Explication du code python realisé

Le code realisé est le suivant

```
Import visa
import time
from tkinter import *
from tkinter.messagebox import *
from matplotlib.backends.backend_tkagg import
        FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
```

```
def get_values(*args):
    freq.get()
    unit.get()
    ampli.get()
    offset.get()
    type_sig.get()
```

Defining the function that obtains initial values from the user

---

```
def pause():
    global a
    a = 1
```

pause function

```
def close():
    if askyesno('Warning', 'If you continue every
                information will be discarded!'):
        root.quit()
        root.destroy()
```

Adding a warning before closing the GUI

```
def send_cmds():
    global a, i, tab_am, tab_frq, tab_ph
    a = 2
    frq = 1
    am = 1
    ph = 1
    instr = 0
    instr1 = 0
    get_values()
```

Defining the function that process data, sends, receives, treats and plot data

Making the variables accessible to all the program

Initializing the values of the variables used inside this function

***a** is a viable for pause/resume*

```
    while True:
        spAmpl.cla()
        spPhas.cla()
```

Declaring a loop that

```
        if a > 1:
            cmds = "APPL:{0} {1:.2f}{2}, {3}VPP,{4}V".format
                    (type_sig.get(), float(freq.get())+i, unit.get(),
                     float(ampli.get()), float(offset.get()))
            instr.write(cmds)
            time.sleep(2)
            instr1.write(':AUToscale')
```

*start button is pressed(**a** is 2) => condition is true*

***i** is a variable that increment the frequency*

Commands are sent
We wait 2 seconds then
we auto scale

```
            ta = am
            if (ta + am) >= 50 or (ta + am) <= -50:
                am = ta
            tf = frq
            if frq >= 10e20:
                frq = tf
```

```
            frq = float(instr1.query(':MEASure:FREQuency?'))
            am = float(instr1.query(':MEASure:VAMPLitude?'))
            ph = float(instr1.query(':MEASure:PHASe?'))
```

Receiving values from the oscillo
And storing them into variables

```
            tab_am.append(am)
            tab_frq.append(frq)
            tab_ph.append(ph)
```

Appending this variables into a list

```
            spAmpl.semilogx(tab_frq, tab_am, 'r')
            spPhas.semilogx(tab_frq, tab_ph, 'r')
            canvas.show()
```

Plotting this lists into a semi-logarithmic graph

```
            i = i*1.1
            root.update()
```

*Incrementing **i** by 10%*
The purpose of this update is to

```
        if a == 1:
            spAmpl.cla()
            spAmpl.semilogx(am, frq, 'b')
            spPhas.cla()
            spPhas.semilogx(ph, frq, 'b')
            root.update()
```

*pause button is pressed(**a**=1) => condition is true*

Declaration/Initialization

```
rm = 0
list = 0
```

Visa variables usage

```
root = Tk()
root.wm_title("BodePlotFilter.py")
```

Empty window creation

```
title = Label(frame, text="GBF Set-UP", font=("Helvetica", 22), fg="blue")
title_graph = Label(root, text = "Bode Representation\t\t\t", font=("Helvetica", 14), bg="#f0f0f0", fg="black")
title_frame = Label(frame, text = "Signal", font=("Helvetica", 14))
type_sig = StringVar()
type_sig.set("###")
```

Window composition

GUI creation

```
fgBode = Figure(figsize=(11,5), dpi=75)
fgBode.suptitle('Bode plot', fontsize=12)
fgBode.subplots_adjust(hspace=0.28)
```

Frame creation

```
spAmpl = fgBode.add_subplot(121)
spAmpl.set_title('Amplitude',fontdict={'fontsize':10})
```

Amplitude graph set-up

```
spPhas = fgBode.add_subplot(122, axisbg='cyan')
spPhas.set_title('Phase',fontdict={'fontsize':10})
```

Phase graph set-up

```
frPlt = Frame(root)
```

Workspace

```
cvPlot = FigureCanvasTkAgg(fgBode, master=frPlt)
cvPlot.show()
```

Graphs Plot

```
tbPlot = NavigationToolbar2TkAgg(cvPlot, frPlt)
tbPlot.update()
```

Toolbar

```
canvas = FigureCanvasTkAgg(fgBode, master=root)
canvas.show()
fgBode.tight_layout()
canvas.get_tk_widget().grid(column=1, row=10)
```

Graphs Set-up

Window elements creation

```
b1 = Radiobutton(frame, text="Sinusoid\t\t\t", variable=type_sig, value="SIN")
b2 = Radiobutton(frame, text="Triangle\t\t\t", variable=type_sig, value="RAMP")
b3 = Radiobutton(frame, text="Square\t\t\t", variable=type_sig, value="SQUARE")
```

Signal selection

```
freq = Spinbox(frame, from_=0, to=1000000000000, width=10)
freq_title = Label(frame, text = "Freq")

unit = Spinbox(frame, values=('HZ', 'KHZ', 'MHZ', 'GHZ'), width=5)

ampli = Spinbox(frame, from_=0, to=1000000000000, width=10)
ampli_title = Label(frame, text = "Amp")

offset = Spinbox(frame, from_=0, to=1000000000000, width=10)
offset_title = Label(frame, text = "Offset")
```

value selection

```
button_start = Button(frame, text="Start", command=send_cmds, width=10, font=("Helvetica", 14),
bg="#00ff80", activebackground="#00ff80", relief=GROOVE)
button_pause = Button(frame, text="Pause / Reset", command=pause, width=12, font=("Helvetica", 9),
relief=GROOVE)
button_close = Button(master=frame, text='Close', command=close, bg="#ff0000",
activebackground="#ff0000",fg="white", relief=GROOVE)
```

Cmd btns

Window elements location

```
frame.grid(column=1, row=0)
```

Empty window as grid

```
title.grid(column=1, row=1, columnspan=3)
title_frame.grid(column=1, row=2, sticky='nsw')
title_graph.grid(column=1, row=9, columnspan=3, sticky='n')
```

Name location inside the grid

```
b1.grid(column=1, row=3, sticky='nsw')
b2.grid(column=1, row=4, sticky='nsw')
b3.grid(column=1, row=5, sticky='nsw')
```

Buttons location inside the grid

```
freq_title.grid(column=2, row=3, sticky='nsw')
freq.grid(column=3, row=3, columnspan=1, sticky='nse')

unit.grid(column=4, row=3, sticky='nsw')

ampli_title.grid(column=2, row=4, sticky='nsw')
ampli.grid(column=3, row=4, sticky='nse')

offset_title.grid(column=2, row=5, sticky='nsw')
offset.grid(column=3, row=5, sticky='nse')
```

Options Selection location inside the grid

```
empty = Label(frame, width=40).grid(row=0, column=5)
```

Empty column

```
button_start.grid(column=5, row=8, sticky='nse')
button_pause.grid(column=6, row=8, sticky='nse')
button_close.grid(column=7, row=8, sticky='nsw')
```

Buttons location inside the grid

```
frPlt.grid(column=1, row=11, columnspan=3, sticky='nsw')
```

Plot location inside the grid

```
root.mainloop()
```

GUI set-up loop