***Project Name:***

***notepad***

**Name: Hafiza Ayesha Sadiq**

**Serial No. : 34**

**Student ID : 84822**

**Class: CIT-PP 2.0 ANJUMAN( 2 to 4 Sat/Sun)**

***Name of Each Member:***

**Hafiza Ayesha Sadiq**

**Sohima Hamid**

**Maheera Saleem**

***Name of Leader:***

**Hafiza Ayesha Sadiq**

***Details about Code:***

* Creating Notepad in Python can be a great exercise for learning the programming language and to practice coding skills and learn more about GUI development.
* In GUI and other applications we import different libraries such as Tkinter to make use of the functionality and pre-built tools that the library provides. By importing these libraries, we can easily create windows, buttons, labels, and other GUI elements without having to write all the code from scratch.
* Additionally, importing libraries allows us to use the work of other developers who have created these tools it saves our time and effort.
* By creating our own notepad, we have the opportunity to enhance our skills in Python programming by implementing features like saving, opening, and editing text files. We can also add features that are not available in standard notepad applications.
* Developing a notepad in Python can help us to understand how text editors work and improve our understanding of programming concepts.

***Real Life Application:***

* Notepad is a very basic text writing and editing program that is used to create computer notes that are typed in English.
* It is used for writing and testing small pieces of codes.
* It loads quickly and is easy to use, making it a good choice for quick edits or when you need to work with plain text without any formatting.
* Before sending formal emails, users can compose and edit drafts in Notepad.
* The use of a notepad in real life is to take quick notes, make to-do lists, and store information for later reference.
* Notepad helps in storing temporary information like phone numbers, addresses, or recipe’s for quick access.
* Notepad is a simple versatile tool that is commonly used in various industries such as education, business, and technology.

***Code for Notepad:***

import tkinter

import os

from tkinter import \*

from tkinter.messagebox import \*

from tkinter.filedialog import \*

from tkinter.simpledialog import \*

from tkinter.font import \*

from tkinter.colorchooser import \*

class Notepad1:

\_\_root1 = Tk()

# default Notepad window width and height

\_\_thisWidth = 300

\_\_thisHeight = 300

\_\_thisTextArea = Text(\_\_root1)

\_\_thisMenuBar = Menu(\_\_root1)

\_\_thisFileMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisEditMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisAlignmentMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisFormatMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisHelpMenu = Menu(\_\_thisMenuBar, tearoff=0)

# To add scrollbar in Notepad

\_\_thisScrollBar = Scrollbar(\_\_thisTextArea)

\_\_file = None

def \_\_init\_\_(self, \*\*note):

# Set icon

try:

self.\_\_root1.wm\_iconbitmap("Notepad.ico")

except:

pass

# Set Notepad window size the default is 300x300

try:

self.\_\_thisWidth = note['width']

except KeyError:

pass

try:

self.\_\_thisHeight = note['height']

except KeyError:

pass

# Set the Notepad window text

self.\_\_root1.title("Untitled - Notepad")

# Center of the window

screenWidth = self.\_\_root1.winfo\_screenwidth()

screenHeight = self.\_\_root1.winfo\_screenheight()

# For left alling

leftallign = (screenWidth / 2) - (self.\_\_thisWidth / 2)

# For right-allign

topallign = (screenHeight / 2) - (self.\_\_thisHeight / 2)

# For top and bottom

self.\_\_root1.geometry('%dx%d+%d+%d' % (self.\_\_thisWidth,

self.\_\_thisHeight,

leftallign, topallign))

# for making the textarea auto resizable

self.\_\_root1.grid\_rowconfigure(0, weight=1)

self.\_\_root1.grid\_columnconfigure(0, weight=1)

# Add the controls (widget)

self.\_\_thisTextArea.grid(sticky=N + E + S + W)

# for open new file in Notepad

self.\_\_thisFileMenu.add\_command(label="New",

command=self.\_\_newFile)

# for open a already existing file

self.\_\_thisFileMenu.add\_command(label="Open",

command=self.\_\_openFile)

# To save current file of Notepad

self.\_\_thisFileMenu.add\_command(label="Save",

command=self.\_\_saveFile)

# To change the background color

self.\_\_thisFileMenu.add\_command(label="Change Background Color",

command=self.\_\_changeBackgroundColor)

# To create a line in the dialog

self.\_\_thisFileMenu.add\_separator()

self.\_\_thisFileMenu.add\_command(label="Exit",

command=self.\_\_quitApplication)

self.\_\_thisMenuBar.add\_cascade(label="File",

menu=self.\_\_thisFileMenu)

# To give a feature of cut

self.\_\_thisEditMenu.add\_command(label="Cut",

command=self.\_\_cut)

# to give a feature of copy

self.\_\_thisEditMenu.add\_command(label="Copy",

command=self.\_\_copy)

# To give a feature of paste

self.\_\_thisEditMenu.add\_command(label="Paste",

command=self.\_\_paste)

# To give a feature of replace

self.\_\_thisEditMenu.add\_command(label="Replace",

command=self.\_\_replaceText)

# To give a feature of selectall

self.\_\_thisEditMenu.add\_command(label="Selectall",

command=self.\_\_selectall)

# To give a feature of bold

self.\_\_thisEditMenu.add\_command(label="Bold",

command=self.\_\_bold)

# To give a feature of italic

self.\_\_thisEditMenu.add\_command(label="Italic",

command=self.\_\_italic)

# To give a feature of underline

self.\_\_thisEditMenu.add\_command(label="underline",

command=self.\_\_underline)

# To give a feature of editing

self.\_\_thisMenuBar.add\_cascade(label="Edit",

menu=self.\_\_thisEditMenu)

# Add font size option to the Format menu

self.\_\_thisFormatMenu.add\_command(label="Font Size",

command=self.\_\_changeFontSize)

# Add font color option to the Format menu

self.\_\_thisFormatMenu.add\_command(label="Font Color",

command=self.\_\_changeFontColor)

#Add font style option to the Format menu

self.\_\_thisFormatMenu.add\_command(label="Font Style",

command=self.\_\_changeFontStyle)

# Add alignment options to the Alignment menu

self.\_\_thisAlignmentMenu.add\_command(label="Align Left",

command=self.\_\_alignLeft)

self.\_\_thisAlignmentMenu.add\_command(label="Align Center",

command=self.\_\_alignCenter)

self.\_\_thisAlignmentMenu.add\_command(label="Align Right",

command=self.\_\_alignRight)

# To give a feature of alignment

self.\_\_thisMenuBar.add\_cascade(label="Alignment",

menu=self.\_\_thisAlignmentMenu)

# Add the Format menu to the main menu bar

self.\_\_thisMenuBar.add\_cascade(label="Format", menu=self.\_\_thisFormatMenu)

# for creating the feature of description of the notepad

self.\_\_thisHelpMenu.add\_command(label="About Notepad",

command=self.\_\_showAbout)

self.\_\_thisMenuBar.add\_cascade(label="Help",

menu=self.\_\_thisHelpMenu)

self.\_\_root1.config(menu=self.\_\_thisMenuBar)

self.\_\_thisScrollBar.pack(side=RIGHT, fill=Y)

# Scrollbar will adjust automatically according to the content

self.\_\_thisScrollBar.config(command=self.\_\_thisTextArea.yview)

self.\_\_thisTextArea.config(yscrollcommand=self.\_\_thisScrollBar.set)

def \_\_quitApplication(self):

self.\_\_root1.destroy()

# exit()

def \_\_showAbout(self):

showinfo("Notepad", "This Notepad is created by using python")

def \_\_openFile(self):

self.\_\_file = askopenfilename(defaultextension=".txt",

filetypes=[("All Files", "\*.\*"),

("Text Documents", "\*.txt")])

if self.\_\_file == "":

# no file to open

self.\_\_file = None

else:

# Try to open the file

# set the window title

self.\_\_root1.title(os.path.basename(self.\_\_file) + " - Notepad")

self.\_\_thisTextArea.delete(1.0, END)

file = open(self.\_\_file, "r")

self.\_\_thisTextArea.insert(1.0, file.read())

file.close()

def \_\_newFile(self):

self.\_\_root1.title("Untitled - Notepad")

self.\_\_file = None

self.\_\_thisTextArea.delete(1.0, END)

def \_\_changeBackgroundColor(self):

color = askcolor()

if color[1]:

self.\_\_thisTextArea.config(bg=color[1])

def \_\_saveFile(self):

if self.\_\_file == None:

# Save as new file

self.\_\_file = asksaveasfilename(initialfile='Untitled.txt',

defaultextension=".txt",

filetypes=[("All Files", "\*.\*"),

("Text Documents", "\*.txt")])

if self.\_\_file == "":

self.\_\_file = None

else:

# Try to save the file

file = open(self.\_\_file, "w")

file.write(self.\_\_thisTextArea.get(1.0, END))

file.close()

# for Change the window title

self.\_\_root.title(os.path.basename(self.\_\_file) + " - Notepad")

else:

file = open(self.\_\_file, "w")

file.write(self.\_\_thisTextArea.get(1.0, END))

file.close()

def \_\_cut(self):

self.\_\_thisTextArea.event\_generate("<<Cut>>")

def \_\_copy(self):

self.\_\_thisTextArea.event\_generate("<<Copy>>")

def \_\_paste(self):

self.\_\_thisTextArea.event\_generate("<<Paste>>")

def \_\_replaceText(self):

replace\_text = askstring("Replace", "Enter text:")

if replace\_text:

replace\_with = askstring("Replace", f"Enter'{replace\_text}' with:")

if replace\_with:

content = self.\_\_thisTextArea.get(1.0, END)

new\_content = content.replace(replace\_text, replace\_with)

self.\_\_thisTextArea.delete(1.0, END)

self.\_\_thisTextArea.insert(1.0, new\_content)

def \_\_selectall(self):

self.\_\_thisTextArea.tag\_add("sel", '1.0', 'end')

def \_\_bold(self):

self.\_\_thisTextArea.tag\_add('bold','sel.first','sel.last')

self.\_\_thisTextArea.tag\_config('bold', font=('Calibri', 18, 'bold'))

def \_\_italic(self):

self.\_\_thisTextArea.tag\_add('italic','sel.first','sel.last')

self.\_\_thisTextArea.tag\_config('italic', font=('Arial', 12, 'italic'))

def \_\_underline(self):

self.\_\_thisTextArea.tag\_add('underline','sel.first','sel.last')

self.\_\_thisTextArea.tag\_config('underline', underline=True)

def \_\_alignLeft(self):

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

if sel\_start and sel\_end:

if sel\_end != self.\_\_thisTextArea.index("end"):

self.\_\_thisTextArea.tag\_configure("alignment\_left", justify=LEFT)

self.\_\_thisTextArea.tag\_add("alignment\_left", sel\_start, sel\_end)

def \_\_alignCenter(self):

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

if sel\_start and sel\_end:

if sel\_end != self.\_\_thisTextArea.index("end"):

self.\_\_thisTextArea.tag\_configure("alignment\_center", justify=CENTER)

self.\_\_thisTextArea.tag\_add("alignment\_center", sel\_start, sel\_end)

def \_\_alignRight(self):

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

if sel\_start and sel\_end:

if sel\_end != self.\_\_thisTextArea.index("end"):

self.\_\_thisTextArea.tag\_configure("alignment\_right", justify=RIGHT)

self.\_\_thisTextArea.tag\_add("alignment\_right", sel\_start, sel\_end)

def \_\_changeFontSize(self):

font\_size = askinteger("Font Size", "Enter Font Size:")

if font\_size:

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

self.\_\_thisTextArea.tag\_add("font\_size", sel\_start, sel\_end)

self.\_\_thisTextArea.tag\_config("font\_size", font=(None, font\_size))

def \_\_changeFontColor(self):

color = askcolor()

if color[1]:

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

self.\_\_thisTextArea.tag\_add("font\_color", sel\_start, sel\_end)

self.\_\_thisTextArea.tag\_config("font\_color", foreground=color[1])

def \_\_changeFontStyle(self):

font\_style = askstring("Font Style", "Enter Font Style (e.g. Arial, Algerian, Calibri, Arial Black, Times New Roman):")

if font\_style:

sel\_start, sel\_end = self.\_\_thisTextArea.index("sel.first"), self.\_\_thisTextArea.index("sel.last")

self.\_\_thisTextArea.tag\_add("font\_style", sel\_start, sel\_end)

self.\_\_thisTextArea.tag\_config("font\_style", font=(font\_style,))

def run(self):

# Run main application

self.\_\_root1.mainloop()

# Run main application

notepad1 = Notepad1(width=600, height=400)

notepad1.run()

***Review about Class And Institute:***

“I took the CIT with Python programming course at [Bano Qabil 2.0], and I am very happier with my experience.

The curriculum of this course was designed to provide a solid foundation in IT, with a particular focus on the Python programming language. The course content was well-structured and comprehensive, covering everything from basic knowledge about MS Office and programming concepts to more advanced topics. The lessons were easy to follow, even for someone like me who had little prior programming knowledge and experience. The course was designed in such a way that it was accessible and engaging for learners of all levels. It provides a strong foundation in IT. If you're looking to learn Python programming in a supportive and enriching environment, this course is definitely worth considering.

One standout aspect of the course was the presence of a female teacher. It was refreshing and inspiring to see a female instructor **(Miss Sumbul Mohammad** **Saleem)** as our course in charge. Miss Sumbul’s class is highly engaging and informative. She explains concepts in a clear and concise manner, making it easy for students to understand complex topics. Her teaching style is student-friendly, encouraging active participation and providing practical examples to reinforce learning.

Miss Sumbul is a dedicated and knowledgeable teacher who is passionate about her subjects. She goes the extra mile to ensure that every student grasps the material and is willing to provide extra help when needed. Her approachable nature creates a positive learning environment where students feel comfortable asking questions and seeking clarification.

Bano Qabil 2.0 Institute is a reputable institution known for its quality education in the field of IT. The institute offers a comprehensive curriculum, experienced faculty, and a supportive learning environment. Students benefit from hands-on training, real-world projects, and career guidance, preparing them for success in the technology industry.

The class was well-structured and the material was presented in a way that was easy to understand. Moreover, I found the learning environment at [Bano Qabil 2.0] to be top-notch. The classrooms were well-equipped with the necessary technology and resources. The staff and faculty were friendly, supportive, and always available to help with any queries or concerns.

Overall, I highly recommend the CIT with Python programming course at Bano Qabil”.

***Thank you***