```
            return "INVALID"
    else:
        return "SYNTAX"

# ============================== controller ================================

# Initialize controller algorithm
def contrInit():
    pass


# Control background processing called at T = 10 mSec
def contrExe():
    b24 = gl.comm.fetch24()
    if gl.comm.validate24(b24):
        updatePars(b24)
        gl.gui.updatePars()
    else:
        gl.gui.emptyPars()          # communication error


# processing of command line parameter
def commandline_pars():
    if len(sys.argv) >= 3 and sys.argv[1] == "-p":
        gl.port = sys.argv[2]       # Set serial communication port ex. COM3


# Edit widged <CR> pressed
def editCr(event):
    s = gl.gui.getEditText()
    gl.gui.setEditText(interpreter(s))

# ============================== main ================================

# Main entry point of this application. Instantiating global objects
def main():
    try:
        window = tk.Tk()
        window.title("VE03327 monitor rev0.11")
#       window.geometry("1350x300")
        gl.gui  = gui.GUI(window)
        gl.comm = cm.com3327(gl.port)
        contrInit()
        window.mainloop()
    except:
        print("\nUnhandled exception :", sys.exc_info()[0], sys.exc_info()[1])

# ============================== Start ================================

# Prevent starting of main when loaded as library
if __name__ == "__main__":
    commandline_pars()
    main()
    sys.exit(0)         # not required for exit code 0

# ============================== End ================================
```