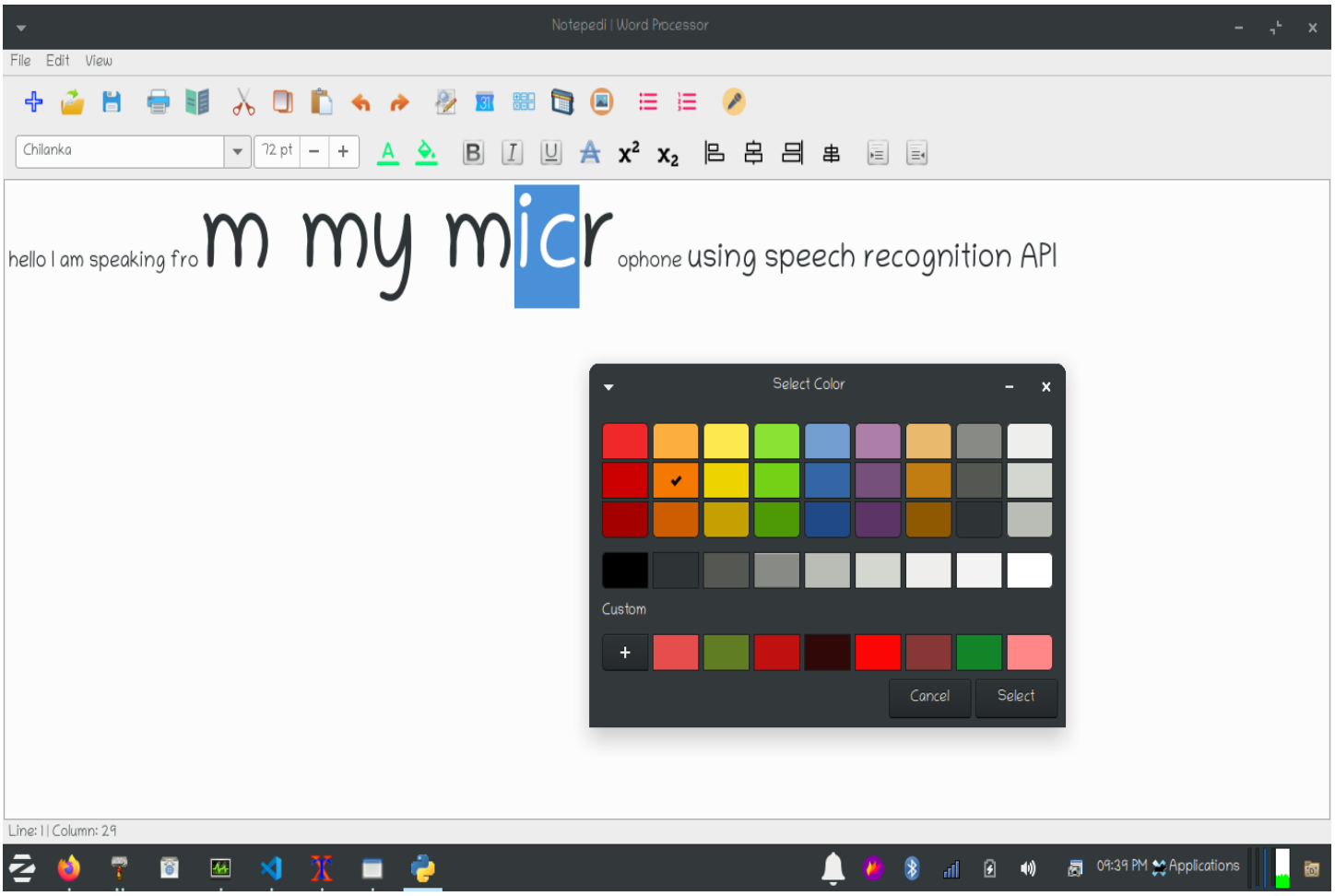
16. CHOOSING FONT SIZES:

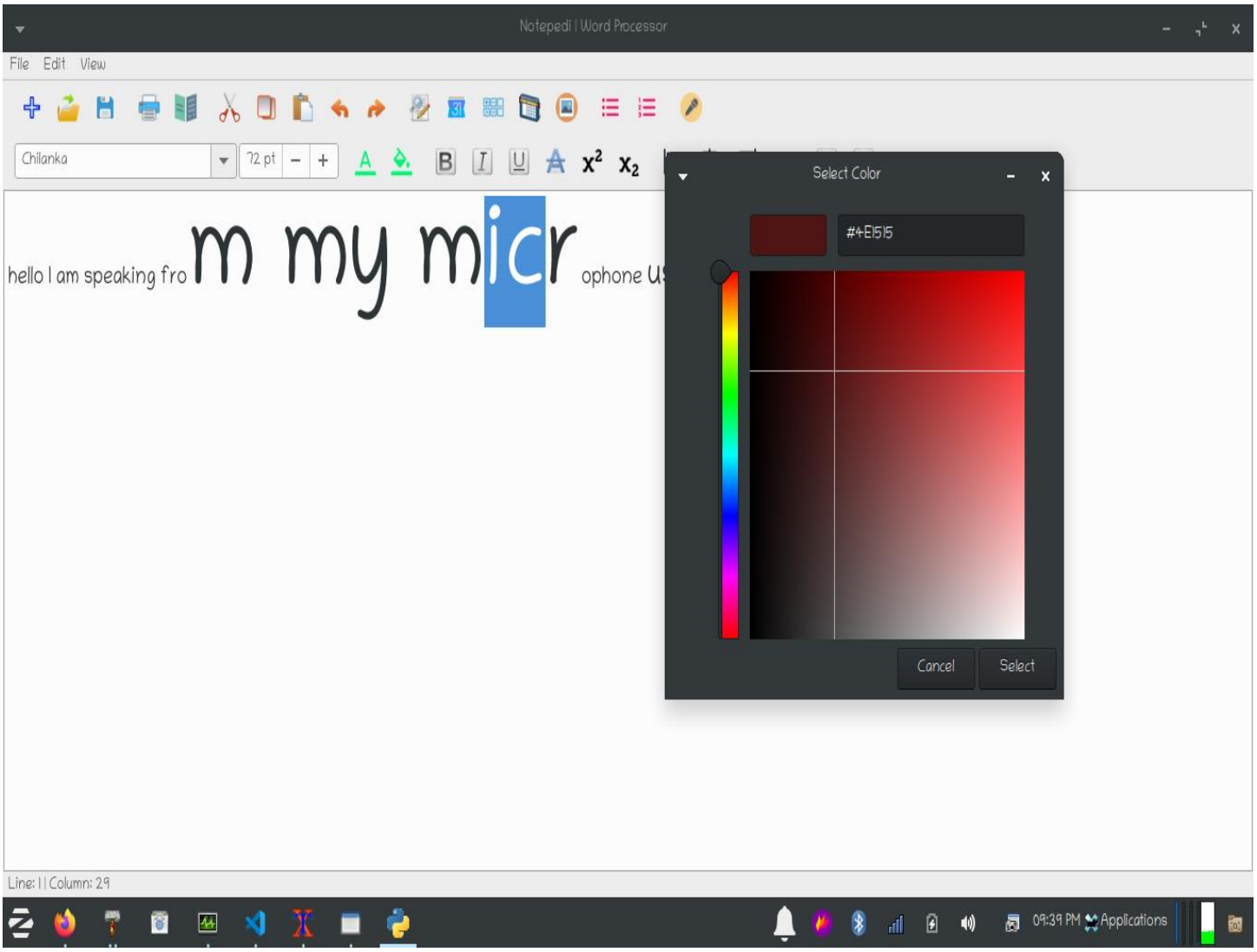
We can choose font sizes to make our text appear bigger or smaller. The font size will be individual for every text selected unlike notepad which has one single font size for a complete document. Font Spin Size is used to increase or decrease the font size.



17. CHANGE TEXT FOREGROUND COLOR:

We can change the Text Foreground Color in our document by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-F** shortcut key to open a color picker dialog. The Color will be individual for every text selected. When Font Foreground color function is triggered, the dialog for choosing a color or a color picker dialog will appear and user can select either from the built-in colors or user can make custom colors as well and the swatches will be saved as well. The customized color picked will also show the hex code as well. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Change text Foreground Color".**

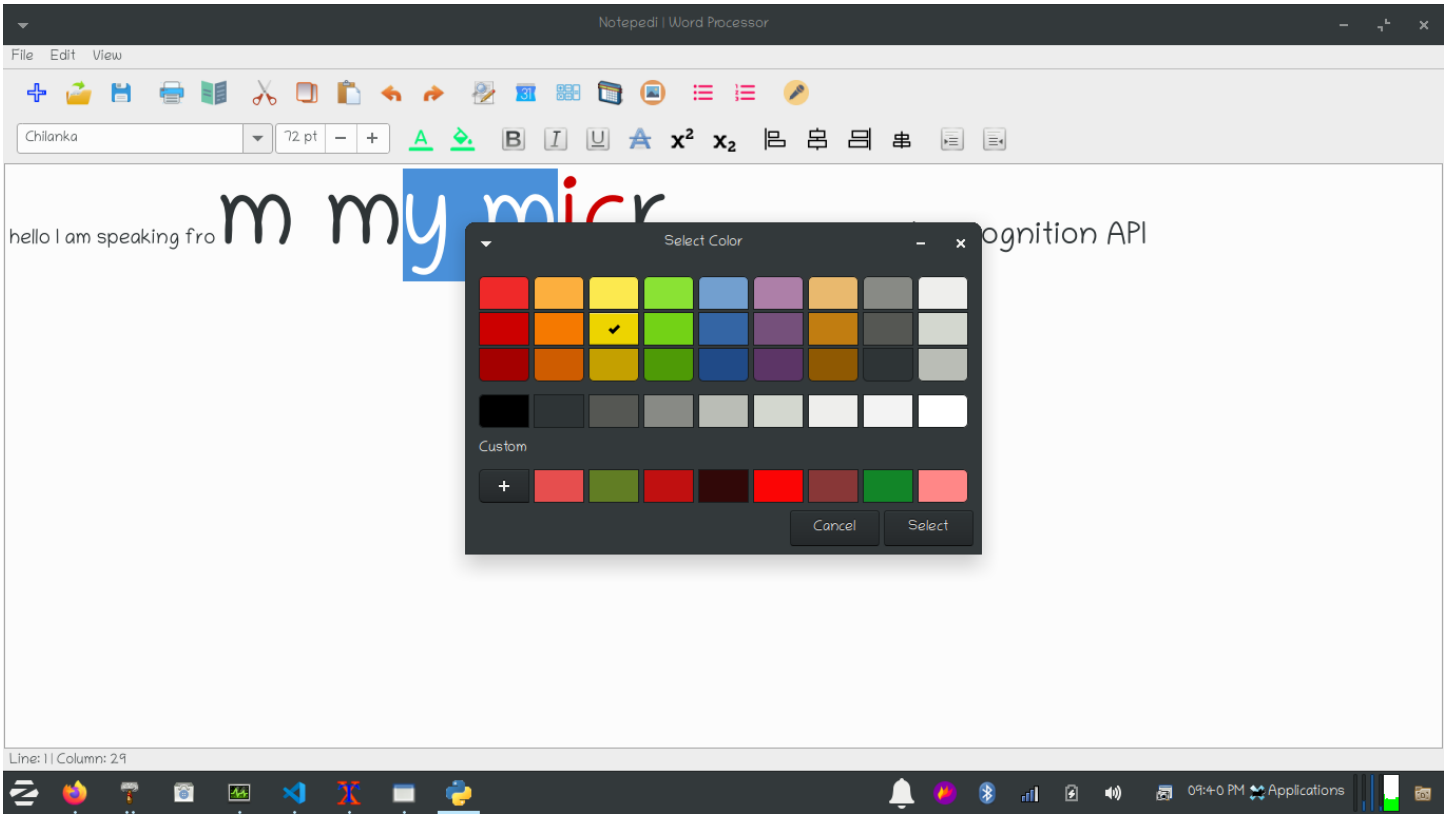


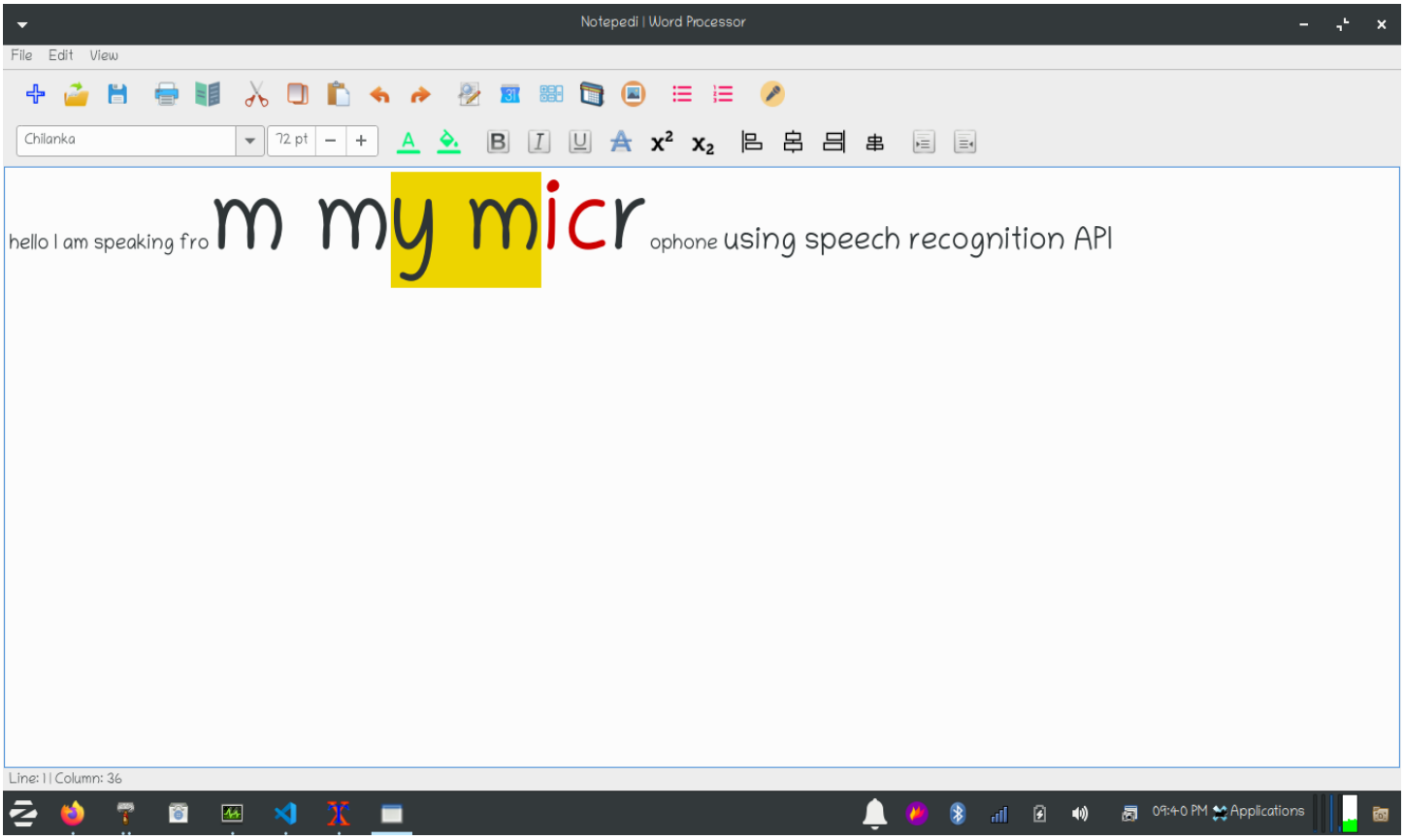


18. CHANGE TEXT BACKGROUND COLOR:

We can change the Text Background Color in our document by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-Alt-B** shortcut key to open a color picker dialog. The Color will be individual for every text selected. When Font Foreground color function is triggered, the dialog for choosing a color or a color picker dialog will appear and user can select either from the built-in colors or user can make custom colors as well and the swatches will be saved as well. The customized color picked will also show the hex code as well. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When

hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Change text Background Color”.**





19. MAKE A TEXT APPEAR BOLD:

We can make a text appear bold by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-B** shortcut key to make a text appear bold. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text appear bold".**

20. MAKE A TEXT APPEAR ITALICIZED:

We can make a text appear italicized by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-I** shortcut key to make a text appear italicized. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text appear italicized".**

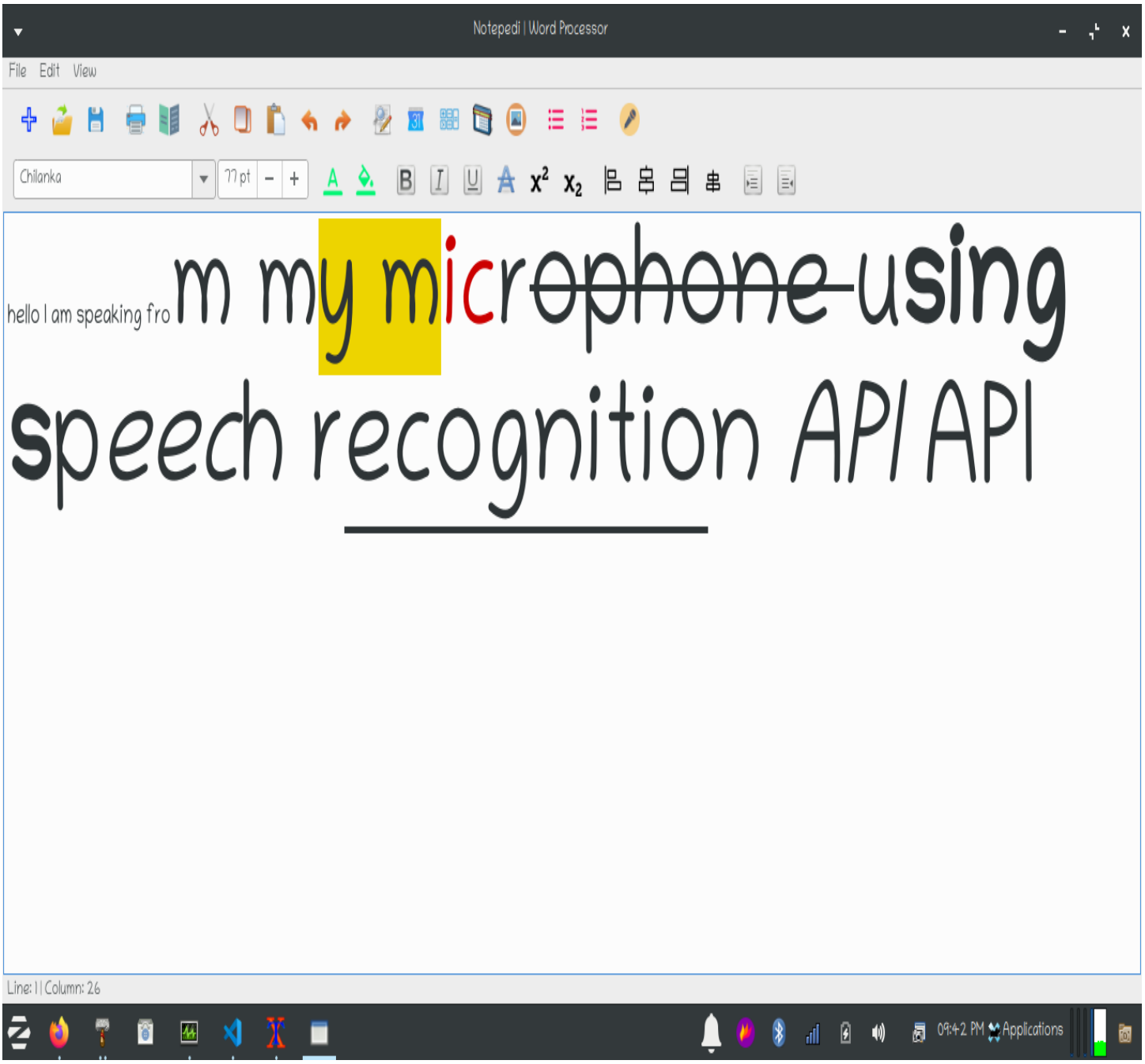
21. MAKE A TEXT APPEAR UNDERLINED:

We can make a text appear underlined by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-U** shortcut key to make a text appear underlined. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text appear underlined".**

22. MAKE A TEXT APPEAR STRIKED-OUT:

We can make a text appear striked-out by clicking on the icon from the toolbar or from the menu bar. The related Icon is also used to indicate a user of what

action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text appear striked-out”.**



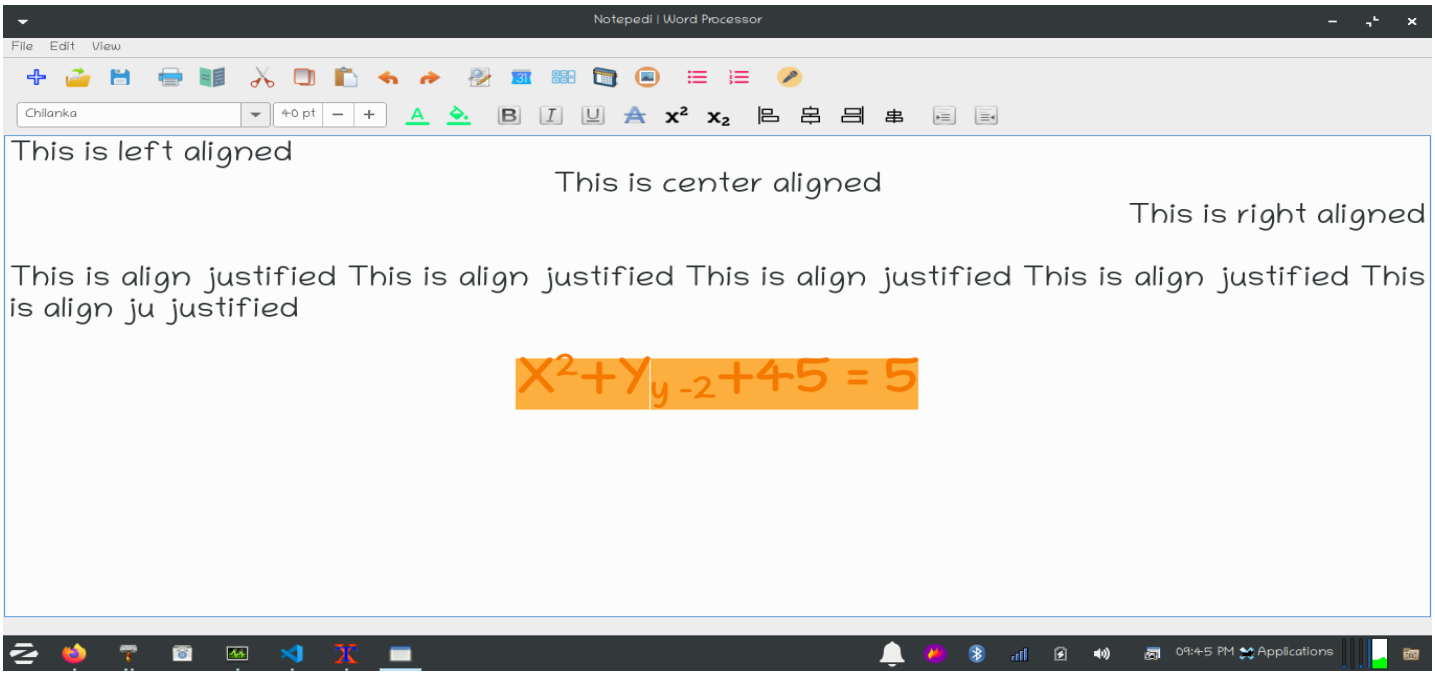
23. MAKE A TEXT SUPERSCRIPTED:

We can make a text appear superscripted by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-PgUp** shortcut key to make a text appear superscripted. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text appear superscripted".**

```
def superScript(self):

    # Grab the current format
    fmt = self.text.currentCharFormat()

    # And get the vertical alignment property
    align = fmt.verticalAlignment()

    # Toggle the state
    if align == QtGui.QTextCharFormat.AlignNormal:

        fmt.setVerticalAlignment(QtGui.QTextCharFormat.AlignSuperScript)

    else:

        fmt.setVerticalAlignment(QtGui.QTextCharFormat.AlignNormal)

    # Set the new format
    self.text.setCurrentCharFormat(fmt)
```

24. MAKE A TEXT SUBSCRIPTED:

We can make a text appear subscripted by clicking on the icon from the toolbar or from the menu bar. We can also use **Ctrl-PgDn** shortcut key to make a text appear subscripted. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text appear subscripted".**

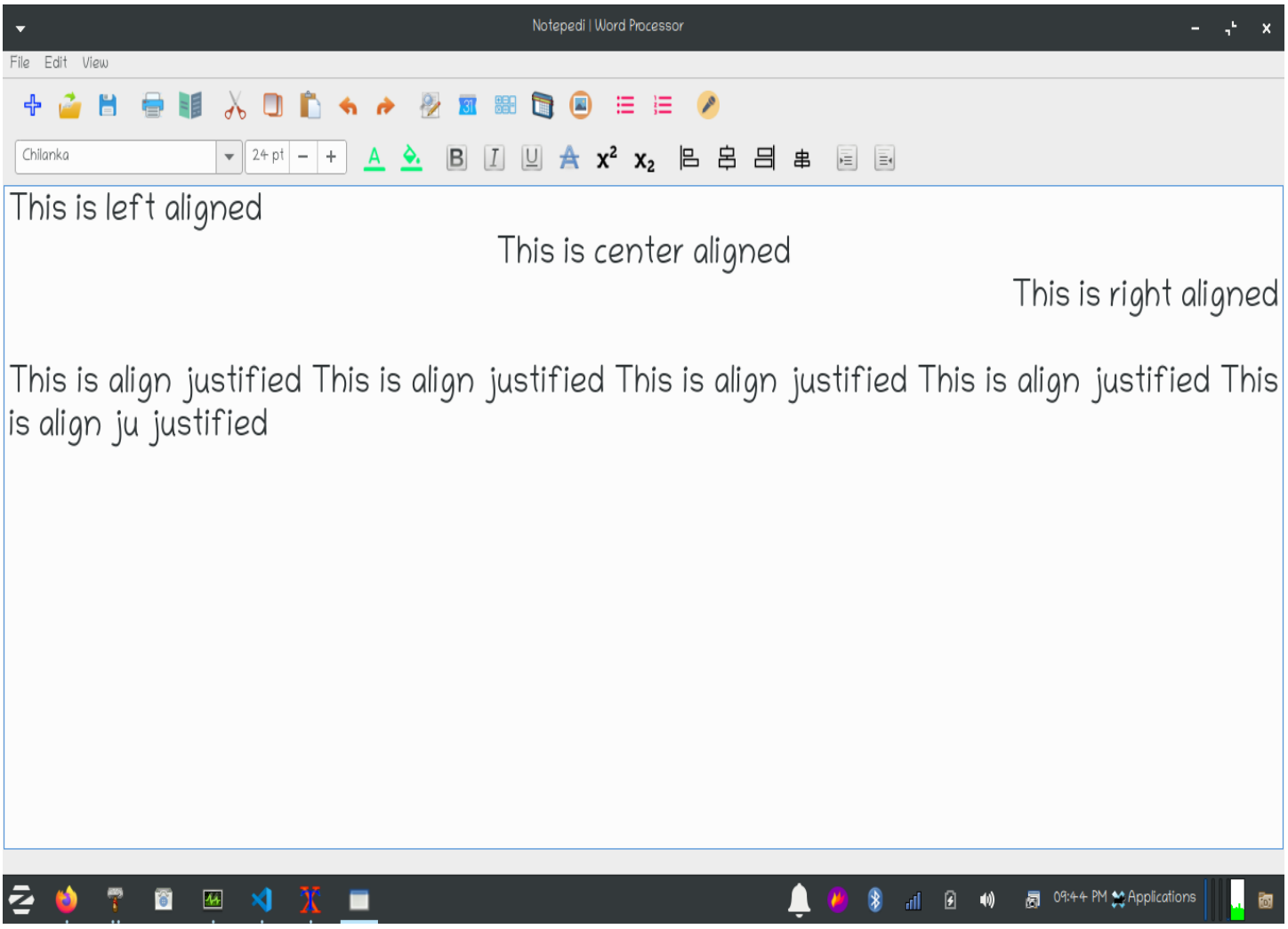



```
def subScript(self):

    # Grab the current format
    fmt = self.text.currentCharFormat()

    # And get the vertical alignment property
    align = fmt.verticalAlignment()

    # Toggle the state
    if align == QtGui.QTextCharFormat.AlignNormal:

        fmt.setVerticalAlignment(QtGui.QTextCharFormat.AlignSubScript)

    else:

        fmt.setVerticalAlignment(QtGui.QTextCharFormat.AlignNormal)
```

25. ALIGN TEXT:

We can make align a text to the left, to the right, to the center or align justify by clicking on the icons from the toolbar or from the menu bar. We can also use the **Ctrl-L** shortcut key for left aligning the text, **Ctrl-R** shortcut key for right aligning the text, **Ctrl-E** shortcut key for center aligning the text, **Ctrl-J shortcut key** for

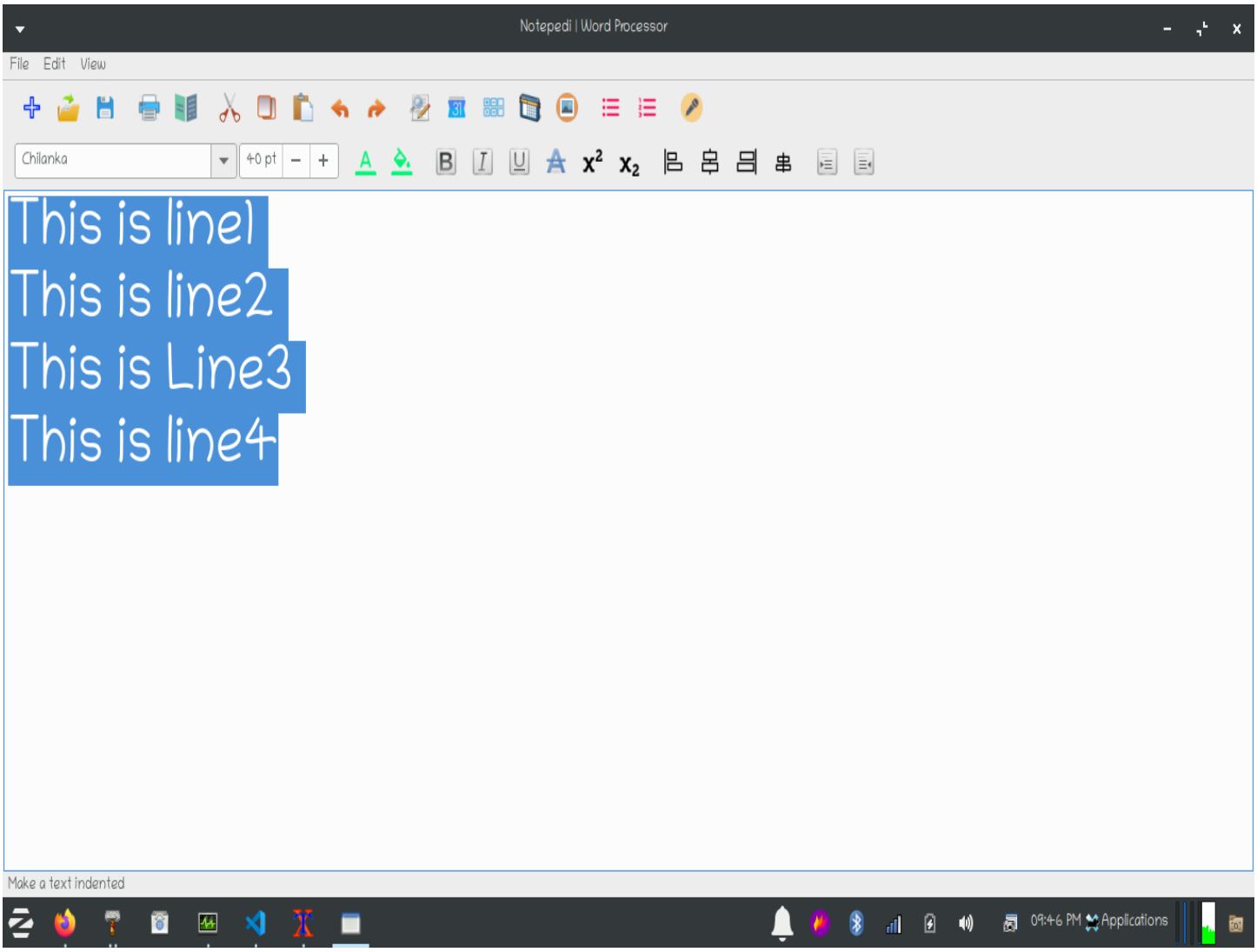
justify the text aligning. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a texts are displayed inside a status bar like **“Align a text to the left”** for left aligning **“Align a text to the right”** for right aligning, **“Align a text to the center”** for center aligning and **“Align a text justified”** for justify aligning a text**.**

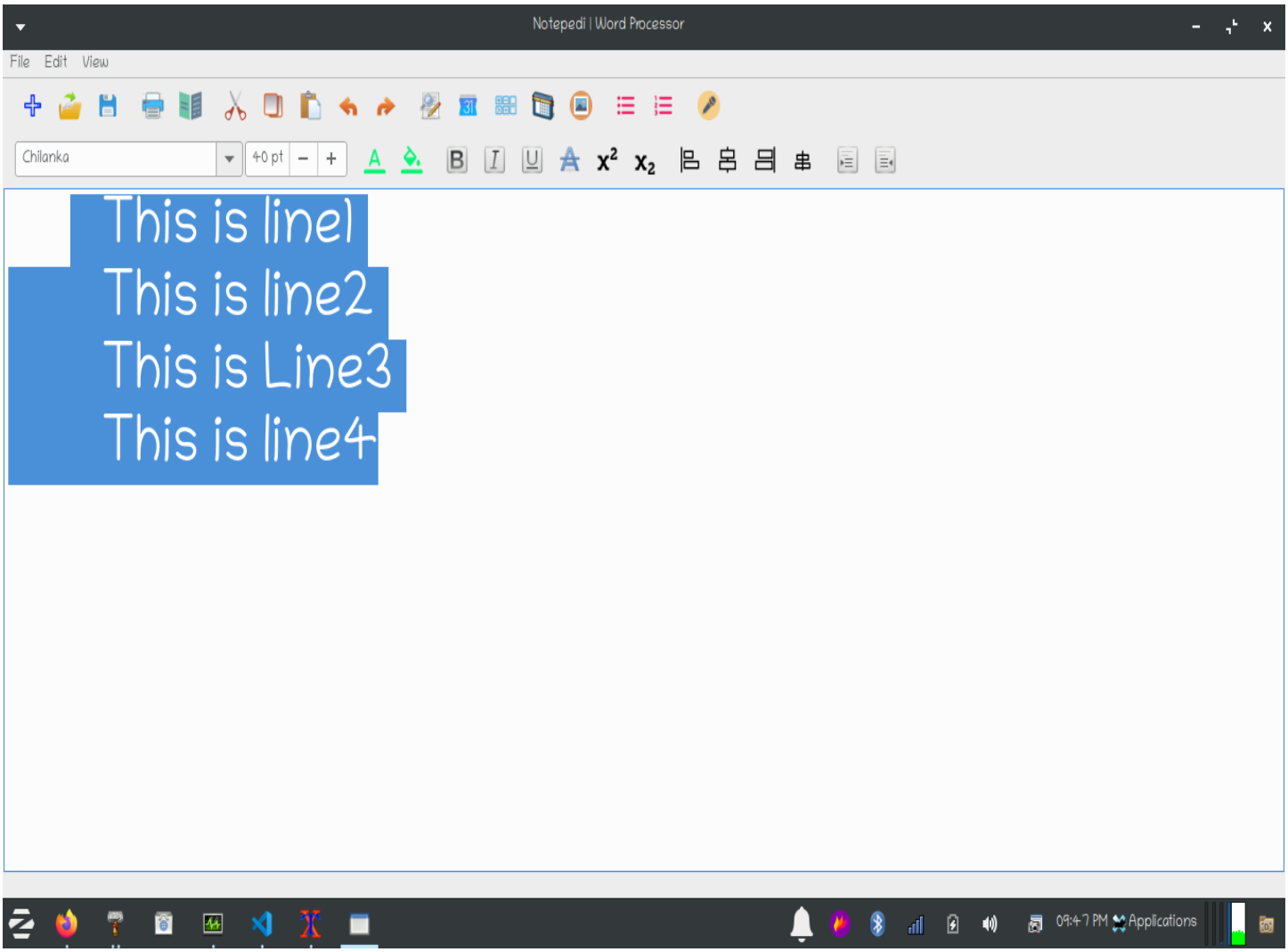


26. INDENT SELECTED LINES OF TEXTS:

We can make our selected text indented in a document by clicking on the indent icon available in the format bar for in the menu bar or we can also use the shortcut **Ctrl-Tab** to make a selected text indented. If we have the selection enabled then it will count how many lines we have selected and then according to

these lines it will place a tab on start of each line. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **“Make a text indented”.**

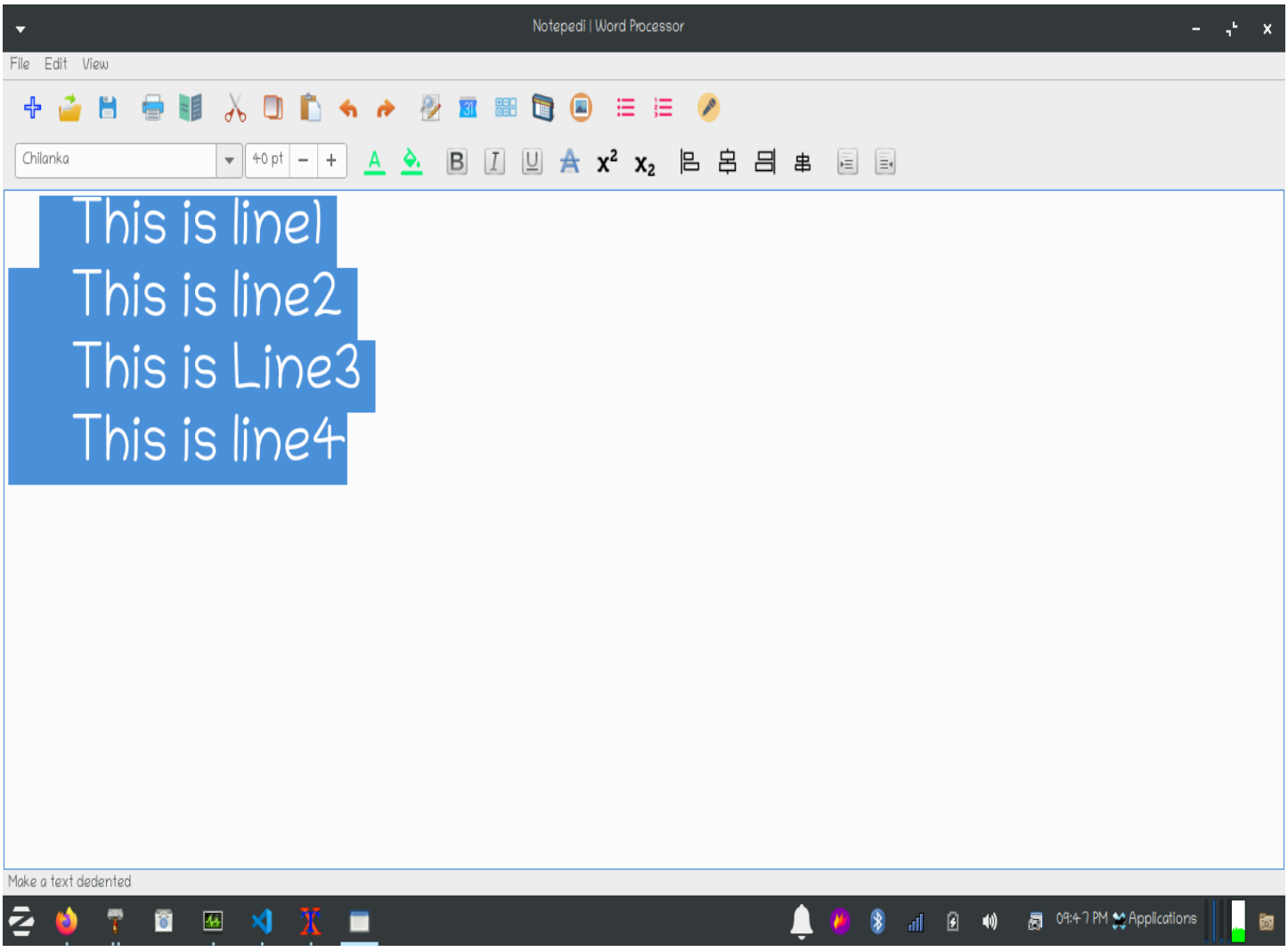





```python
def indent(self):

    # Grab the cursor
    cursor = self.text.textCursor()

    if cursor.hasSelection():

        # Store the current line/block number
        temp = cursor.blockNumber()

        # Move to the selection's end
        cursor.setPosition(cursor.anchor())

        # Calculate range of selection
        diff = cursor.blockNumber() - temp

        direction = QtGui.QTextCursor.Up if diff > 0 else QtGui.QTextCursor.Down

        # Iterate over lines (diff absolute value)
        for n in range(abs(diff) + 1):

            # Move to start of each line
            cursor.movePosition(QtGui.QTextCursor.StartOfLine)

            # Insert tabbing
            cursor.insertText("\t")

            # And move back up
            cursor.movePosition(direction)
```

27. DEDENT SELECTED LINES OF TEXT:

We can make our selected text dedented in a document by clicking on the dedent icon available in the format bar for in the menu bar or we can also use the shortcut **Ctrl-Shift-Tab** to make a selected text dedented. If we have the selection enabled then it will count how many lines we have selected and then according to these lines it will remove a tab on start of each line. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut. When hovered over the mouse we get a glimpse of what the icon is about and a text is displayed inside a status bar **"Make a text dedented".**

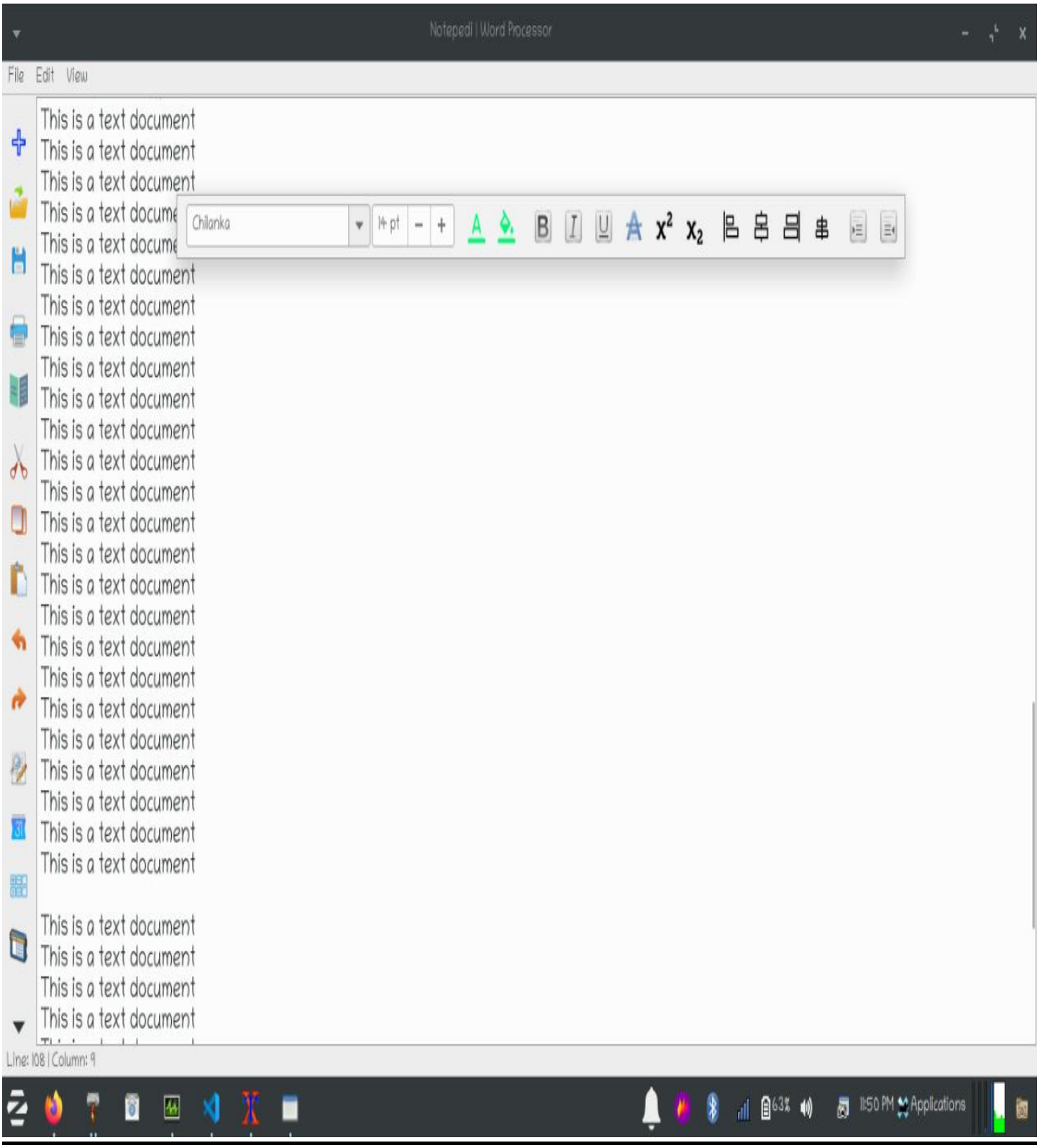


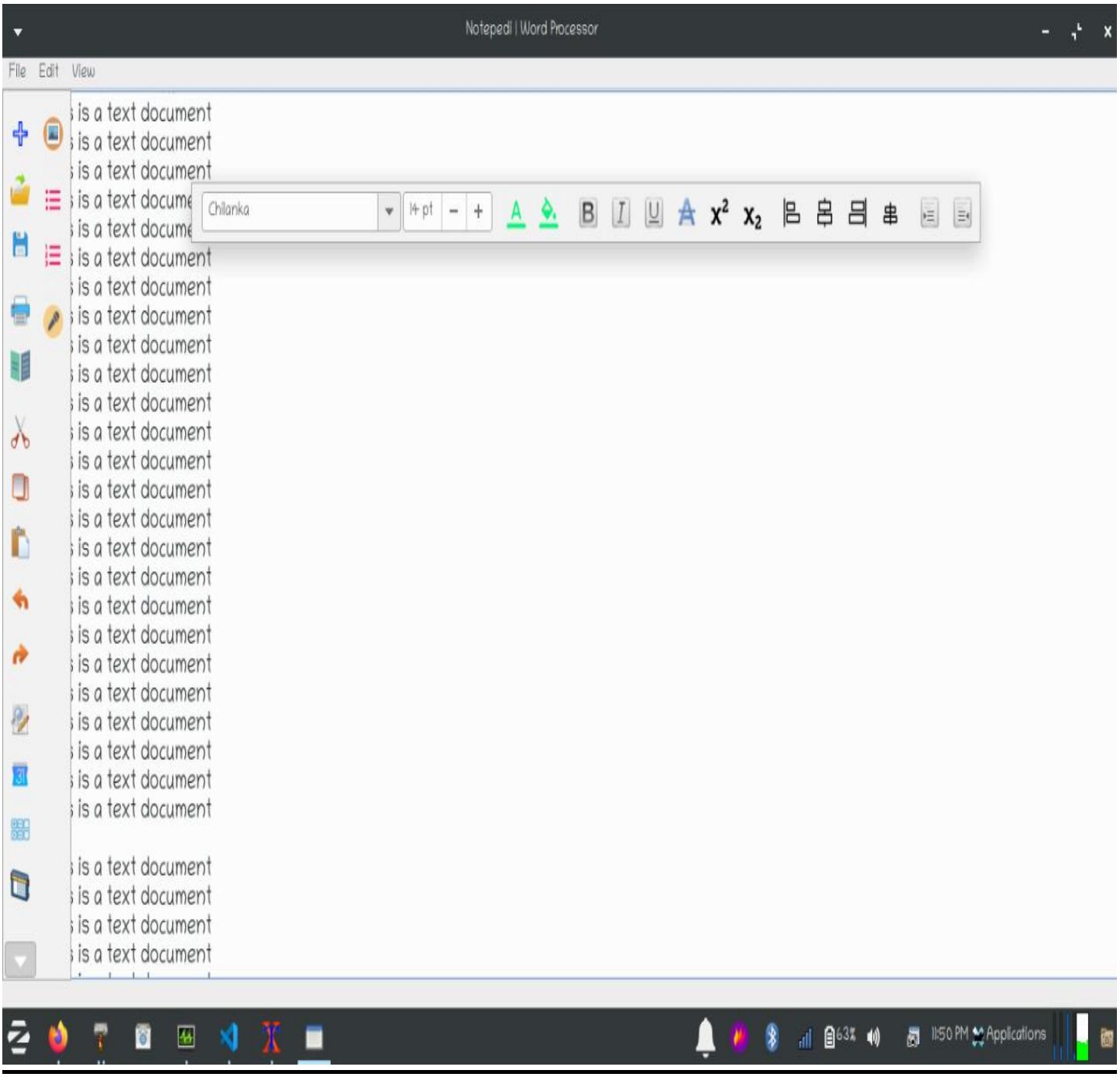
```
def handleDedent(self,cursor):

    cursor.movePosition(QtGui.QTextCursor.StartOfLine)

    # Grab the current line
    line = cursor.block().text()

    # If the line starts with a tab character, delete it
    if line.startswith("\t"):

        # Delete next character
        cursor.deleteChar()

    # Otherwise, delete all spaces until a non-space character is met
    else:
        for char in line[:8]:

            if char != " ":
                break

            cursor.deleteChar()

def dedent(self):

    cursor = self.text.textCursor()

    if cursor.hasSelection():
```

28. DRAG AND DROP UI ELEMENTS

A Special functionality that is not available in most of the word processing or text processing software is the drag and drop UI elements we can make use of these drag and drop UI elements to drop the format bar to whatever position we want it to be placed and we can even place the the format bar, menu bar or toolbar outside of our software anywhere we want and that is the power of drag and drop UI elements and if the icons are not showing properly due to space limitations then it will show an arrow that we can click and show rest of the icons. A highlighter will appear whenever we want to drop the UI element inside our software these kind of functionalities are normally available in professional-grade softwares like Photoshop or Adobe Illustrator where we are given full use

case on how well we can customise the software and now this functionality is available in Notepedia word processor. The related Icon is also used to indicate a user of what action will be performed when icon is triggered by clicking or by triggering the shortcut.




Notepadl | Word Processor
File Edit View
is a text document
Chilanka
14 pt

PROJECT SCOPE

The Scope of this project is very large. In future, this software can be implemented on a large scale and can include features such as making audio notes from the text files, OCR scanning, Built-in Dictionary, Text to speech conversion, Full Image editing, Translator, Voice Assistance Internet searching, Voice Assistant, Using as a code editor like Visual Studio Code, etc.