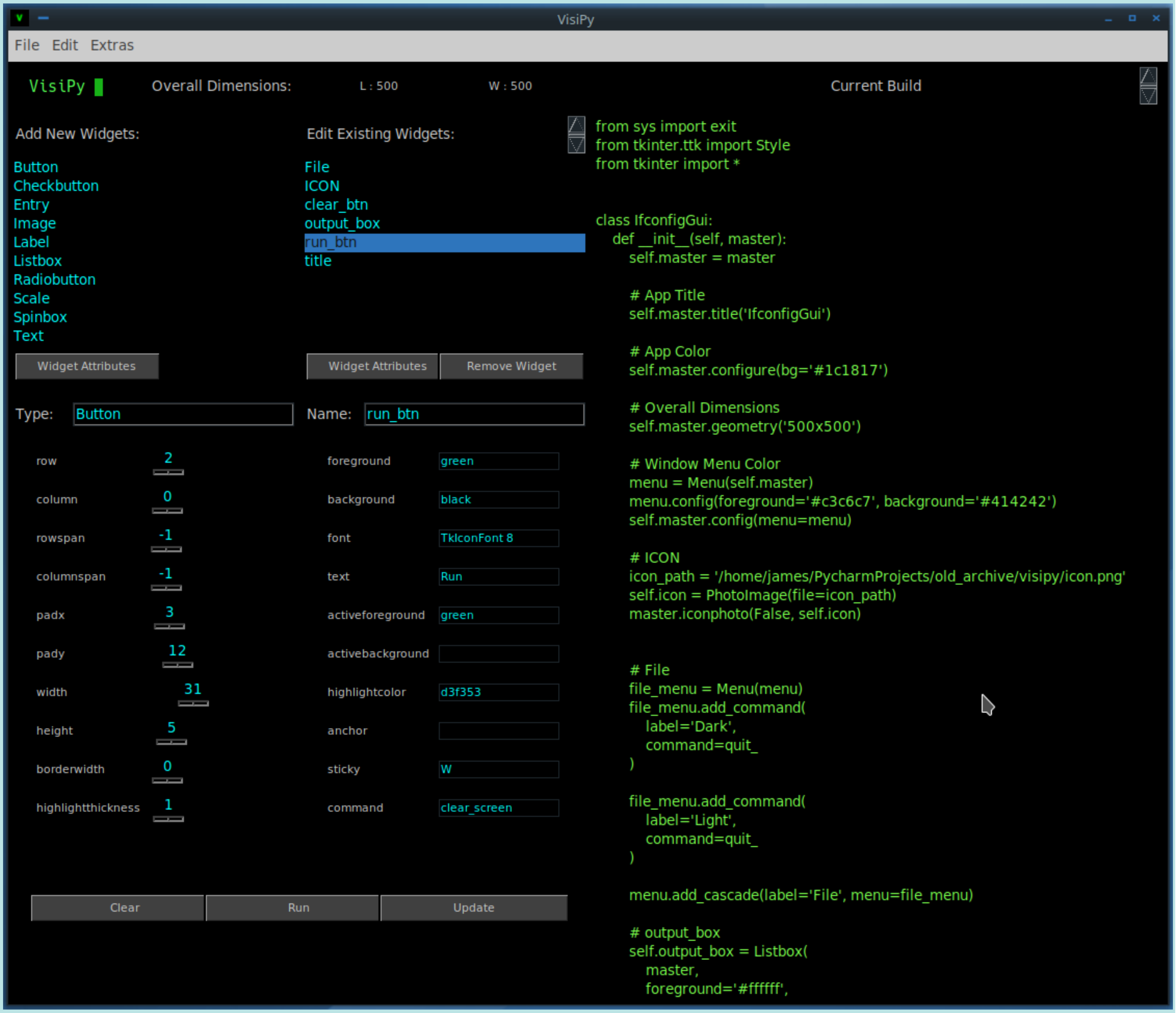
VisiPy Tutorial



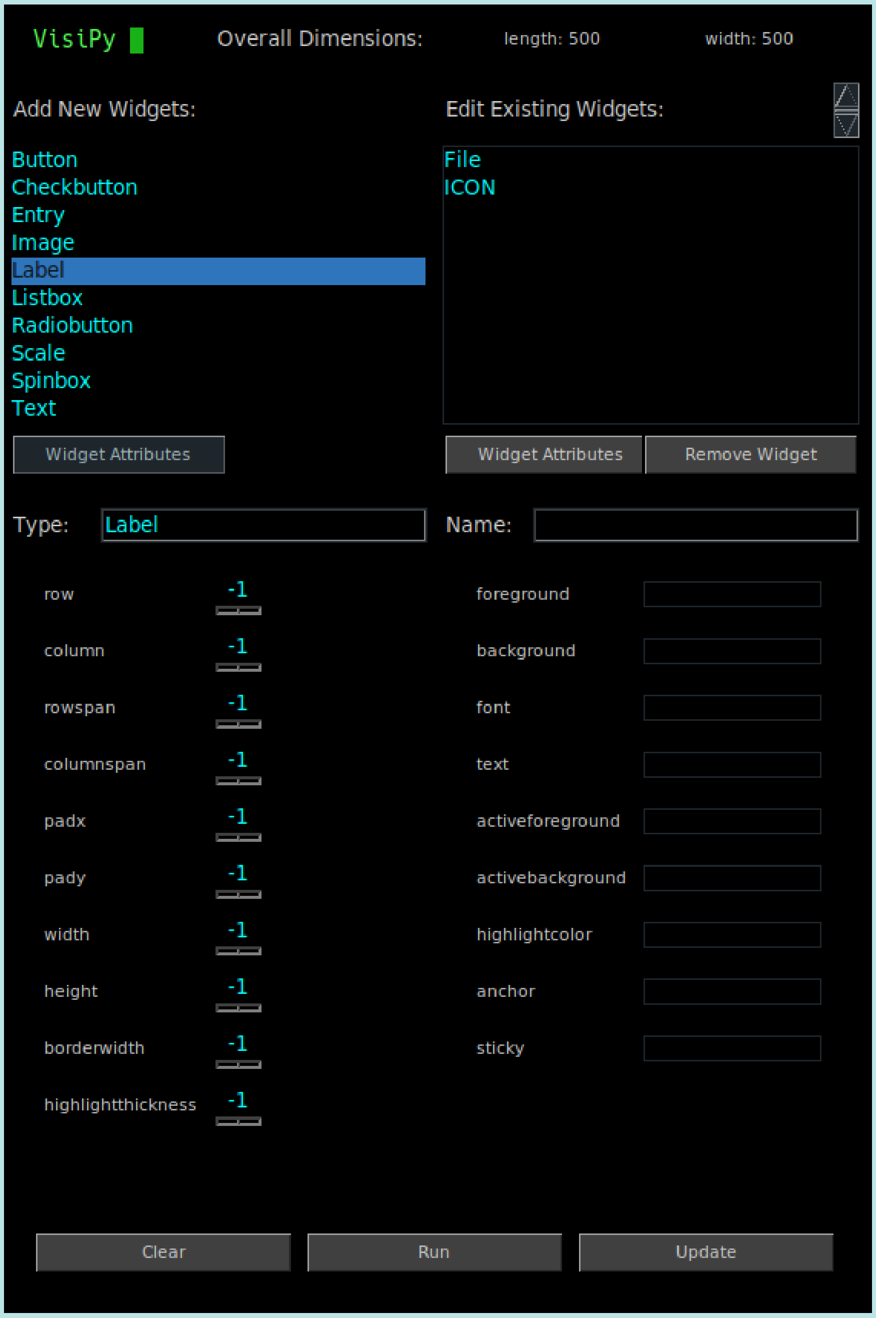
Type: Button

Name: run_btn

row	2	foreground	green
column	0	background	black
rowspan	-1	font	TkIconFont 8
columnspan	-1	text	Run
padx	3	activeforeground	green
pady	12	activebackground	
width	31	highlightcolor	d3f353
height	5	anchor	
borderwidth	0	sticky	W
highlightthickness	1	command	clear_screen

```
from sys import exit
from tkinter.ttk import Style
from tkinter import *


class IfconfigGui:
    def __init__(self, master):
        self.master = master

        # App Title
        self.master.title('IfconfigGui')

        # App Color
        self.master.configure(bg='#1c1817')

        # Overall Dimensions
        self.master.geometry('500x500')

        # Window Menu Color
        menu = Menu(self.master)
        menu.config(foreground='#c3c6c7', background='#414242')
        self.master.config(menu=menu)

        # ICON
        icon_path = '/home/james/PycharmProjects/old_archive/visipy/icon.png'
        self.icon = PhotoImage(file=icon_path)
        master.iconphoto(False, self.icon)


        # File
        file_menu = Menu(menu)
        file_menu.add_command(
            label='Dark',
            command=quit_
        )

        file_menu.add_command(
            label='Light',
            command=quit_
        )

        menu.add_cascade(label='File', menu=file_menu)

        # output_box
        self.output_box = Listbox(
            master,
            foreground='#ffffff',
```

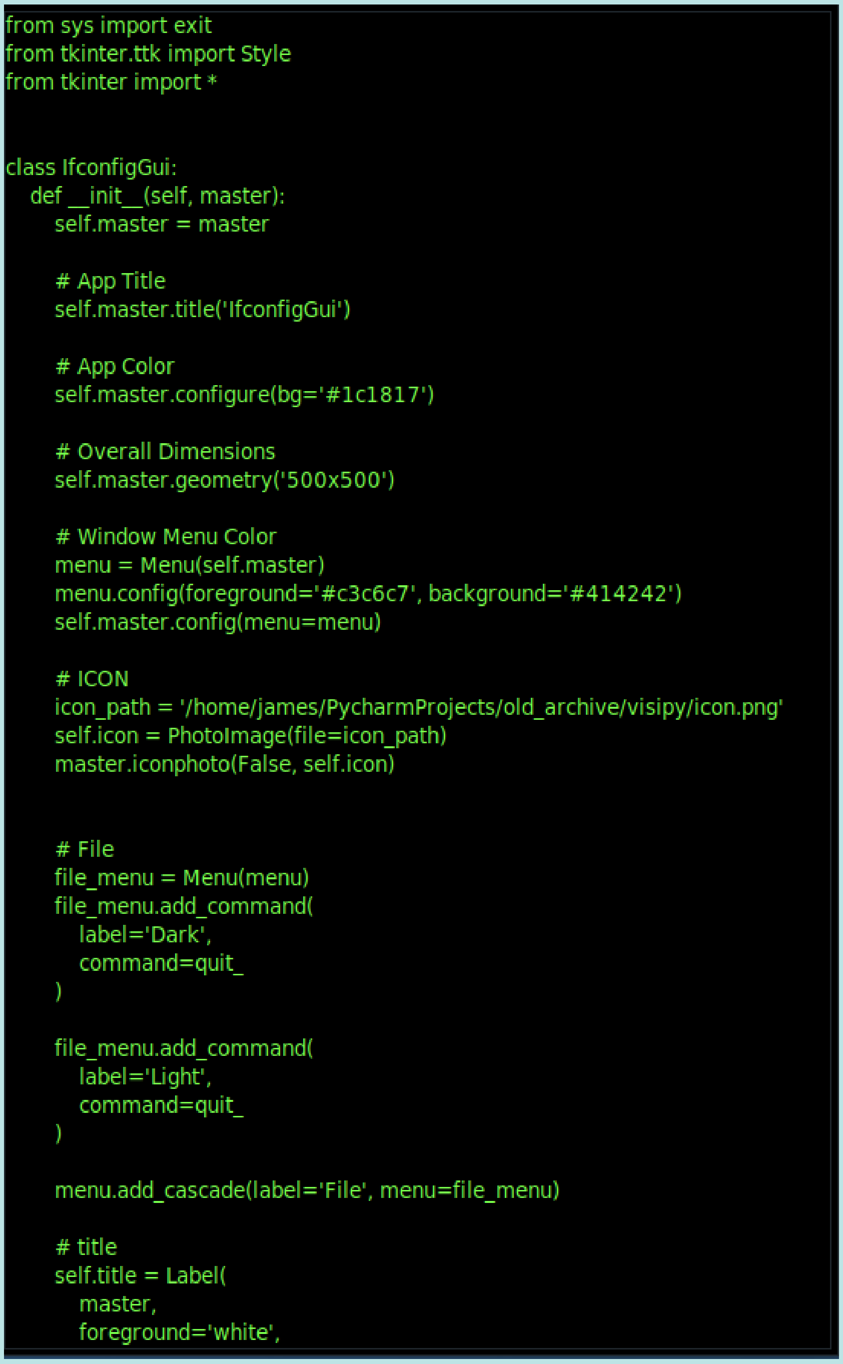
Directly under the *Add New Widgets* label there is a listbox containing all default widgets.
Any of these may be selected for editing by selecting the widget, and pressing *Widget Attributes* (left).
The selected widget's available attributes will appear in the area below.
Attributes vary per widget. To add a new attribute, *adjust a slider above -1 or enter a value in a text box.*
Fields that are empty or contain -1 will be ignored when *Update* is pressed.



Type: Label
Name:

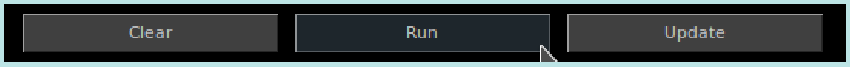
row	-1	foreground	
column	-1	background	
rowspan	-1	font	
columnspan	-1	text	
padx	-1	activeforeground	
pady	-1	activebackground	
width	-1	highlightcolor	
height	-1	anchor	
borderwidth	-1	sticky	
highlightthickness	-1		

The current code build (right side of VisiPy) will be updated each time the *Update* button is pressed. This is the main build window that will change when a widget is created, updated, or removed:

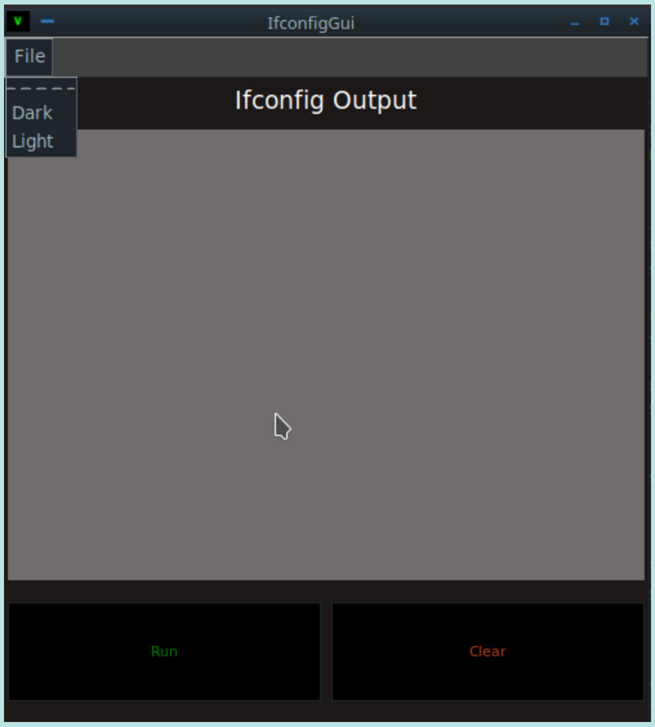


```
from sys import exit
from tkinter.ttk import Style
from tkinter import *


class IfconfigGui:
    def __init__(self, master):
        self.master = master

        # App Title
        self.master.title('IfconfigGui')

        # App Color
        self.master.configure(bg='#1c1817')

        # Overall Dimensions
        self.master.geometry('500x500')

        # Window Menu Color
        menu = Menu(self.master)
        menu.config(foreground='#c3c6c7', background='#414242')
        self.master.config(menu=menu)

        # ICON
        icon_path = '/home/james/PycharmProjects/old_archive/visipy/icon.png'
        self.icon = PhotoImage(file=icon_path)
        master.iconphoto(False, self.icon)


        # File
        file_menu = Menu(menu)
        file_menu.add_command(
            label='Dark',
            command=quit_
        )

        file_menu.add_command(
            label='Light',
            command=quit_
        )

        menu.add_cascade(label='File', menu=file_menu)

        # title
        self.title = Label(
            master,
            foreground='white',
```

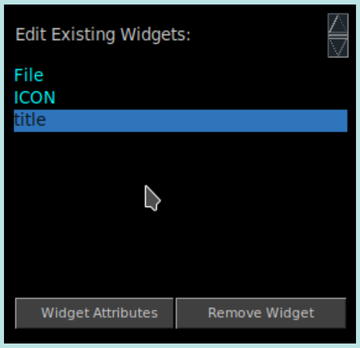
Any time that the *Run* button is pressed, the current build/code-base will be executed.
The *Update* button should be pressed to update the current build after selecting a widget (under *Add New Widgets or Edit New Widgets)* and then filling out all of the associated widget attributes. The current code base will update in the build window and reflect the new changes.



Pressing *Run* will run the current code build, and the current state of the GUI can be examined. (Example app to be built in this tutorial after pressing *Run*):

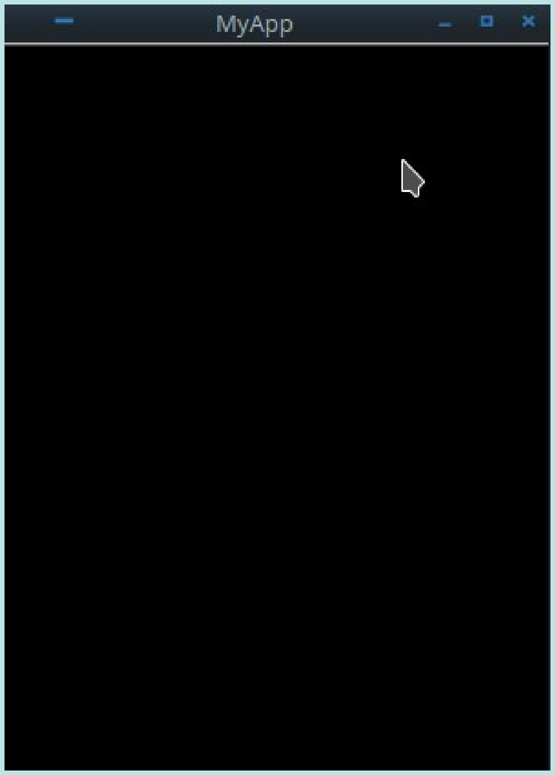


The listbox under *Edit Existing Widgets* contains widgets that have been added as well as the items added from the *Extras* window menu. Selecting an existing widget under *Edit Existing Widgets* and pressing "Widget Attributes" will load the highlighted widget's details for editing.

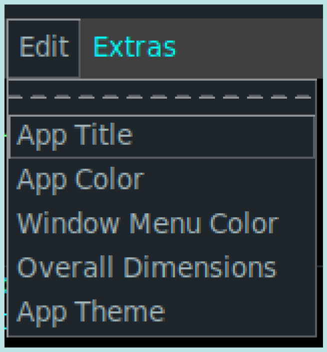


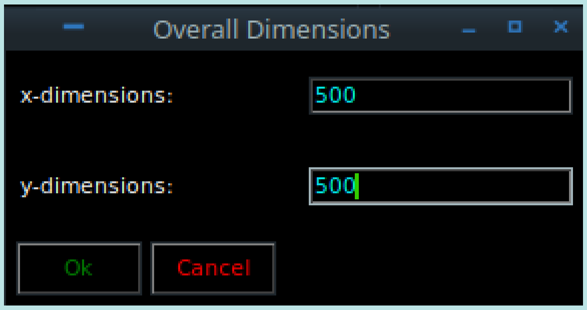
Building an application:

An example app that displays IP information will be built.
When Visipy first opens, try pressing the *Run* button.
The default application should appear:



Select the *Edit* Menu, and then select *Overall Dimensions* (enter 500x500):

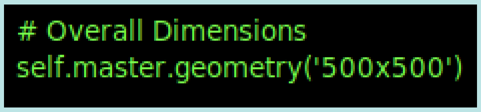




x-dimensions: 500

y-dimensions: 500

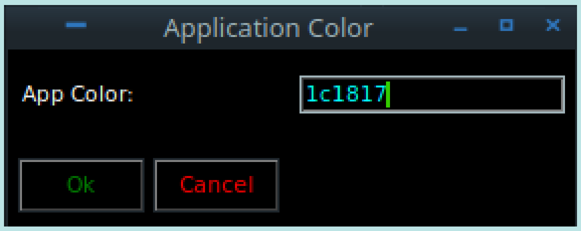
The build window should update as soon as *Ok* is pressed. Note the change made inside of the build box:



```
# Overall Dimensions
self.master.geometry('500x500')
```

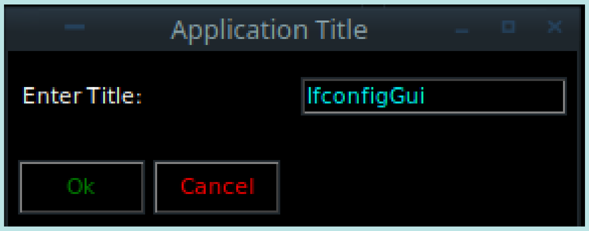
Note: The *Update* button does not need to be pressed for any of the entries in the *Edit* or *Extras* menus. It only needs to be pressed when finished filling out and/or editing a widget in the *Add New Widgets* listbox or the *Edit Existing Widgets* listbox.

Next set the *App Color* and *App Title*, also found in the *Edit* menu:



Application Color

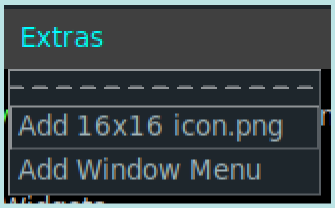
App Color: 1c1817

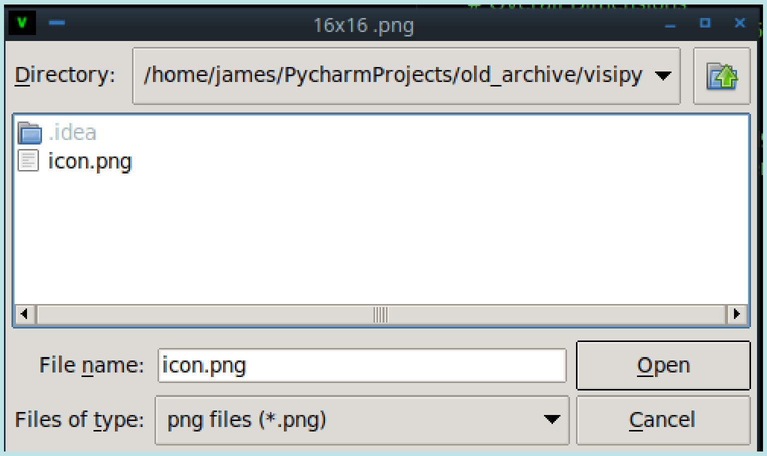


Application Title

Enter Title: IfconfigGui

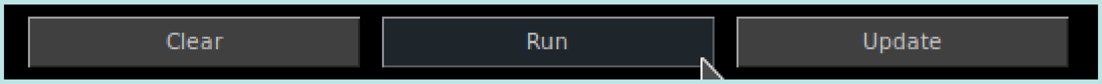
Select *Add 16x16 icon* in the Extras menu.
For demonstration purposes, the same icon used by Visipy will be loaded.
Note: The icon file **does not** have to be in your project directory.



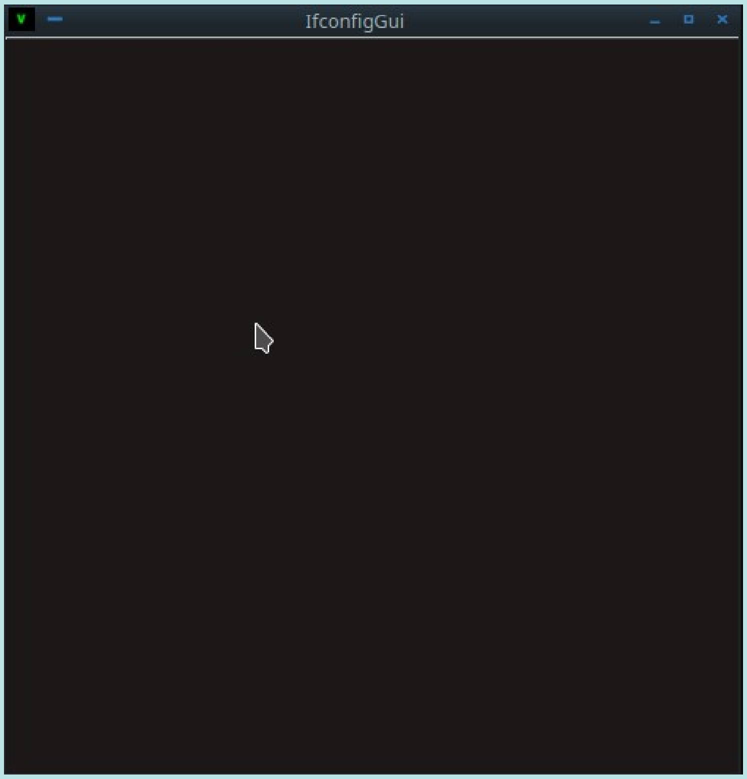

16x16 .png
Directory: /home/james/PycharmProjects/old_archive/visipy
.idea
icon.png
File name: icon.png
Open
Files of type: png files (*.png)
Cancel

If you do not have a 16x16 icon.png (Linux/Mac) or a 16x16 icon.ico (Windows) skip this step for now.

Press the *Run* button if you haven't already done so:

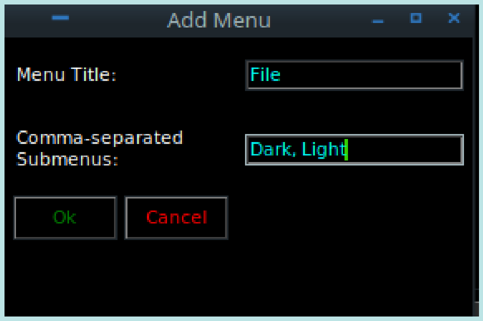


The current build/GUI should open:



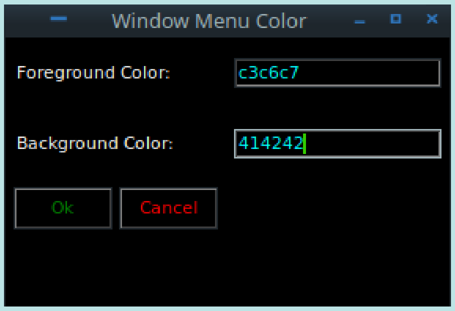
Now add a menu to the window menu bar by selecting *Extras* and then *Add Window Menu*:

The values inside of *Comma-separated Submenus* should be a comma separated string of submenu names. Notice how ICON and File menu additions from the *Extras* menu appear in *Edit Existing Widgets a*fter pressing the Ok button. The code build will also show new entries for the menu and submenus.


Menu Title: File

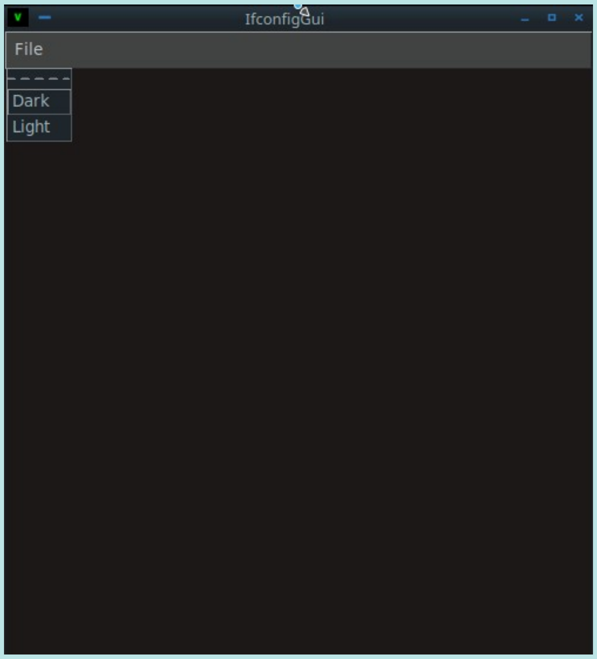
Comma-separated Submenus: Dark, Light

Add some color to the window menu bar and the menu text. Select *Edit* and then *Window Menu Color*:

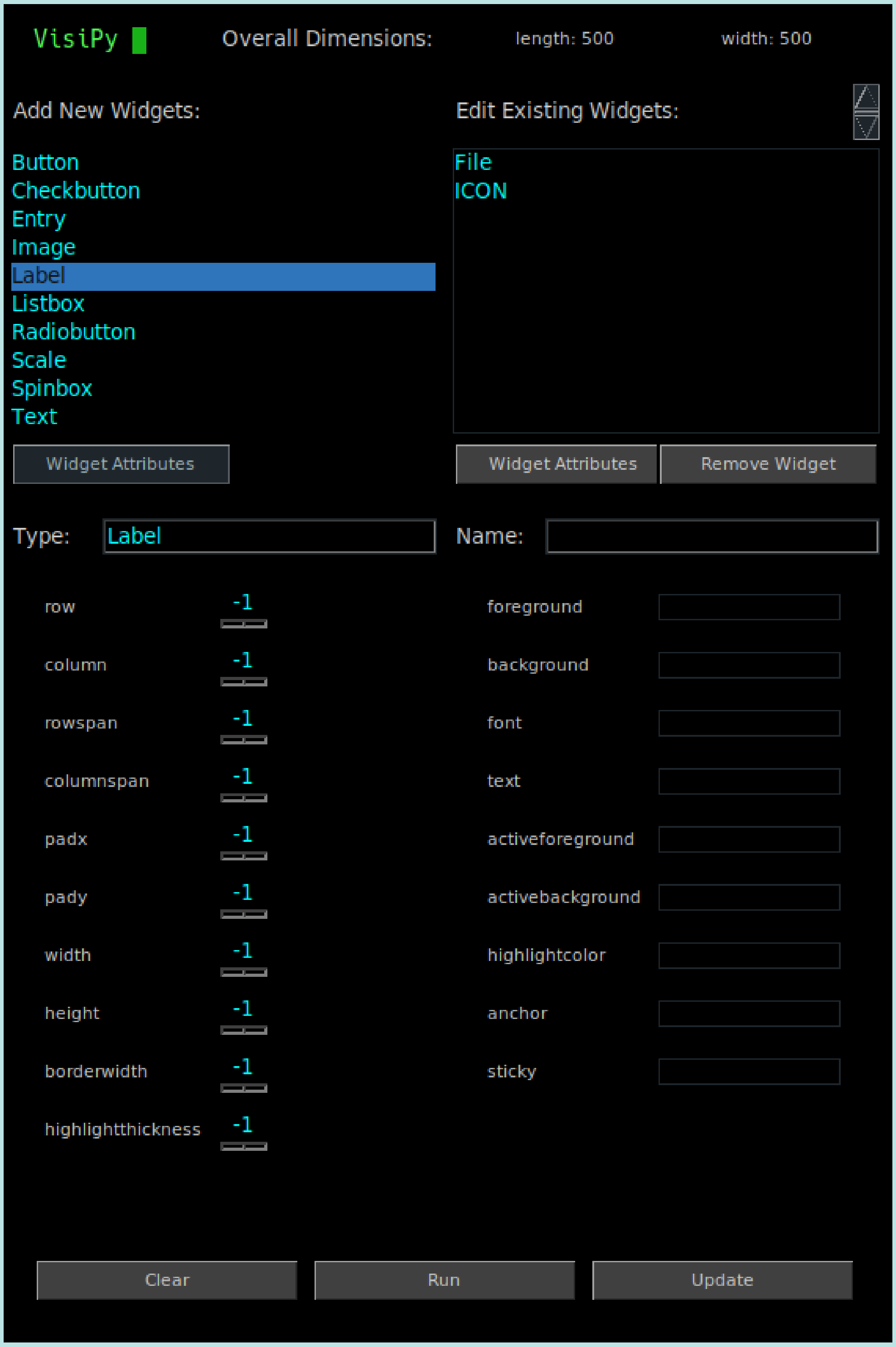

Foreground Color: c3c6c7

Background Color: 414242

Pressing Run should show the new additions:



Now lets add some real widgets.


VisiPy

Overall Dimensions: length: 500 width: 500

Add New Widgets:

Button
Checkbutton
Entry
Image
Label
Listbox
Radiobutton
Scale
Spinbox
Text

Edit Existing Widgets:

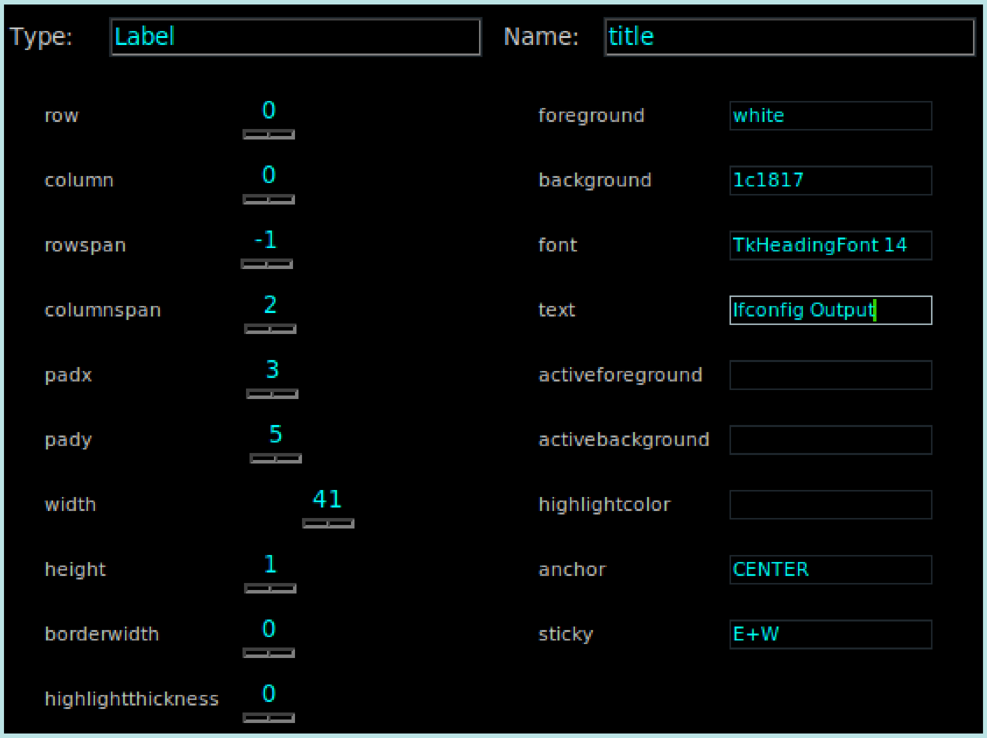
File
ICON

Type: Label

Name:

row	-1	foreground	
column	-1	background	
rowspan	-1	font	
columnspan	-1	text	
padx	-1	activeforeground	
pady	-1	activebackground	
width	-1	highlightcolor	
height	-1	anchor	
borderwidth	-1	sticky	
highlightthickness	-1		

Select the *Label* widget under *Add New Widgets* and then press the *Widget Attributes* button (left). Make the following adjustments:

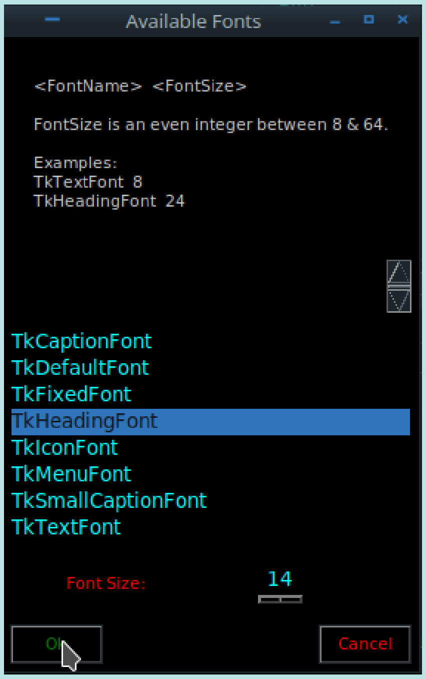


Type: Label
Name: title

row	0	foreground	white
column	0	background	1c1817
rowspan	-1	font	TkHeadingFont 14
columnspan	2	text	Ifconfig Output
padx	3	activeforeground	
pady	5	activebackground	
width	41	highlightcolor	
height	1	anchor	CENTER
borderwidth	0	sticky	E+W
highlightthickness	0		

Note: Clicking *File* and *Available Fonts* will display a popup window and autofill the *font* field.

Note: Visipy only allows the system fonts when working.
After templating you may change the fonts to any style.


Available Fonts

<FontName> <FontSize>

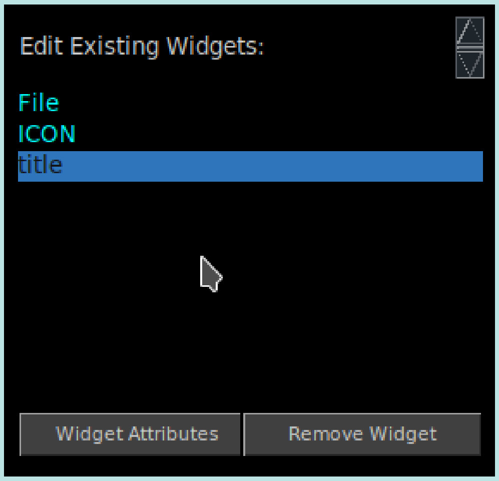
FontSize is an even integer between 8 & 64.

Examples:
TkTextFont 8
TkHeadingFont 24

TkCaptionFont
TkDefaultFont
TkFixedFont
TkHeadingFont
TkIconFont
TkMenuFont
TkSmallCaptionFont
TkTextFont

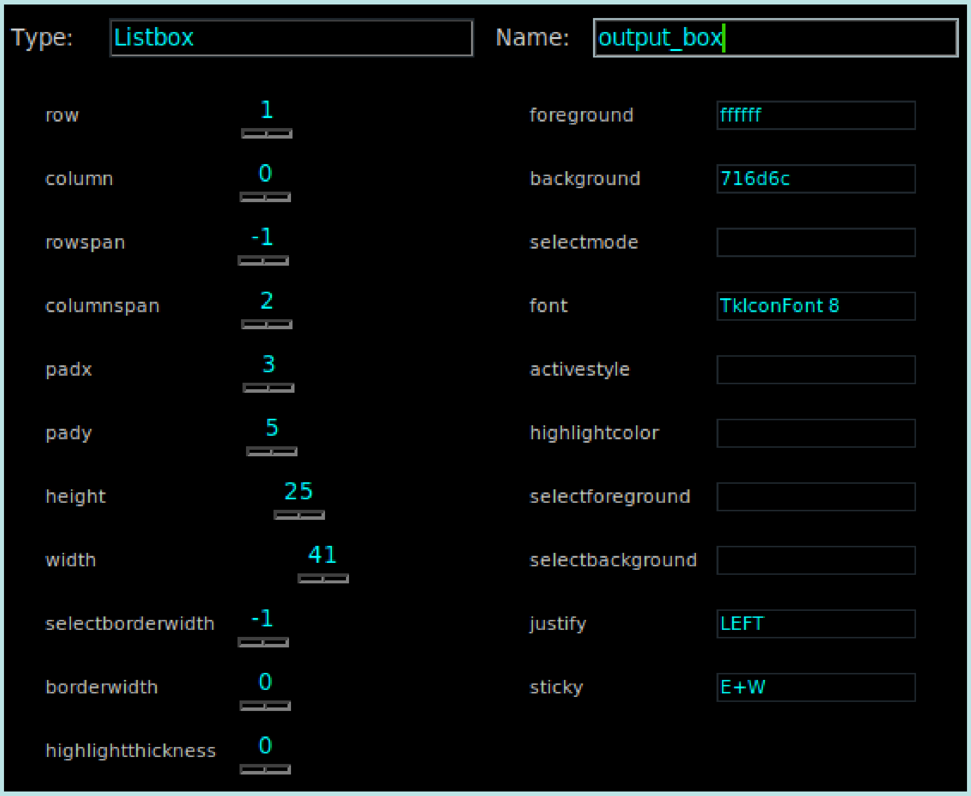
Font Size: 14

After pressing the *Update* button, the code build window will update, and a new widget will appear in the existing widgets listbox reflecting the chosen settings. Press the *Run* button if desired to check the GUI.


Edit Existing Widgets:

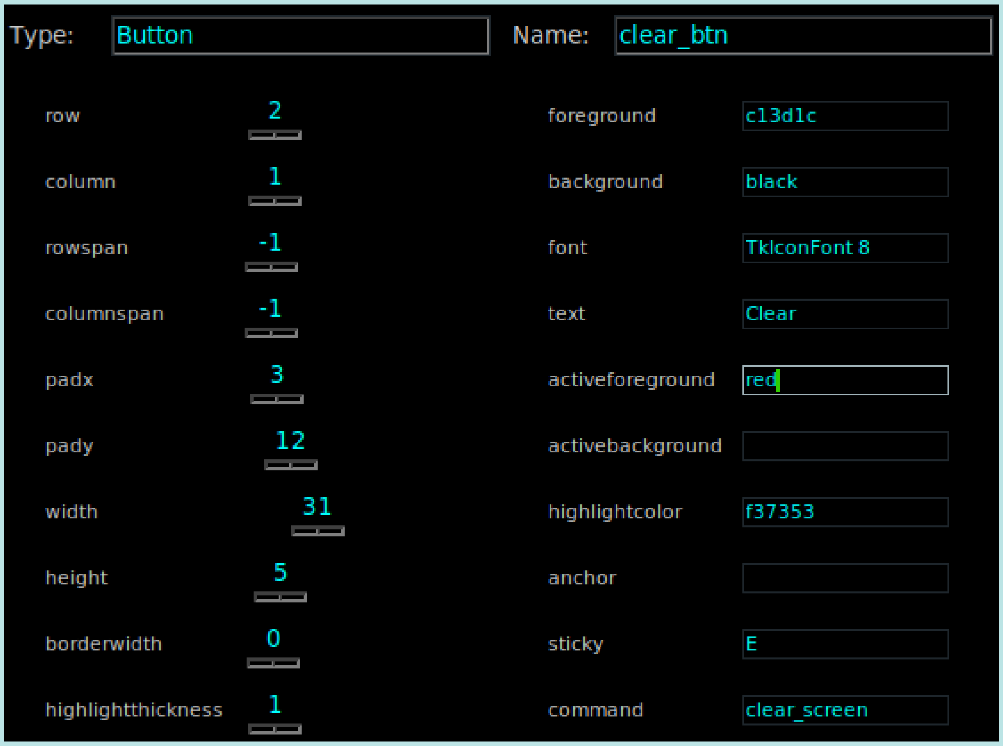
File
ICON
title

Now add a *Listbox* with the following attributes in the same manner:



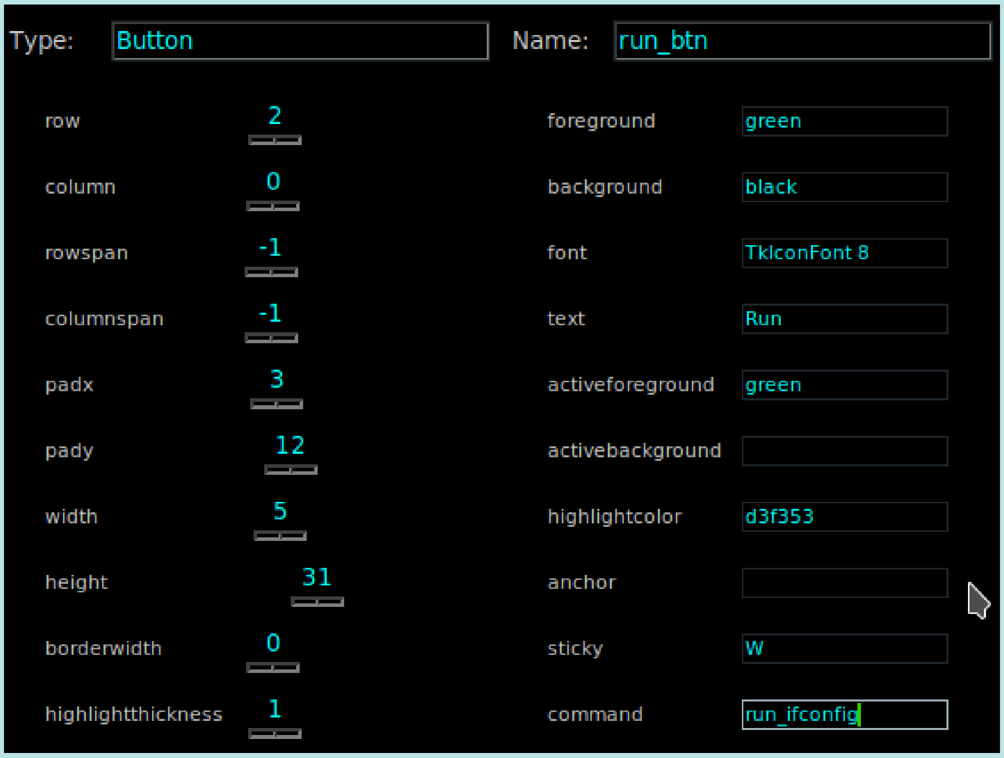
Type:	Listbox	Name:	output_box
row	1	foreground	ffffff
column	0	background	716d6c
rowspan	-1	selectmode	
columnspan	2	font	TkIconFont 8
padx	3	activestyle	
pady	5	highlightcolor	
height	25	selectforeground	
width	41	selectbackground	
selectborderwidth	-1	justify	LEFT
borderwidth	0	sticky	E+W
highlightthickness	0		

Now add a Clear button and a Run button:



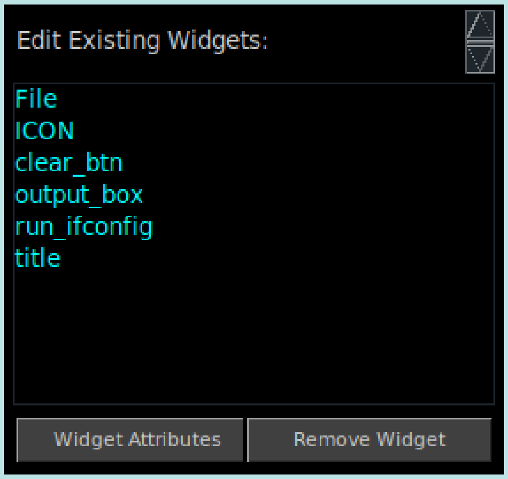
Type:	Button		Name:	clear_btn	

row	2	foreground	c13d1c
column	1	background	black
rowspan	-1	font	TkIconFont 8
columnspan	-1	text	Clear
padx	3	activeforeground	red
pady	12	activebackground	
width	31	highlightcolor	f37353
height	5	anchor	
borderwidth	0	sticky	E
highlightthickness	1	command	clear_screen



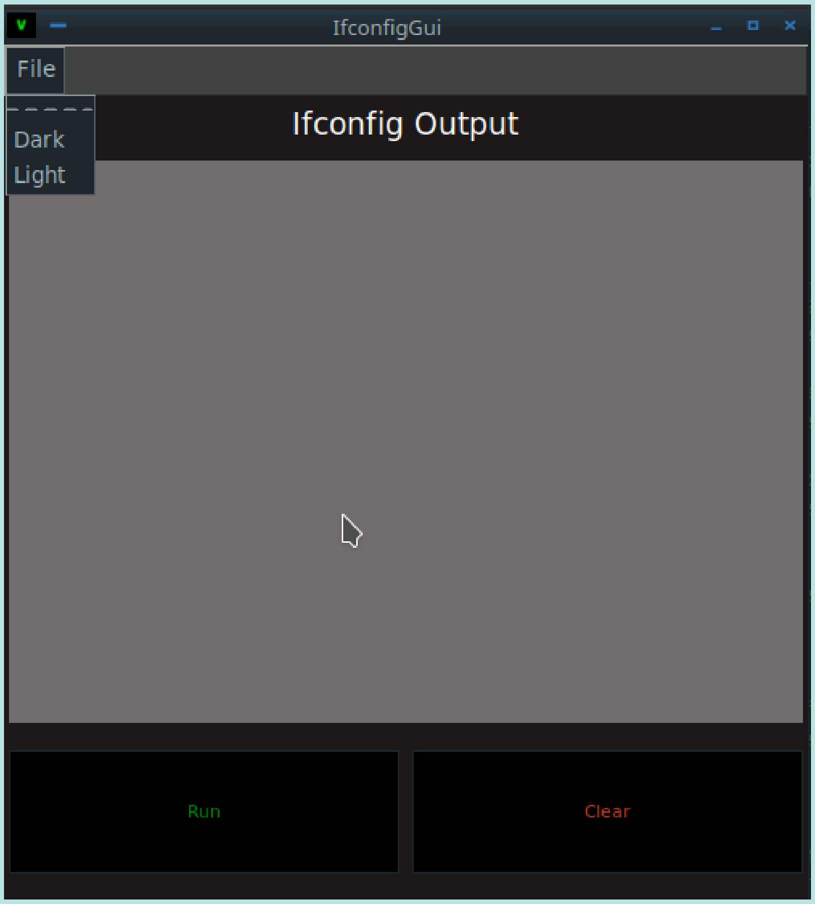
Type:	Button	Name:	run_btn
row	2	foreground	green
column	0	background	black
rowspan	-1	font	TkIconFont 8
columnspan	-1	text	Run
padx	3	activeforeground	green
pady	12	activebackground	
width	5	highlightcolor	d3f353
height	31	anchor	
borderwidth	0	sticky	W
highlightthickness	1	command	run_ifconfig

Once finished, the *Edit Existing Widgets* listbox should have the following contents:

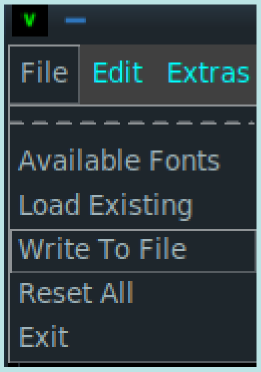

Edit Existing Widgets:

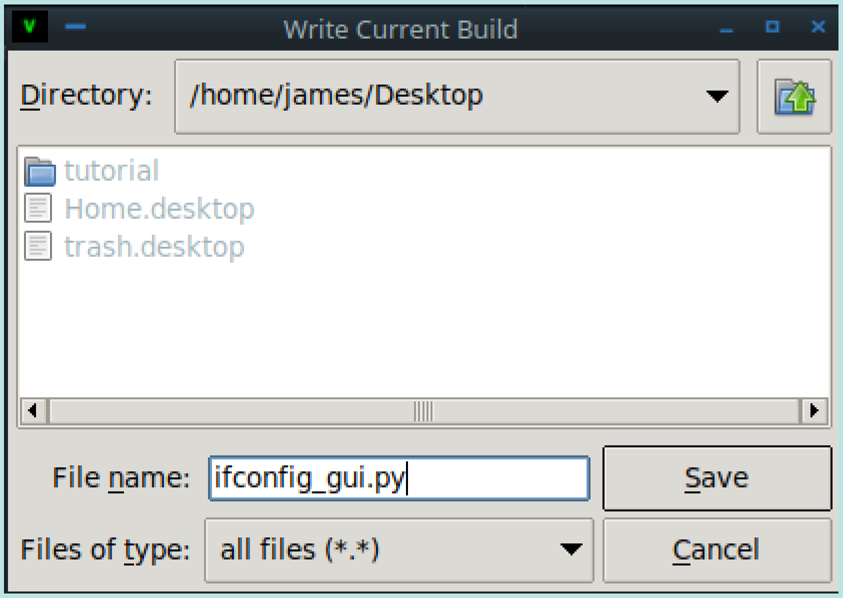
File
ICON
clear_btn
output_box
run_ifconfig
title

The app should build as shown:



Your current build can be written to a *.py* file at any time.
Simply select *File* and *Write To File.*





Directory: /home/james/Desktop
File name: ifconfig_gui.py
Files of type: all files (*.*)

A *.py* file will be created along with a *.project* file. The project file will contain valid JSON, and represents the current state of your app whenever *Write To File* was used.

The *project* file may be discarded or used to load your project at a later date.

To load a an existing project, select *File* and *Load Existing.* Provide a path to a *.project* JSON file.

The final code should resemble the below code once written to a file.
It’s up to you the developer to replace the *command* placeholders with your own methods.

Note: The quit_() function is placed just below the class by default.

Any window bar menus’ *command* attributes will be linked to the quit_() function by default (to be replaced by the developer after templating).

Any widgets other than the window bar menus that include a *command* attribute, will also have their corresponding methods templated to a placeholder function signifying TODO/action handling (also to be replaced by the developer after templating).

Here is the example code after writing the Ifconfig project to a file:

```
from sys import platform
from tkinter.ttk import Style
from tkinter import *


class IfconfigGui:
    def __init__(self, master):
        self.master = master

        # App Title
        self.master.title('IfconfigGui')

        # App Color
        self.master.configure(bg='#1c1817')

        # Overall Dimensions
        self.master.geometry('500x500')

        # Window Menu Color
        menu = Menu(self.master)
        menu.config(foreground='#c3c6c7', background='#414242')
        self.master.config(menu=menu)

        # ICON
        icon_path = '/home/james/PycharmProjects/old_archive/visipy/icon.png'
        self.icon = PhotoImage(file=icon_path)
        master.iconphoto(False, self.icon)

        # File
        file_menu = Menu(menu)
        file_menu.add_command(
            label='Dark',
            command=quit_
        )
```

```
file_menu.add_command(
    label='Light',
    command=quit_
)

menu.add_cascade(label='File', menu=file_menu)

# title
self.title = Label(
    master,
    foreground='white',
    background='#1c1817',
    font='TkHeadingFont 14',
    text='Ifconfig Output',
    anchor=CENTER,
    width=41,
    height=1,
    borderwidth=0,
    highlightthickness=0
)

self.title.grid(
    row=0,
    column=0,
    columnspan=2,
    padx=3,
    pady=5,
    sticky=E+W
)

# output_box
self.output_box = Listbox(
    master,
    foreground='#ffffff',
    background='#716d6c',
    font='TkIconFont 8',
    justify=LEFT,
    width=41,
    height=25,
    borderwidth=0,
    highlightthickness=0
)

self.output_box.grid(
    row=1,
    column=0,
    columnspan=2,
    padx=3,
    pady=5,
    sticky=E+W
)
```

```
        # clear_btn
        self.clear_btn = Button(
            master,
            foreground='#c13d1c',
            background='black',
            font='TkIconFont 8',
            text='Clear',
            activeforeground='red',
            command=self.clear_screen,
            highlightcolor='#f37353',
            width=31,
            height=5,
            borderwidth=0,
            highlightthickness=1
        )

        self.clear_btn.grid(
            row=2,
            column=1,
            padx=3,
            pady=12,
            sticky=E
        )

        # run_btn
        self.run_btn = Button(
            master,
            foreground='green',
            background='black',
            font='TkIconFont 8',
            text='Run',
            activeforeground='green',
            command=self.run_ifconfig,
            highlightcolor='#d3f353',
            width=31,
            height=5,
            borderwidth=0,
            highlightthickness=1
        )

        self.run_btn.grid(
            row=2,
            column=0,
            padx=3,
            pady=12,
            sticky=W
        )

    def run_ifconfig(self):
        """ TODO: Add handling code here """
        print('Handle ifconfig here')

    def clear_screen(self):
        """ TODO: Add handling code here """
        print('Handle clear_screen here')


def quit_():
    exit()
```

```
# App Theme
def run_gui():
    root = Tk()
    root.style = Style()
    root.style.theme_use('default')
    IfconfigGui(root)
    root.mainloop()


if __name__ == '__main__':
    run_gui()
```

Now that the code is templated, the project can be finished in any editor.

The below code is just an example of capturing stdout from ifconfig and loading into a listbox:

(additions are highlighted)

```
from subprocess import Popen, PIPE
from sys import platform
from tkinter.ttk import Style
from tkinter import *


class IfconfigGui:
    def __init__(self, master):
        self.master = master

        # App Title
        self.master.title('IfconfigGui')

        # App Color
        self.master.configure(bg='#1c1817')

        # Overall Dimensions
        self.master.geometry('500x500')

        # Window Menu Color
        menu = Menu(self.master)
        menu.config(foreground='#c3c6c7', background='#414242')
        self.master.config(menu=menu)

        # ICON
        icon_path = '/home/james/PycharmProjects/old_archive/visipy/icon.png'
        self.icon = PhotoImage(file=icon_path)
        master.iconphoto(False, self.icon)

        # File
        file_menu = Menu(menu)
        file_menu.add_command(
            label='Dark',
            command=self.change_dark
        )

        file_menu.add_command(
            label='Light',
            command=self.change_light
        )
```

```
menu.add_cascade(label='File', menu=file_menu)

# clear_btn
self.clear_btn = Button(
    master,
    foreground='#c13d1c',
    background='black',
    font='TkIconFont 8',
    text='Clear',
    activeforeground='red',
    command=self.clear_screen,
    highlightcolor='#f37353',
    width=31,
    height=5,
    borderwidth=0,
    highlightthickness=1
)

self.clear_btn.grid(
    row=2,
    column=1,
    padx=3,
    pady=12,
    sticky=E
)

# run_btn
self.run_btn = Button(
    master,
    foreground='green',
    background='black',
    font='TkIconFont 8',
    text='Run',
    activeforeground='green',
    highlightcolor='#d3f353',
    width=31,
    height=5,
    borderwidth=0,
    highlightthickness=1,
    command=self.run_ifconfig
)

self.run_btn.grid(
    row=2,
    column=0,
    padx=3,
    pady=12,
    sticky=W
)

# title
self.title = Label(
    master,
    foreground='white',
    background='#1c1817',
    font='TkHeadingFont 14',
    text='Ifconfig Output',
    anchor=CENTER,
    width=41,
    height=1,
    borderwidth=0,
    highlightthickness=0
)
```

```
        self.title.grid(
            row=0,
            column=0,
            columnspan=2,
            padx=3,
            pady=5,
            sticky=E+W
        )

        # output_box
        self.output_box = Listbox(
            master,
            foreground='#ffffff',
            background='#716d6c',
            font='TkIconFont 8',
            justify=LEFT,
            width=41,
            height=25,
            borderwidth=0,
            highlightthickness=0
        )

        self.output_box.grid(
            row=1,
            column=0,
            columnspan=2,
            padx=3,
            pady=5,
            sticky=E+W
        )

    def run_ifconfig(self):
        self.output_box.delete(0, 'end')
        try:
            out = Popen(['ifconfig'], stdout=PIPE, stderr=PIPE).communicate()
            stdout = out[0].strip().decode('utf-8').split('\n')
            self.output_box.insert('end', *stdout)
        except Exception as err:
            self.output_box.insert('end', *['ERROR', str(err)])

    def clear_screen(self):
        self.output_box.delete(0, 'end')


    def change_dark(self):
        self.master.configure(
            background='#1c1817'
        )
        self.title.configure(
            background='#1c1817',
            foreground='white'
        )
        self.output_box.configure(
            background='#716d6c',
            foreground='#ffffff'
        )
        self.clear_btn.config(
            background='black'
        )
        self.run_btn.config(
            background='black'
        )
    def change_light(self):
        self.master.configure(
            background='#c7cbcd'
        )
        self.title.configure(
            background='#d4dcdf',
            foreground='black'
        )
        self.output_box.configure(
            background='#edf0f1',
            foreground='black'
        )
        self.clear_btn.config(
            background='#ececec'
        )
        self.run_btn.config(
            background='#ececec'
        )


def quit_():
    exit()


# App Theme
def run_gui():
    root = Tk()
    root.style = Style()
    root.style.theme_use('default')
    IfconfigGui(root)
    root.mainloop()


if __name__ == '__main__':
    run_gui()
```