Modern Tkinter – reference card

Pack geometry	.pack(side=TOP, anchor=W, fill=X, expand=YES, padx=5, pady=5) **side**=LEFT,TOP,RIGHT,BOTTOM; **anchor**=NEWS or CENTER; **fill**=X, Y, BOTH, NONE; **expand**=YES, NO **ipadx**, **ipady** – internal padding; **padx**, **pady** – external padding
Grid	.grid(row=1, column=2, rowspan=2, columnspan=2, sticky=NEWS or CENTER, padx=5, pady=5)
Place	.place(x=5, y=5, relwidth=1, relheight=1, width=-10, height=-10)

import tkinter as tk

Canvas	Canvas(**parent**) id = canvas.create_rectangle((10, 10, 30, 30), fill="red", tags=(“palette”, “palettered”)) canvas.tag_bind(id, "<Button-1>", lambda x: setColor("red")) canvas.itemconfigure('palette', width=5) if self.canvas.find_withtag(tk.CURRENT): self.canvas.itemconfig(tk.CURRENT, fill="blue")
Listbox	ListBox(**parent, bg=”#1C3D7D”, fg=”#A0B9E9”, selectmode=EXTENDED**) mylist.insert(0, "First Item"); mylist.insert(END, "Last Item"); mylist.curselection()[0] # returns first selected item
PhotoImage	photoimage = PhotoImage(**file=”images/openfile.gif”**) # has to be GIF or PNG
ScrolledText	scrolledtext.ScrolledText(**parent**)
Spinbox	Spinbox(**parent, from_=1, to=10, width=5, textvariable=varint**)
Text	Text(**parent, background=”#101010”, foreground="#D6D6D6", borderwidth=18, relief=”sunken”, width=16, height=5**)
tkMessageBox	showwarning("Beware", "You are warned"); showinfo("FYI", "This is FYI", icon='question'); showerror("Err..", "its leaking."); askquestion("?", "Can you read this ?"); askokcancel("OK", "Quit Postponing ?"); askyesno("Yes or No", " What Say ?"); askretrycancel("Retry", "Load Failed")
tkFileDialog	askopenfile, askopenfilename, asksaveasfile, asksaveasfilename, askdirectory
Toplevel	Toplevel(**parent**) # main class for windows and dialogs

from tkinter import ttk

Button	Button(**parent, text=”Search”, image=photoimage, compound=tk.LEFT, command=func**) btn.image = photoimage # for some reason this is required to set image
Checkbutton	Checkbutton(**parent, text=”Remember me”, variable=varint, onvalue=3, offvalue=44**)
Combobox	Combobox(**parent, values=[“one”, “two”], state=”readonly”**) cm.current(newindex=None); cm.get(); cm.set(value)
Entry	Entry(**parent, width=30, textvariable=varstr**); entry.set(“value”); var = entry.get()
Frame	Frame(**parent, height=25, bg=”light sea green”)**); # use it to hold widgets for toolbars, sidebars…
Label	Label(**parent, text=”I am a label widget”**); label["text"] = “reset text”
LabelFrame	LabelFrame(**text=”some label”, height=200, width=200**); # used in dialogs to group items
Menubutton	Menubutton(**parent, text=”some text”, menu=mymenu**)
Notebook	Notebook(**parent**); note.add(child, text=”tab 1”, state=”normal”)
OptionMenu	OptionMenu(**parent, var, “Select Country”, “USA”, “UK”, “India”**) # similar to readonly Combobox
PanedWindow	PanedWindow(**master, orient=HORIZONTAL, sashwidth=8**); paned.add(child, width=300)
Progressbar	Progressbar(**parent**); prog.step(amount=None); prog.start(interval=None); prog.stop()
Radiobutton	Radiobutton(**parent, text=”one”, variable=varint, value=1**)
Scale	Scale(**root, variable=varint, from_=0, to=10**)
Scrollbar	Scrollbar(**parent, orient=VERTICAL, command=mytext.yview**)
Separator	Separator(**parent, orient=HORIZONTAL**)
Sizegrip	Sizegrip(**root**).pack(side=BOTTOM, anchor=E)
Treeview	ttk.Treeview(**parent, columns=(“size”, “modified”)**) tree.column(“size”, width=100, anchor=NEWS or CENTER); tree.heading(“size”, text=”Size”) tree.set(“widgets”, “size”, “12KB”) **# tree.insert() if id = “” then inserts as a root item, if id has value then inserts as child of node id** tree.insert(id, “end”, text=”button”, tags=(“one”, ”simple”), values=(“15KB”, “Yesterday”)) tree.tag_configure(“one”, background=”yellow”) tree.tag_bind(“one”, “<1>”, itemClicked); # the item clicked can be found via tree.focus()

```
root = Tk()          # root is instance of Toplevel class
root.title("title of my program")
root.geometry("142x280+150+200")
root.iconbitmap("mynewicon.ico")              # seems to work only on Windows OS
root.configure(background="#4D4D4D")          # top level styling
root.mainloop()
```

Adding Menubar in the widget

```
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0 )          # File menu
mymenu.add_command(label="Mylabel", accelerator="<F5>", compound=LEFT, image=myimage,
underline=0, command=callback)
viewmenu.add_checkbutton(label="Show Line Number", variable=showln)
viewmenu.add_cascade(label="Themes", menu=themesmenu)
themesmenu.add_radiobutton(label="Default White", variable=theme)
root.config(menu=menubar)           # this line actually displays menu
```

Adding Pop-up menu

```
self.context_menu = Menu(self.root, tearoff=0)
self.context_menu.add_command(label="Play", command=self.identify_track_to_play)
self.context_menu. add_checkbutton(label="checkmark_here", variable=myvar)
def show_context_menuContext_menu(self, event):
   self.context_menu.tk_popup(event.x_root+45, event.y_root+10,0)
```

mystring = StringVar() ticked_yes = BooleanVar() option1 = IntVar() volume = DoubleVar() **myvar.set("Wassup Dude")** # setting value of variable **myvar.get()** # Assessing the value of variable from say a callback	Tie scrollbar to widget: tree = TreeView(root, height=6, width=15) scroll = Scrollbar(root, command=tree.yview) tree.configure(yscroll=scroll.set)

Validation

Works on **Entry, Combobox, Spinbox**	
%P – entered value **%s** - value prior to editing **%S** - text string being inserted/deleted, {} otherwise.	vcmd = (self.master.register(self.validate_email), "**%P**") invcmd = (self.master.register(self.invalid_email), "**%P**") self.emailentry = tk.Entry(self.master, validate = "**focusout**", validatecommand=vcmd, invalidcommand=invcmd)
none - no validation **focus** - combines focusin and focusout **focusin** - validate when the widget receives focus **focusout** - validate when the widget loses focus **key** - validate when the entry is edited **all** - validate called in all the above cases	def invalid_email(self, P): self.errormsg.config(text="Invalid Email Address") self.emailentry.focus_set()
Return **False** if validation fails	def validate_email(self, P): self.errormsg.config(text="") x = re.match(r"[^@]+@[^@]+\.[^@]+", P) return (x != None) # **True/False** valid email/invalid email

Widget Events

An application-level binding: Application-level bindings will let you use the same binding across all windows and widgets of the application, as long as any one window of the application is in focus. The syntax for application-level bindings is:
root.bind_all("<F1>", show_help)

mouse buttons double click mouse movement over widget keyboard events button released drag like motion widget invalidated widget changed size connect to event disconnect from event emit event	**<Button-1>**, <Button-3> <Double-Button-1>, <Double-Button-3> <FocusIn>, <FocusOut>, **<Motion>, <Enter>, <Leave>** <Return>, <Escape>, <A>, <F5>, <Key> , <Shift-Up> <ButtonRelease-1> <B1-Motion> The mouse is moved, with mouse button 1 being held down (use B2 for the middle button, B3 for the right button) **<Expose>** **<Configure>** widget.bind(event) widget.unbind(event) widget.event_generate("<Expose>")
Button, Checkbutton, Radiobutton, menu.add_command	Use **command=func** parameter for default action, read value of the variable attached to the button if it has state.
Combobox	<<ComboboxSelected>> virtual event when the user selects an element from the list of values
Notebook	<<NotebookTabChanged>> virtual event after a new tab is selected
Treeview	<<TreeviewSelect>> Generated whenever the selection changes. <<TreeviewOpen>> Generated just before settings the focus item to open=True. <<TreeviewClose>> Generated just after setting the focus item to open=False. .focus() and .selection() methods can be used to determine the affected item or items

TopLevel	root.protocol("WM_DELETE_WINDOW", self.exit_app) # handle closing via "X" button
	root.bind("<Return>", self.ok)
	root.bind("<Escape>", self.cancel)

Widget State

active	The mouse cursor is over the widget and pressing a mouse button will cause some action to occur
disabled	Widget is disabled under program control
focus	Widget has keyboard focus
pressed	Widget is being pressed
selected	"On", "true", or "current" for things like Checkbuttons and radiobuttons
background	Windows and Mac have a notion of an "active" or foreground window. The background state is set for widgets in a background window, and cleared for those in the foreground window
readonly	Widget should not allow user modification
alternate	A widget-specific alternate display format
invalid	The widget's value is invalid

A state is a **sequence of state names**, optionally prefixed with an **exclamation point** indicating that the bit is off.