ntk

Nj Nafir

Jul 01, 2021

CONTENTS:

—

CHAPTER

ONE

OVERVIEW

NTK is set of widgets and utils that implement high-level APIs for accessing many aspects of modern desktop systems. These include database management with orm, widgets like js components, as well as desktop UI development in less code.

NTK is a comprehensive set of Python bindings for tkinter and ttk. It is a comprehensive and easy to use GUI library for python. It lets you create attractive design in a quick and convenient way.

It also comes with a dynamic ORM that lets you easily create update and delete your fully functioning database system. It's also packed with high quality widgets with a variety of choices that can be easily configured to your application.

1.1 Installation

The latest version of ntk can be installed from PyPI:

```
pip install ntk
```

1.2 Quick use

Copy code from below and paste into a python file, then see the magic difference between base tkinter and ntk

```
from ntk import Tk
def main(): root = Tk() root.mainloop()
if __name__=='__main__': main()
```

1.3 Author

NTK is a open source libraries for Python, Initially developed by Nj Nafir, Everyones can contribute to build NTK as friendly and handy library.

1.4 Contribute

- Issue Tracker: NTK Issues
- Source Code: NTK Sources

1.5 Support

If you are having issues, please let us know. We have a mailing list located at: njnafir@gmail.com

1.6 License

The project is licensed under the MIT license.

CHAPTER

TWO

OBJECTS

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

2.1 gv

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

gv stands for global variable, which is a file object, we can set attribute to it and whenever time we need we can import the gv from ntk and get and set value from this object

simple example:

```
gv.myname = "John"
```

from another file we can import this file object and get attribute value

`name = gv.myname` or we can simply use it by `gv.myname`

2.2 basev

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

basev stands for base variable which is used by ntk initially, but you can change it to control ntk defaults

the only function is defined here is `base_var`, and it is

```
def base_var():
    gv.db_name = "sqlite3.db"
    gv.db_timeout = 10
    gv.check_same_thread = False
    gv.db = {}
    gv.cr = {}
    gv.cache = {}
    gv.models = {}
```

we can change these by gv object also by

```
from ntk import gv
gv.db_name = "mydb"
```

but for most cases we don't need to do that

CHAPTER

THREE

DATABASE

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

CHAPTER

FOUR

WIDGETS

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

4.1 Tk

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

ntk Tk is extended version of tkinter base Tk with more functionality and responsive grid system, to use this Tk window we need to import first it from ntk by

```
from ntk import Tk
```

and initialize it by calling it

```
root = Tk() root.mainloop()
```

This will create tk window and basic grid will be applied, you need to pass parameters described below to get your desired window size, style and grid.

available parameters are:

- `title="Main Window"`, # title to showing on top bar
- `resize_x=True`, # resize x is to set horizontal resizable
- `resize_y=True`, # resize y is to set vertical resizable
- `width=360`, # tkinter window width
- `height=380`, # tkinter window height
- `x=False`, # tkinter window positioning root x
- `y=False`, # tkinter window positioning root y
- `state="normal"`, # tkinter window state
- `bg="bg-white"`, # background of tkinter window, default is bootstrap referenced white
- `icon=False`, # tkinter window icon
- `mainframe=True`, # set if tkinter hold a main frame window or not
- `fullscreen=False`, # set if you want to set this window to full window
- `gridrow=1`, # grid configure row position
- `gridcolumn=1`, # grid configure column position

- `topbar=True`

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Tk class.

4.2 Toplevel

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Toplevel is window like Tk window, but it has no mainloop method so it can be used for sub window

ntk Toplevel is extended version of tkinter base Toplevel with more functionality and responsive grid system, to use this Toplevel window we need to import first it from ntk by

```
from ntk import Toplevel
```

and initialize it by calling it

```
window = Toplevel(root)
```

This will create window and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root=None`, # root is a master window to place this toplevel into it
- `title="Toplevel"`, # title to showing on top bar
- `bg="bg-white"`, # background color, default is bootstrap referenced white
- `bd=0`, # border width
- `class_="Toplevel"`, # class is important when you want to inherit any design or some methods
- `colormap=False`, # color map
- `container=0`, # container
- `cursor="arrow"`, # cursor style for toplevel
- `height=480`, # toplevel window height
- `width=360`, # toplevel window width
- `highlightbackground="bg-light"`, # background color when toplevel is highlighted default is bootstrap referenced light
- `highlightcolor="fg-dark`", # foreground color when toplevel is highlighted default is bootstrap referenced dark
- `highlightthickness=0`, # thickness width of toplevel when highlighted
- `menu=False`, # top level menu can be set with this command
- `padx=0`, # grid padding left and right
- `pady=0`, # grid padding top and bottom
- `relief="flat"`, # toplevel relief style
- `screen=""`, # toplevel screen
- `takefocus=1`, # set if toplevel window can take focus or not
- `use=False`, # use

- `visual=False`, # visual
- `resize_x=0`, # toplevel window resizing horizontally is allowed or not
- `resize_y=0`, # toplevel window resizing vertically is allowed or not
- `x=False`, # toplevel window positioning left right position
- `y=False`, # toplevel window positioning top bottom position
- `topbar=True`

an example of creating Toplevel window:

```
from ntk import Tk, Toplevel
root = Tk(title='Example of ntk window')
sub_window = Toplevel(root, title='Example of sub window')
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Toplevel class.

4.3 Panedwindow

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

PanedWindow is widget wrapper for ntk window, it's highly responsible, it can give us resizable box layout

ntk PanedWindow is extended version of tkinter base PanedWindow with more functionality and responsive grid system, to use this PanedWindow window we need to import first it from ntk by

```
from ntk import PanedWindow
```

and initialize it by calling it

```
paned_window = PanedWindow(root)
```

This will create window and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this panedwindow into it
- `bg="bg-light"`, # background color, default is bootstrap referenced light
- `bd=2`, # border width
- `cursor="arrow"`, # cursor style for paned window
- `handlepad=0`, # panedwindow subwidgets handling bar, this also work as bar width
- `handlesize=0`, # panedwindow subwidgets handling bar, this also work as bar width
- `height=370`, # panedwindow height
- `opaqueresize=0`, # opaque resize
- `orient="vertical"`, # panedwindow subwidgets orientation
- `relief="flat"`, # panedwindow relief style flat groove etc
- `sashcursor="sizing"`, # panedwindow subwidgets handling sash bar cursor

- `sashpad=0`, # panedwindow subwidgets handling sash bar show or hide
- `sashrelief="flat"`, # panedwindow subwidgets handling sash bar relief
- `sashwidth=4`, # panedwindow subwidgets handling sash bar width
- `showhandle=True`, # sash handle with handle pad
- `width=370`, # panedwindow width
- `row=0`, # grid row position
- `column=0`, # grid column position
- `rowspan=1`, # grid row span width
- `columnspan=1`, # grid column span width
- `padx=(0, 0)`, # grid padding left and right
- `pady=(0, 0)`, # grid padding top and bottom
- `ipady=0`, # grid internal padding top and bottom
- `sticky="wsen"`, # grid sticky position
- `gridrow=1`, # grid configure row config weight
- `gridcolumn=1`, # grid configure column config weight

ntk Tk window already contain a PanedWindow object by default because we passed mainframe=True we can access it by root.mainframe

an example of creating PanedWindow window:

```
from ntk import Tk, PanedWindow
root = Tk(title='Example of PanedWindow in ntk window')
paned_window = PanedWindow(root)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter PanedWindow class.

4.4 Notebook

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Notebook is multi window wrapper for ntk window, we can create a notebook and add tabs for getting multi window in tabs

ntk Notebook is extended version of tkinter ttk/tcl-tk base Notebook with more functionality and responsive grid system, to use this Notebook window we need to import first it from ntk by

```
from ntk import Notebook
```

and initialize it by calling it

```
window = Notebook(root)
```

This will create wrapper and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this notebook into it
- `class_="TNotebook"`, # notebook class to inherit object
- `cursor="arrow"`, # cursor style for notebook window
- `height=180`, # notebook height
- `width=350`, # notebook width
- `takefocus=0`, # set notebook window can take focus or not
- `padding=0`, # padding in square window
- `bg="bg-light"`, # background color, default is bootstrap referenced light
- `bd=0`, # border width
- `fg="fg-dark"`, # foreground color, default is bootstrap referenced dark
- `lightcolor="fg-info`", # light color, default is bootstrap referenced info
- `row=0`, # grid row position
- `column=0`, # grid column position
- `sticky="wn"`, # grid sticky position
- `rowspan=1`, # grid row span width
- `columnspan=1`, # grid column span width
- `padx=1`, # grid padding left and right
- `pady=1`, # grid padding top and bottom
- `style=False`, # notebook default style

an example of creating Notebook window:

```
from ntk import Tk, Notebook, PanedWindow
root = Tk(title='Example of ntk window')
notebook = Notebook(root)
root.mainloop()
```

Notebook wrapper has a method called add, which can be used for adding a tab in Notebook for doing this,

```
panedwindow = PanedWindow(notebook) # create a panedwindow to add it into NoteBook
notebook.add(child=panedwindow, text='First tab')
```

we can pass other parameters to add method, and those are,

- `width=16`, # tab head title width/text width
- `image=False`, # not supported yet
- `compound="left"`, # can be used for image
- `underline=99`, # underline position for tab header text
- `sticky="wn`", # sticky position
- `padding=0` # padding

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Notebook class.

4.5 Frame

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Frame is multi widget wrapper for ntk window, we can create a frame and separate sub widgets in main window by combining into frames

ntk Frame is extended version of tkinter base Frame with more functionality and responsive grid system, to use this Frame window we need to import first it from ntk by

```
from ntk import Frame
```

and initialize it by calling it

```
window = Frame(root)
```

This will create wrapper and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this frame into it
- `bg="bg-white"`, # background color, default is bootstrap referenced white
- `bd=0`, # border width
- `colormap=None`, # color map
- `class_=False`, # class name to inherit styles and methods
- `container=0`, # container
- `cursor="arrow"`, # mouse cursor style arrow hand2 etc
- `height=64`, # frame height
- `highlightbackground="bg-light"`, # background color when frame is highlighted, default is bootstrap referenced light
- `highlightcolor="bg-dark"`, # foreground color when frame is highlighted, default is bootstrap referenced dark
- `highlightthickness=0`, # thickness width when frame is highlighted
- `row=0`, # grid row position
- `column=0`, # grid column position
- `padx=0`, # grid padding left and right
- `pady=0`, # grid padding top and bottom
- `relief="flat"`, # relief style flat groove etc
- `sticky="w"`, # grid sticky position
- `takefocus=0`, # set frame can take focus or not
- `visual=0`, # visual
- `width=128`, # frame width
- `gridrow=1`, # grid row configure row weight
- `gridcolumn=1`, # grid column configure column weight

an example of creating Frame widget:

```
from ntk import Tk, Frame
root = Tk(title='Example of ntk window')
frame = Frame(root)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Frame class.

4.6 Canvas

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Canvas is highly customizable and most usedful widget for drawing anything

ntk Canvas is extended version of tkinter base Canvas with more functionality and responsive grid system, to use this Canvas window we need to import first it from ntk by

```
from ntk import Canvas
```

and initialize it by calling it

```
window = Canvas(root)
```

This will create wrapper and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this button into it
- `bg="bg-white"`, # background color
- `highlightbackground="bg-white"`, # background color when canvas is highlighted
- `highlightcolor="fg-dark"`, # foreground color when canvas is highlighted
- `selectbackground="bg-primary"`, # element background color when canvas element is selected
- `scrollregion=[0,0,350,96]`, # [x1, y1, x2, y2] region when canvas is scrolling via scrollbar or mouse
- `relief="flat"`, # relief design can be flat, groove etc
- `width=350`, # canvas width
- `height=96`, # canvas height
- `row=0`, # row position
- `column=0`, # column position
- `rowspan=1`, # row spanning size
- `columnspan=1`, # column spanning size
- `padx=1`, # padding in left and right
- `pady=1`, # padding in top and bottom
- `mousescroll=True`, # set False if you don't want to scrolling when scrolling via mouse
- `gridcolumn=1`, # set 0 if you don't want responsiveness by column in it's root window

- `gridrow=1`, # set 0 if you don't want responsiveness by row in it's root window

an example of creating Canvas widget:

```
from ntk import Tk, Canvas
root = Tk(title='Example of ntk window')
canvas = Canvas(root)
root.mainloop()
```

canvas widget have some other custom method to get extra power

select_clicked is one of method which can be used for selecting clicked item from canvas widget for doing this, we can call it in anywhere, using callback or event binding

```
canvas.select_clicked()
```

mousewheel method is used by canvas itself, to scroll on canvas height when scrolling by mouse

increase_scrollragion is can be used to increase canvas scrollable area, it take's four parameters

`x1=False`, # area start left position

`y1=False`, # area start top position

`x2=False`, # area start right position

`y2=False` # area start bottom position

decrease_scrollragion is can be used to decrease canvas scrollable area, it take's four parameters

`x1=False`, # area start left position

`y1=False`, # area start top position

`x2=False`, # area start right position

`y2=False` # area start bottom position

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Canvas class.

4.7 Scroller

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Scrollbar is scroll maintaining widget in ntk, we can create a scroller widget to set it with other master window like canvas

ntk Scrollbar is extended version of tkinter base Scrollbar, with more functionality, responsive grid system and with automation, to use this Scrollbar window we need to import first it from ntk by

```
from ntk import Scrollbar
```

and initialize it by calling it

```
scroller = Scrollbar(root)
```

This will create a scroller in given orient and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this scrollbar into it

- `scroll_on=False`, # set scroll on window/widget, set scroll effect on this window
- `orient="vertical"`, # scroll orient left-right or top-bottom
- `width=12`, # scrollbar width
- `row=0`, # grid row position
- `column=1`, # grid column position
- `columnspan=1`, # grid column span
- `padx=0`, # grid padding left and right
- `pady=0`, # grid padding top and bottom
- `sticky="ns"`, # grid sticky position
- `activebg="bg-light"`, # background color when scrollbar is active, default is bootstrap referenced light
- `activerelief="flat"`, # relief style when scrollbar is active
- `bg="bg-dark"`, # background color, default is bootstrap referenced dark
- `bd=0`, # border width
- `cursor="arrow"`, # cursor style for scrollbar
- `elementborderwidth=0`, # element border width
- `highlightbg="bg-primary"`, # background color when bar is highlighted, default is bootstrap ref primary
- `highlightcolor="fg-dark"`, # foreground color when bar is highlighted, default is bootstrap ref dark
- `highlightthickness=1`, # bar thickness width when bar is highlighted
- `jump=True`, # jump scrolling when clicked on bar
- `relief="flat"`, # relief style for bar
- `repeatdelay=300`, # bar repeat delay
- `repeatinterval=100`, # bar repeat interval
- `takefocus=1`, # set if bar can take focus or not
- `troughcolor="fg-secondary"`, # bar trough color, default is bootstrap referenced secondary
- `gridrow=1`, # grid configure row weight
- `gridcolumn=1`, # grid configure column weight

an example of creating Scrollbar widget:

```
from ntk import Tk, Scrollbar, Canvas
root = Tk(title='Example of ntk window')
canvas = Canvas(root)
scroller = Scrollbar(root, scroll_on=canvas)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Scrollbar class.

4.8 Entry

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Entry is a box widget in ntk where we can write anything, and it has more functionality to get and set value, binding events etc

ntk Entry is extended version of tkinter base Entry, with more functionality, responsive grid system and with automation, to use this Entry window we need to import first it from ntk by

```
from ntk import Entry
```

and initialize it by calling it

```
entry = Entry(root)
```

This will create a entry in given grid and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this entry into it
- `bg="bg-light"`, # background color, default is bootstrap referenced light
- `bd=0`, # border width of entry widget
- `cursor="xterm`", # default cursor style
- `dbg="bg-warning"`, # background color when entry is disabled, default is bootstrap referenced warning
- `dfg="fg-dark"`, # foreground color when entry is disabled, default is bootstrap referenced dark
- `eselect=1`, # when entry text is selected it will automatically exported to clipboard
- `font=("Calibri", 11)`, # font styles
- `fg="fg-secondary"`, # foreground color, default is bootstrap referenced secondary
- `hlbg="bg-primary"`, # background color when entry is highlighted, default is bootstrap referenced primary
- `hlc="fg-primary"`, # foreground color when entry is highlighted, default is bootstrap referenced primary
- `hlt=1`, # thickness width when entry is highlighted
- `ibg="bg-dark"`, # background color for inserted char, default is bootstrap referenced dark
- `ibd=0`, # border width for inserted char
- `iofftime=500`, # insert off time for shuffling insert bar
- `iontime=500`, # insert on time for shuffling insert bar
- `iwidth=2`, # insert bar width
- `invcmd=""`, # inv cmd line
- `justify="left"`, # entry text justify left right center
- `rbg="bg-white"`, # background color when entry is readonly, default is bootstrap referenced white
- `relief="flat"`, # entry relief style flat groove etc
- `sbg="bg-primary"`, # entry selection background, default is bootstrap referenced primary
- `sbd=2`, # border width of selection text
- `sfg="fg-white"`, # entry selection foreground, default is bootstrap referenced white

- `show=None`, # show is to define text visual, if you set it to * all text chars will be visible as * but it's main value original
- `state="normal"`, # entry state normal disable readonly etc
- `takefocus=1`, # set entry widget can take focus or not
- `tvar=None`, # text variable for entry to get and set value dynamically
- `validate=None`, # validate
- `vcmd=None`, # vc md
- `width=24`, # entry width
- `xscrollcommand=None`, # scrolling command when scrolling in left-right position
- `row=0`, # grid row position
- `column=0`, # grid column position
- `columnspan=1`, # grid column span position
- `rowspan=1`, # grid row span position
- `padx=10`, # grid padding left and right
- `pady=10`, # grid padding top and bottom
- `ipady=2`, # grid internal padding top and bottom
- `sticky='w'`, # grid sticky position
- `default=""`, # entry widget default value
- `focusinbg="bg-white"`, # background color when entry is focused, default is bootstrap referenced white
- `focusoutbg=False`, # background color when entry is unfocused, default is False means it will set main bg again
- `focusinhlc="bg-warning"`, # foreground color when entry is focused, default is bootstrap referenced warning
- `focusouthlc=False`, # foreground color when entry is unfocused, default is False means it will set main bg again

an example of creating Entry widget:

```
from ntk import Tk, Entry
root = Tk(title='Example of ntk window')
entry = Entry(root)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Entry class.

Entry has another method called set, and set method can be used for setting text value in it. like this

```
entry.set("New value")
```

4.9 SelectBox

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

SelectBox is a box widget, where we can show up a list of possible options, we can scroll on it and we can bind events for advance handling

ntk SelectBox is merged version of ntk Entry, Toplevel and Canvas, with more functionality, responsive grid system and with automation, to use this SelectBox window we need to import first it from ntk by

```
from ntk import SelectBox
```

and initialize it by calling it

```
SelectBox = SelectBox(root)
```

This will create a SelectBox in given grid and basic style will be applied, you need to pass parameters described below

available parameters are:

- `root`, # root is a master window to place this entry into it
- `values=['Values:list/tuple']`, # values can be a list or tuple
- `height=10`, # height of value list
- `default=True`, # default value in entry box
- `selectcommand=False`, # to perform when any value is selected
- `bg='#F4F4F4'`, # background color of select box
- `onclick=''`, # perform set rule when entry box clicked

an example of creating SelectBox widget:

```
from ntk import Tk, SelectBox
root = Tk(title='Example of ntk window')
select = SelectBox(root, values=['First', 'Second', 'Third'])
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to ntk Entry class.

SelectBox class have other custom method, which can be used to handle SelectBox widget

SelectBox.show_selection() method is can be used to pop up the list of selection options in Once

```
select.show_selection()
```

it has passable three parameters

- `e=None`, # event object
- `show=False`, # list show or hide
- `values=False` # pass value list another time

SelectBox.update_list() method is can be used to reset list of values from box

```
select.update_list()
```

SelectBox.mouse_entered() method is used by SelectBox object itself to set option hover style

```
select.mouse_entered(1, 2)
```

SelectBox.mouse_leaved() method is used by SelectBox object itself to set option hover removed style

```
select.mouse_leaved(1, 2)
```

SelectBox.select_text() method is used by SelectBox object itself to select text and call selectcommand

```
select.select_text("Third")
```

SelectBox.typed() method is used by SelectBox object itself bind method when key pressed in entry

`select.typed(e)` # e is event object

SelectBox.destroy_list() method is used by SelectBox object itself bind method when we want to destroy the list

`select.destroy_list(e)` # e is event object

4.10 ImageFile

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

ImageFile is a image object creator class, which can be used to automat image object creation

ntk ImageFile is extended version of tkinter base PhotoImage and Pillow Image, ImageTk, with more functionality, responsive grid system and with automation, to use this ImageFile window we need to import first it from ntk by

```
from ntk import ImageFile
```

and initialize it by calling it

```
ImageFile = ImageFile()
```

This will create a ImageFile in given grid and basic style will be applied, you need to pass parameters described size, orient, style

available parameters are:

- `file=False`, # file url
- `resize=False`, # resize param is a tuple acceptable by PIL resize
- `format="png"`, # get and save images in this file format
- `pillow=True`, # if pillow is False, ntk will be return a object which opened by tkinter PhotoImage

an example of creating ImageFile widget:

```
from ntk import Tk, ImageFile

root = Tk(title='Example of ntk window')

image = ImageFile("C:\\Users\User\Desktop\photo.png")

root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter ImageFile class.

set_image method is used by ntk ImageFile class, but you can use it for your purpose like set an image again after previous sets, like,

```
image.file = "C:\\Users\User\Desktop\photo2.png" image.set_image()
```

4.11 Label

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Label is a most used text wrapping widget for tkinter

ntk Label is extended version of tkinter base Label, with more functionality, responsive grid system and with automation, to use this Label window we need to import first it from ntk by

```
from ntk import Label
```

and initialize it by calling it

```
Label = Label(root)
```

This will create a Label in given grid and basic style will be applied, you need to pass parameters described below

available parameters are:

- `root`, # root is a master window to place this label into it
- `text="New label"`, # label text value
- # bg="bg-light",
- # fg="fg-dark",
- `var=None`, # label text variable
- `case="lower"`, # text style lower upper etc
- `width=False`, # label width
- `image=None`, # label image
- `image_file=False,` # image file to getting image object from it
- `image_size=(32, 32)`, # image size to getting image in custom size from image file
- `position="left"`, # image and text position left right center
- `font=('Calibri', 10)`, # label font style
- `row=0`, # grid row position
- `column=0`, # grid column position
- `rowspan=1`, # grid row span
- `columnspan=1`, # grid column span
- `padx=(5, 5)`, # grid padding left and right
- `pady=(5, 5)`, # grid padding top and bottom
- `ipady=10`, # grid internal padding top and bottom
- `sticky='w'`, # grid sticky position
- `length=False`, # label wrap length width

an example of creating Label widget:

```
from ntk import Tk, Label

root = Tk(title='Example of ntk window')

image = Label(root)
```

```
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Label class.

4.12 Text

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Text is a box widget, where we can type text in multi line, we can scroll on it and we can bind events for advance handling

ntk Text is extended version of tkinter base Text, with more functionality, responsive grid system and with automation, to use this Text window we need to import first it from ntk by

```
from ntk import Text
```

and initialize it by calling it

```
Text = Text(root)
```

This will create a Text in given grid and basic style will be applied, you need to pass parameters described below

available parameters are:

- `root`, # root is a master window to place this text into it
- `sep=1`, # auto separator
- `bg="bg-light"`, # background of text widget, default is bootstrap referenced light
- `bd=0`, # border width
- `cursor="xterm"`, # cursor style to showing in insert position
- `eselection=1`, # export selection to clipboard when text is selected
- `font=("Calibri", 10)`, # font style
- `fg="fg-dark"`, # foreground of text widget, default is bootstrap referenced dark
- `height=12`, # height of text widget
- `hlbg="bg-light"`, # background when text widget is highlighted, default is bootstrap referenced light
- `hlc="bg-light"`, # foreground when text widget is highlighted, default is bootstrap referenced light
- `hlt=1`, # thickness width of text when it's highlighted
- `ibg="bg-light"`, # background of inserted position, default is bootstrap referenced light
- `ibd=0`, # insert position border width
- `iofftime=500`, # insert off time for text widget
- `iontime=1000`, # insert on time for text widget
- `iwidth=1`, # insert position width
- `maxundo=1`, # max undo with control z
- `padx=5`, # grid padding left and right
- `pady=5`, # grid padding top and bottom
- `relief="flat"`, # relief style for text widget

- sbg="bg-primary", # background color of select background, default is bootstrap referenced primary
- sbd=0, # select border width
- sfg="fg-white", # foreground color of select background, default is bootstrap referenced primary
- setgrid=0, # set grid
- spacing1=0, # first spacing
- spacing2=0, # second spacing
- spacing3=0, # third spacing
- state="normal", # text widget state
- tabs=None, # text widget tabs
- takefocus=1, # set if text widget can take focus or not
- undo=1, # set undo length
- width=48, # text widget width
- wrap="char", # text widget wrapping method char or word
- xscroll=None, # horizontal scroll
- yscroll=None, # vertical scroll
- row=0, # grid row position
- column=0, # grid column position
- columnspan=1, # grid column span
- rowspan=1, # grid row span
- sticky="w", # grid sticky position
- tvar=False, # text variable for dynamic get and set

an example of creating Text widget:

```
from ntk import Tk, Text
root = Tk(title='Example of ntk window')
text = Text(root)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Text class.

Text class have other custom method, which can be used to handle text widget

text.clear() method is can be used to clear all text from this text widget

```
text.clear()
```

text.get() method is can be used to get text widget value

```
text.get()
```

text.set() method is can be used to set text widget value

```
text.set("New value")
```

4.13 Combobox

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to good looking and os level implementation.

Combobox is a box widget in ntk where we can show up selection list

ntk Combobox is extended version of tkinter base Combobox, with more functionality, responsive grid system and with automation, to use this Combobox window we need to import first it from ntk by

```
from ntk import Combobox
```

and initialize it by calling it

```
Combobox = Combobox(root)
```

This will create a Combobox in given grid and basic style will be applied, you need to pass parameters described below to get your desired window size and style

available parameters are:

- `root`, # root is a master window to place this combobox into it
- `class_="TCombobox"`, # combobox class which can be inherited
- `cursor="arrow"`, # cursor style when mouse over combobox
- `exportselection=1`, # copy selected text when selection appeared in combobox
- `height=24`, # height of value list
- `justify="left"`, # justify combobox text left right or center
- `postcommand=""`, # combobox postcommand when selected item
- `style="TCombobox"`, # combobox style object
- `takefocus=1`, # set take focus to 0 if you don't want to focusing effect
- `textvariable=False`, # combobox text variable, to get and set value dynamically
- `validate=None`, # validate
- `validatecommand=False`, # validate command
- `values=['No more item']`, # combo values to set as a list
- `width=24`, # combobox width
- `xscrollcommand=False`, # combobox left right scrolling
- `font=("Calibri", 10)`, # combobox font style
- `row=0`, # row position
- `column=0`, # column position
- `padx=0`, # padding for left and right
- `pady=0`, # padding for top and bottom
- `ipady=2`, # internal padding for top and bottom
- `sticky='w'`, # combobox sticky position w, e, s, n, we, ne, se etc
- `text="-----"`, # default text in combobox widget
- `default=0`, # default text index from value list

an example of creating Combobox widget:

```
from ntk import Tk, Combobox
root = Tk(title='Example of ntk window')
Combobox = Combobox(root)
root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Combobox class.

4.14 Button

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

ntk Button is extended version of tkinter base Button with more functionality and eye touchable design, to use this Button widget we need to import first it from ntk by

```
from ntk import Button
```

and initialize it by calling it

`button1 = Button(root)` # root is master window, your tkinter window or sub window

call it and see the magic, this will create button and basic grid will be applied, you need to pass parameters described below to get your desired button style and grid.

available parameters are:

- `root`, # root is a master window to place this button into it
- `abg="bg-secondary"`, # background color when button is active
- `afg="fg-light"`, # foreground color when button is active
- `anchor="center"`, # anchor text to left right or center
- `bg="bg-info"`, # background color
- `bitmap=None`, # bitmap image to place
- `bd=0`, # border width to square
- `command=None`, # command will execute when button is clicked
- `compound="left"`, # compound image to left right center top or bottom
- `cursor="hand2"`, # mouse cursor style
- `default="normal"`, # default state style
- `dfg="fg-primary"`, # foreground color when button is disabled
- `font=("Calibri", 9)`, # font style
- `fg="fg-white"`, # foreground color
- `height=2`, # height of the button instance
- `hlbg="bg-light"`, # background color when button is highlighted
- `hlc="bg-light"`, # foreground color when button is highlighted
- `hlt=1`, # thickness width when button is highlighted

- `image=None`, # image to set in button
- `justify="center"`, # justify button text left right or center
- `overrelief="groove"`, # over relief
- `padx=3`, # padding in left and right
- `pady=2`, # padding in top and bottom
- `relief="flat"`, # relief
- `rdelay=1000`, # repeat delay
- `rinterval=2000`, # repeat interval
- `state="normal"`, # default state normal disable or active
- `takefocus=1`, # set button property if button can take focus or not
- `text="Button"`, # button text value
- `tvar=None`, # text variable to set and get dynamic value
- `underline=99`, # underline position for text chars
- `width=16`, # button width
- `wraplength=0`, # text wrap length
- `row=0`, # row position
- `column=0`, # column position
- `ipadx=0`, # inner button padding for left and right
- `ipady=1`, # inner button padding for top and bottom
- `hoverbg="bg-warning"`, # background color when button is hovered
- `hoverfg="fg-dark"`, # foreground color when button is hovered

ntk can work with grid manager and it's automated, you just need to pass grid options into Button class grid options by default set's, so if you don't pass them it will be grided in default row-column setting

an example of creating Button:

```
from ntk import Tk, Button

root = Tk(title='Example of ntk window')

button1 = Button(root, text="click to see effect", row=1, column=1)

root.mainloop()
```

you can pass extra arguments and keyword arguments, and those will be passed to tkinter Button class.

CHAPTER

FIVE

UTILS

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

5.1 Admin

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

ntk admin utils are some os level functions which can be used to check or set program as admin mode

```
from ntk.utils import *
```

`admin = is_admin()` # check if your program is running as administrator or normal

run_as_admin() function can be used to open a popup which will take permission to run as administrator and restart your program

```
run_as_admin()
```

it has three optional params

`python_exe=sys.executable` # is can be passed, if you'r using multiple system at once `cmdLine=None` # is can be passed, if you want to run some cmd command by attach it to run as admin call `wait=True` # is can be passed, if want to perform wait term

5.2 Utils

ntk will solve your problem when you can't learn and implement, python tkinter to create desktop application in concern to, good looking and os level implementation.

ntk Utils are some functions which can be used to get some powerfull options in our program

```
from ntk.utils import *
```

```
bg_colors = bg_colors()
```

it will return ntk supported background colors referenced to bootstrap

```
fg_colors = fg_colors()
```

it will return ntk supported background colors referenced to bootstrap

```
white = color("#FFFFFF")
```

it will return absolute color value for given param, by matching query from foreground, background and actual

```
delete_child(master)
```

delete_child function will destroy all child widget from any widget which is passed as a master

exclude param can be used to exclude any single type, like

```
delete_child(master, exclude='label') # it will delete all child widget except label widget
```

just param can be used to delete any single type, like

```
delete_child(master, just='label') # it will delete all child widget which is a label widget
```

another useful utils is w and h, this two utils can be used to get dynamic and responsive width and height for given width or height

```
width = w(500)
```

it will return responsive 500 width, it can be smaller or bigger number with respect to device width

```
height = h(620)
```

it will return responsive 500 height, it can be smaller or bigger number with respect to device height

w and h function also can be imported and used by gv (global var) object, we will talk about gv in the objects.gv page

CHAPTER

SIX

HISTORY

6.1 1.0.0 (2020-08-10)

- First pre-release

6.2 1.1.0 (2020-09-01)

- Alpha release

6.3 1.2.0 (2020-09-14)

- Better argument docs

6.4 1.3.0 (2020-09-27)

- Better argument docs
- Advance functionality

6.5 2.0.0 (2020-11-16)

- Available in pypi
- Advance functionality

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search