```python
from tkinter import *
from tkinter import messagebox
from tkinter.ttk import Combobox
from tkinter.font import Font
from tkinter import simpledialog
from tkinter import filedialog


# arbitrary functions

def f_maximize():
    print("trying to maximize")
    # implementing input through pop-up window
    name = simpledialog.askstring(title="Name", prompt="Enter your name:")
    # simpledialog.askfloat()
    # simpledialog.askinteger()
    print(name)
    return


def f_minimize():
    print("trying to minimize")
    top = Toplevel()
    # will create another window
    top.title("Another window")
    top.geometry("300x300+150+150")
    # can add components just like the main window
    return


def f_close():
    print("trying to close")
    return


def do_nothing():
    print("in progress")
    return


def exit_prompt():
    # showinfo for showing any info
    # showerror for showing any error
    # showwarning for warning the user
    # messagebox.showwarning("Confirm", "Do you want to exit?")
    user_reply = messagebox.askquestion("Exit", "Do you want to exit?")
    if user_reply == "yes":
        main_frame.quit()
    return


def on_submit():
    print(name_entry.get())
    print(password_entry.get())
    # StringVar class can be used for the same purpose, but I prefer this way
    print(is_checked.get())
    print(gender.get())
    return


def on_select():
    indices = list_of_languages.curselection()
    # this method returns a tuple of indices of the items selected in the listbox
    for index in indices:
        print(index)
        print(list_of_languages.get(index))
        # this method takes an index as parameter and returns the item in that index of the listbox
    return


def on_delete():
    indices = list_of_languages.curselection()
    # this method returns a tuple of indices of the items selected in the listbox
    for index in indices:
        print(index)
        list_of_languages.delete(index)
        # this method takes an index as parameter and deletes that item of that index from the listbox
    return


def combobox_choose():
    print(cb.get())
    return


def spin_button_command():
    print(spin.get())
    return


def open_file():
    opened_file = filedialog.askopenfile(initialdir="/media/rmshoeb/Porashuna",
                                         title="Select file",
                                         filetypes=(("text files", "*.txt"), ("all files", "*.*")))
    # print(opened_file)# will print object location
    # to print original content
    try:
        for c in opened_file:
            print(c)
    except:
        print("No file opened")
    return


def save_file():
    f = filedialog.asksaveasfile(defaultextension=".txt")
    if f is None:
        return
    f.write("hello world")
    f.close()
    return


# try do use classes and functions for clean code
class MyButton:
    def __init__(self, master):  # master is the reference to the main_frame
        pass


# the main frame
main_frame = Tk()
main_frame.title("Main Window")
main_frame.geometry("750x500+120+120")  # setting a default size for the mainframe

# font
font_specify = Font(size=16, family="Times New Roman")
# arguments: weight(use to bold text, numeric value),
# slant(italicize, "italic/roman"), underline(0 or 1), overstrike(line through text, 0/1)

c_b = MyButton(main_frame)

# main menu creation and adding to main frame
main_menu = Menu(main_frame)
main_frame.config(menu=main_menu)

# file menu create and add to main frame
file_menu = Menu(main_menu)
main_menu.add_cascade(label="File", menu=file_menu)

# sub-menus of file menu
file_menu.add_command(label="New", command=do_nothing)
file_menu.add_command(label="Open", command=do_nothing)
file_menu.add_separator()  # add a separator line
# file_menu.add_command(label="Save", command=do_nothing)
# add a sub-menu
save_menu = Menu(file_menu)
```

```python
# submenus
save_menu.add_command(label="Save", command=do_nothing)
save_menu.add_command(label="Save as", command=do_nothing)

file_menu.add_cascade(label="Save", menu=save_menu)
file_menu.add_command(label="Close", command=do_nothing)

# edit menu creation and adding to main frame
edit_menu = Menu(main_menu)
main_menu.add_cascade(label="Edit", menu=edit_menu)

# all other frames
# frame = Frame(main_frame, width=300, height=300)#only works when there is nothing else, do not know why
input_frame = Frame(main_frame)
test_frame = Frame(main_frame)
listbox_frame = Frame(main_frame)
combobox_frame = Frame(main_frame)
text_frame = Frame(main_frame)
named_frame = LabelFrame(main_frame, text="named frame", padx=5, pady=5)

input_frame.grid(rowspan=5, columnspan=2)
listbox_frame.grid(row=0, rowspan=10, column=3)

# take inputs
# every input has a label and an input area, just like HTML
name_label = Label(input_frame, text="Name: ")
name_entry = Entry(input_frame)
password_label = Label(input_frame, text="Password: ")
password_entry = Entry(input_frame)
# is_checked              = IntVar()
is_checked = StringVar()
gender = StringVar()
# stay_logged_in_checkbox = Checkbutton(main_frame, text="Stay logged in?", variable=is_checked)
# creates a checkbox, text parameter is the label for it
stay_logged_in_checkbox = Checkbutton(input_frame, text="Stay logged in?", variable=is_checked,
offvalue="unchecked",
                                      onvalue="checked")
gender_label = Label(input_frame, text="Gender")
radio_button_male = Radiobutton(input_frame, text="Male", value="male", variable=gender)
radio_button_female = Radiobutton(input_frame, text="Female", value="female", variable=gender)
submit_button = Button(input_frame, text="Submit", command=on_submit)

# place elements according to grid
name_label.grid(row=0, column=0,
                sticky=E)  # sticky is to specify the position of label, here E represents east, i.e.
                right
name_entry.grid(row=0, column=1)
gender_label.grid(rowspan=2, column=0)
radio_button_male.grid(row=1, column=1, sticky=W)
radio_button_female.grid(row=2, column=1, sticky=W)
password_label.grid(row=3, column=0)
password_entry.grid(row=3, column=1)
stay_logged_in_checkbox.grid(columnspan=2)  # spans it to two columns but one row
submit_button.grid(row=5, column=1, sticky=E)

# listbox and scrollbar
listbox_label = Label(listbox_frame, text="Available now", font=font_specify, bg="yellow")
scroll_bar = Scrollbar(listbox_frame)
list_of_languages = Listbox(listbox_frame, width=30, height=10, selectmode=SINGLE,
yscrollcommand=scroll_bar.set)
# selectmode controls how many elements can be selected by mouse and how they can be selected
# BROWSE, MULTIPLE, EXTENDED
select_button = Button(listbox_frame, text="Select", command=on_select)
languages = ["C", "C++", "C#", "Java", "Python", "Ruby", "Visual Basic", "Assembly", "PHP", "Perl", "R",
"SAS",
             "JavaScript"]
# for i in range(len(languages)):
for language in languages:
    list_of_languages.insert(END, language)
# list_of_languages.insert(1, "C")
# list_of_languages.insert(2, "C#")
# listbox_label.grid(row=0, column=2)
listbox_label.pack()
```

```python
# list_of_languages.grid(row=1, column=2)
list_of_languages.pack(side=LEFT)
scroll_bar.pack(side=LEFT, fill=Y)
scroll_bar.config(command=list_of_languages.yview)
# select_button.grid(row=10, column=2, sticky=E)
select_button.pack(side=BOTTOM)
# have to find a way to add the scrollbar using grid(), because pack() is quite disappointing

# creating and configuring buttons
button1 = Button(test_frame, text="Maximize", command=f_maximize)
button2 = Button(test_frame, text="Minimize", command=f_minimize)
# button3 = Button(test_frame, text="Close", command=main_frame.quit)
button3 = Button(test_frame, text="Close", command=exit_prompt)

# since the label is placed before buttons are packed, it will appear before the buttons, if used packing
# this won't affect in grid system though
the_label = Label(main_frame, text="a very long and useless but necessary text", font=font_specify,
bg="yellow")

# placing the buttons in the frame
button1.pack(side=LEFT)
button2.pack(side=LEFT)
button3.pack(side=RIGHT)

# found out you can't use packing if you are using grid somewhere
# and I like grid system, feels safe
# test_frame.pack()
test_frame.grid(columnspan=3)
# the_label.pack(fill=X)
the_label.grid(columnspan=3)
# there is another method, called place(), this works somewhat like grid system
# but instead of using row and columns, it uses (x, y) co-ordinate system
# it can be take some time to find the correct co-ordinates to place the components correctly
# i'll still use grid system

# Combobox
# combobox_frame.grid()
# creates a dropdown menu
v = list(range(1, 32))
cb = Combobox(combobox_frame, values=v, width=10, height=10)
# values takes the list of values for the menu
# width is how many characters will show in a single row
# height is how many rows will show at a time
cb.set("Select")  # kind of a message to show for user
combobox_button = Button(combobox_frame, text="Choose", command=combobox_choose)
# cb.grid()
# combobox_button.grid()

# textbox
# text_frame.grid(columnspan=4)
text_box = Text(text_frame, height=20, width=100, wrap=WORD, padx=10, pady=10)
text_box.configure(font=font_specify)
# text_box.grid()

# spinbox
named_frame.grid()
spin = Spinbox(named_frame, from_=1, to=5)
spin_button = Button(named_frame, text="Spin", command=spin_button_command)
open_file_button = Button(named_frame, text="Open file", command=open_file)
save_file_button = Button(named_frame, text="Save File", command=save_file)
spin.grid()
spin_button.grid()
open_file_button.grid()
save_file_button.grid()

main_frame.mainloop()
```