Day - 27

**Tkinter, args, kwargs and**

**Creating GUI Programs**

Lables, Buttons, Button-click etc, Default arguments: \*Args, \*\*kwargs

**27.1 Tkinter GUI**

* Graphical User Inferface, ti kei inter
* It is Built in GUI in python.
* Window & main-loop:

**import** tkinter

wiNdw = tkinter**.Tk**()

#*Title*

wiNdw**.title**("First GUI ????")

#*size*

wiNdw**.minsize**(width = 500, height = 300)

wiNdw**.mainloop**() #*always in end of the program*

#*python tkinter\_demo\_window.py*

* **label()** and **pack()**:

#*Label() and pack()*

my\_label = tkinter**.Label**(text= "I am a Label", font=("Courier", 24, "bold"))

#*Geometry manager pack() needs to display this label*

my\_label**.pack**()

* Geometry manager ***pack()*** needs to display this label. It defines the layout.

my\_label**.pack**(side= "bottom")

**27.2 Introduction to the Tkinter pack geometry manager:**

So far, you have learned how to use the ***pack()*** method to add widgets to a window.

To arrange widgets on a window, you use geometry managers. The ***pack()*** method is one of three geometry managers in Tkinter. The other geometry managers are ***grid()*** and ***place()***.

The pack geometry manager has many configurations. The following are the most commonly used options: ***fill***, ***expand***, ***side***, ***ipadx***, ***ipady***, ***padx***, and ***pady***.

* Python | ***pack()*** method in ***Tkinter***
* The Pack geometry manager packs widgets in rows or columns. We can use options like fill, expand, and side to control this geometry manager.
* Compared to the ***grid*** ***manager***, the ***pack*** ***manager*** is somewhat limited, but it’s much easier to use in a few, but quite common situations:
* Put a *widget* *inside* a *frame* (or any other container widget), and have it fill the entire frame
* Place a *number* of *widgets* on *top* of *each* *other*
* Place a *number* of *widgets* *side* by *side*

#*Importing tkinter module*

**from** tkinter **import** \* **from** tkinter**.**ttk **import** \*

#*creating Tk window*

master = **Tk**()

#*creating a Fra, e which can expand according*

#*to the size of the window*

pane = **Frame**(master)

pane**.pack**(fill = BOTH, expand = **True**)

#*button widgets which can also expand and fill*

#*in the parent widget entirely*

#*Button 1*

b1 = **Button**(pane, text = "Click me !")

b1**.pack**(fill = BOTH, expand = **True**)

#*Button 2*

b2 = **Button**(pane, text = "Click me too")

b2**.pack**(fill = BOTH, expand = **True**)

#*Execute Tkinter*

master**.mainloop**()

* Python | ***grid()*** method in ***Tkinter***
* The ***Grid geometry manage***r puts the widgets in a ***2-dimensional table***. The master widget is split into a number of rows and columns, and each “cell” in the resulting table can hold a widget.
* The ***grid*** ***manager*** is the most flexible of the geometry managers in **Tkinter**. If you don’t want to learn how and when to use all three managers, you should at least make sure to learn this one.

#*import tkinter module*

**from** tkinter **import** \* **from** tkinter**.**ttk **import** \*

#*creating main tkinter window/toplevel*

master = **Tk**()

#*this wil create a label widget*

l1 = **Label**(master, text = "First:")

l2 = **Label**(master, text = "Second:")

#*grid method to arrange labels in respective*

#*rows and columns as specified*

l1**.grid**(row = 0, column = 0, sticky = W, pady = 2)

l2**.grid**(row = 1, column = 0, sticky = W, pady = 2)

#*entry widgets, used to take entry from user*

e1 = **Entry**(master)

e2 = **Entry**(master)

#*this will arrange entry widgets*

e1**.grid**(row = 0, column = 1, pady = 2)

e2**.grid**(row = 1, column = 1, pady = 2)

#*infinite loop which can be terminated by keyboard*

#*or mouse interrupt*

**mainloop**()

#*import tkinter module*

**from** tkinter **import** \* **from** tkinter**.**ttk **import** \*

#*creating main tkinter window/toplevel*

master = **Tk**()

#*this will create a label widget*

l1 = **Label**(master, text = "Height")

l2 = **Label**(master, text = "Width")

#*grid method to arrange labels in respective*

#*rows and columns as specified*

l1**.grid**(row = 0, column = 0, sticky = W, pady = 2)

l2**.grid**(row = 1, column = 0, sticky = W, pady = 2)

#*entry widgets, used to take entry from user*

e1 = **Entry**(master)

e2 = **Entry**(master)

#*this will arrange entry widgets*

e1**.grid**(row = 0, column = 1, pady = 2)

e2**.grid**(row = 1, column = 1, pady = 2)

#*checkbutton widget*

c1 = **Checkbutton**(master, text = "Preserve")

c1**.grid**(row = 2, column = 0, sticky = W, columnspan = 2)

#*adding image (remember image should be PNG and not JPG)*

img = **PhotoImage**(file = r"C:\Users\Admin\Pictures\capture1.png")

img1 = img**.subsample**(2, 2)

#*setting image with the help of label*

**Label**(master, image = img1)**.grid**(row = 0, column = 2,

    columnspan = 2, rowspan = 2, padx = 5, pady = 5)

#*button widget*

b1 = **Button**(master, text = "Zoom in")

b2 = **Button**(master, text = "Zoom out")

#*arranging button widgets*

b1**.grid**(row = 2, column = 2, sticky = E)

b2**.grid**(row = 2, column = 3, sticky = E)

#*infinite loop which can be terminated*

#*by keyboard or mouse interrupt*

**mainloop**()

* Python | ***place()*** method in ***Tkinter***
* The ***Place******geometry***manager is the *simplest* of the *three general geometry managers* provided in Tkinter. It allows you explicitly set the ***position*** and ***size*** of a window, either in absolute terms, or relative to another window.
* You can access the ***place*** manager through the ***place()*** method which is available for all standard widgets.
* It is usually not a good idea to use ***place()*** for ordinary window and dialog layouts; its simply to much work to get things working as they should. Use the ***pack()*** or ***grid()*** managers for such purposes.

Syntax: widget.place(relx = 0.5, rely = 0.5, anchor = CENTER)

* Note: ***place()*** method can be used with ***grid()*** method as well as with ***pack()*** method.

#*Importing tkinter module*

**from** tkinter **import** \* **from** tkinter**.**ttk **import** \*

#*creating Tk window*

master = **Tk**()

#*setting geometry of tk window*

master**.geometry**("200x200")

#*button widget*

b1 = **Button**(master, text = "Click me !")

b1**.place**(relx = 1, x =-2, y = 2, anchor = NE)

#*label widget*

l = **Label**(master, text = "I'm a Label")

l**.place**(anchor = NW)

#*button widget*

b2 = **Button**(master, text = "GFG")

b2**.place**(relx = 0.5, rely = 0.5, anchor = CENTER)

#*infinite loop which is required to*

#*run tkinter program infinitely*

#*until an interrupt occurs*

**mainloop**()

#*Importing tkinter module*

**from** tkinter **import** \* **from** tkinter**.**ttk **import** \*

#*creating Tk window*

master = **Tk**()

#*setting geometry of tk window*

master**.geometry**("200x200")

#*button widget*

b2 = **Button**(master, text = "GFG")

b2**.pack**(fill = X, expand = **True**, ipady = 10)

#*button widget*

b1 = **Button**(master, text = "Click me !")

#*This is where b1 is placed inside b2 with in\_ option*

b1**.place**(in\_= b2, relx = 0.5, rely = 0.5, anchor = CENTER)

#*label widget*

l = **Label**(master, text = "I'm a Label")

l**.place**(anchor = NW)

#*infinite loop which is required to*

#*run tkinter program infinitely*

#*until an interrupt occurs*

**mainloop**()

**27.3 Python advanced arguments**

* keyword-value arguments: Recall keyword-value arguments from previous lessons. It is a way to call a function:

function(parameter1 = argument1, parameter2 = argument2, . . . . )

* Default value arguments: Some functions has one or more required arguments (which is required to call the function). But it may have one or more other arguments set to default values already. Those are called Default value arguments (optional arguments), they can be modified if we want to.
* Default values are set when function is defined. Following has **x**, **y** as required args and **a**, **b**, **c** are default values args.

**def** my\_func(x, y, a=1, b=2, c=3):

function definition

* **\*args** (variable arguments)**:** Many positional arguments (variable arguments): Used for Unlimited positional arguments.

**def** my\_func(\*args):

function definition

***for*** n ***in*** ***args***:  *#how to use parameters of variable-arguments*

print(n)

* Returns a tuple "which are the inputs": **print**(arGs)
* So ***\*args*** is actually a variable ***tuple***. So we can access them with Indexing:

**print**(f"first argument is {arGs[0]}")

* ***args*** is just a name, we can use any name if we want, but ***\**** is needed to declare variable-arguments.

**def** **add**(\*arGs):

**print**(arGs) #*returns a tuple "which are the inputs"*

    #*So we can access them as Index*

**print**(f"first argument is {arGs[0]}")

    sum = 0

**for** n **in** arGs:

**print**(n)

        sum += n

**return** sum

**print**(f"Sum is : {**add**(1, 2, 3)}")

#*python playground.py*

* **\*\*Kwargs**: Many Keywarded Arguments: It basically creates a ***key-value dictionary*** instead of ***tuples*** as above.
* The *dictionary* is created when the function is called with *key-value paired arguments*.

**def** **calculate**(\*\*kWArgs):

**print**(kWArgs)

# Accessing kwargs arguments

**for** (key, value) **in** kWArgs**.items**():

**print**(key)

**print**(value)

# Accessing Specific kwargs arguments

**print**(kWArgs["sum"])

**calculate**(sum = 8, multiply = 9)

#  **\*\*kwargs** in **class-objects**. create **class** with optional keyword arguments

**class** mycar:

**def** **\_\_init\_\_**(self, \*\*kw):

        #*self.maker = kw["make"]*

        #*self.carName = kw["model"]*

        #*using***get()***insted of***[]***makes the aguments varible.***kwarg.get("key")**

        #*So if any of these arguments misses no error wil arise. Returns None instead*

**self.**maker = kw**.get**("make")

**self.**carName = kw**.get**("model")

car = **mycar**(make= "Nissan", model = "GT-8")

**print**(car**.**carName)

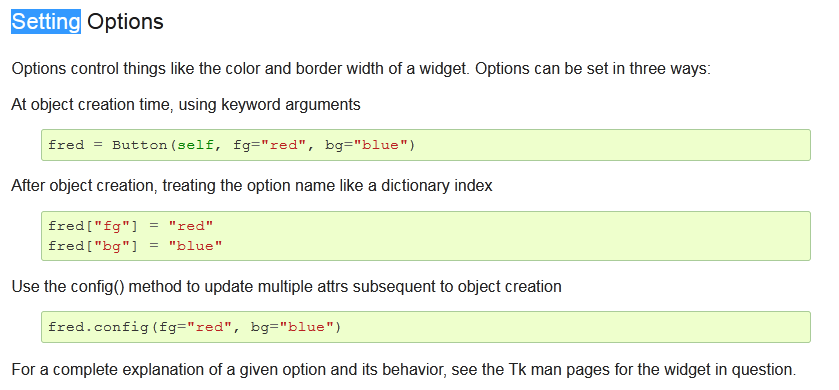
car2 = **mycar**(make= "Paganni")

**print**(f"maker : {car2**.**maker} and model : {car2**.**carName}")

#*python kw\_arg.py*

* ***kwarg.get("key")*** instead of ***kwarg["key"]:*** If any ***kwarg*** argument missed the value is set to ***None***. using ***get()*** insted of ***[]*** makes the arguments variable. ***kwarg.get("key")***

**27.4 More Tkinter options**



**import** tkinter

wiNdw = tkinter**.Tk**()

#*Title*

wiNdw**.title**("First GUI ????")

#*size*

wiNdw**.minsize**(width = 500, height = 300)

#*Label() and pack()*

my\_label = tkinter**.Label**(text= "I am a Label", font=("Courier", 24, "bold"))

#*Geometry manager pack() needs to display this label*

my\_label**.pack**(side= "bottom")

my\_label["text"] = "Hello"

#*Setting options in different way*

my\_label\_2 = tkinter**.Label**(font=("Courier", 24, "bold"))

#*Geometry manager pack() needs to display this label*

my\_label\_2**.pack**(side= "left")

my\_label\_2**.config**(text = "Bye !! ")

wiNdw**.mainloop**() #*always in end of the program*

#*python tkinter\_more.py*

* Importing all modules using "\*": **from** tkinter **import** \*

**import** tkinter

wiNdw = tkinter**.Tk**()

#*Title*

wiNdw**.title**("First GUI ????")

#*size*

wiNdw**.minsize**(width = 500, height = 300)

# --------------- interactivity -----------

#*Creating button*

# Event-listener function for Event-listener

**def** **clicked**():

**print**("clicked")

    new\_label = tkinter**.Label**()

    new\_label**.config**(text = "Button clicked")

    new\_label**.pack**()

# to make click event use above function with command = clicked

button = tkinter**.Button**(text = "click me", command = clicked)

button**.pack**()

# Input - field "Entry"

text\_input = tkinter**.Entry**(width = 10)

text\_input**.pack**()

# Grabbing the input

**def** **enter\_text**():

    user\_input = text\_input**.get**()

**print**(user\_input)

#*to make click event use above funtion with command = clicked*

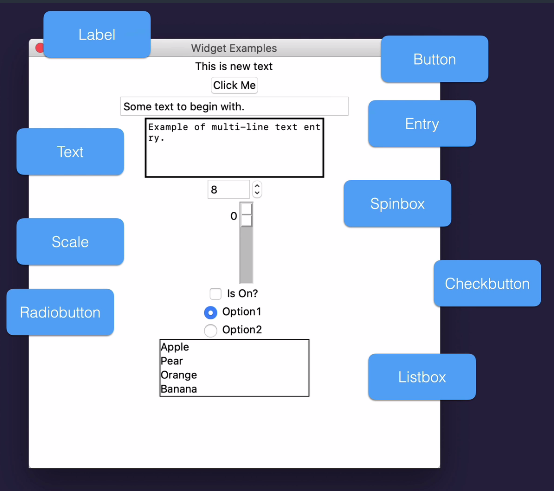
button\_2 = tkinter**.Button**(text = "click me", command = enter\_text)

button\_2**.pack**(side = "bottom")

wiNdw**.mainloop**() #*always in end of the program*

#*python tkinter\_more.py*

**27.4 Some Tkinter widgets**



#*-------------  Tkinter Widgets  -------------*

**from** tkinter **import** \*

#*Creating a new window and configurations*

window = **Tk**()

window**.title**("Widget Examples")

window**.minsize**(width=500, height=500)

#*Labels*

label = **Label**(text="This is old text")

label**.config**(text="This is new text")

label**.pack**()

#*Buttons*

**def** **action**():

**print**("Do something")

#*calls action() when pressed*

button = **Button**(text="Click Me", command=action)

button**.pack**()

#*Entries*

entry = **Entry**(width=30)

#*Add some text to begin with*

entry**.insert**(END, string="Some text to begin with.")

#*Gets text in entry*

**print**(entry**.get**())

entry**.pack**()

#*Text*

text = **Text**(height=5, width=30)

#*Puts cursor in textbox.*

text**.focus**()

#*Adds some text to begin with.*

text**.insert**(END, "Example of multi-line text entry.")

#*Get's current value in textbox at line 1, character 0*

**print**(text**.get**("1.0", END))

text**.pack**()

#*Spinbox*

**def** **spinbox\_used**():

    #*gets the current value in spinbox.*

**print**(spinbox**.get**())

spinbox = **Spinbox**(from\_=0, to=10, width=5, command=spinbox\_used)

spinbox**.pack**()

#*Scale*

#*Called with current scale value.*

**def** **scale\_used**(value):

**print**(value)

scale = **Scale**(from\_=0, to=100, command=scale\_used)

scale**.pack**()

#*Checkbutton*

**def** **checkbutton\_used**():

    #*Prints 1 if On button checked, otherwise 0.*

**print**(checked\_state**.get**())

#*variable to hold on to checked state, 0 is off, 1 is on.*

checked\_state = **IntVar**()

checkbutton = **Checkbutton**(text="Is On?", variable=checked\_state, command=checkbutton\_used)

checked\_state**.get**()

checkbutton**.pack**()

#*Radiobutton*

**def** **radio\_used**():

**print**(radio\_state**.get**())

#*Variable to hold on to which radio button value is checked.*

radio\_state = **IntVar**()

radiobutton1 = **Radiobutton**(text="Option1", value=1, variable=radio\_state, command=radio\_used)

radiobutton2 = **Radiobutton**(text="Option2", value=2, variable=radio\_state, command=radio\_used)

radiobutton1**.pack**()

radiobutton2**.pack**()

#*Listbox*

**def** **listbox\_used**(event):

    #*Gets current selection from listbox*

**print**(listbox**.get**(listbox**.curselection**()))

listbox = **Listbox**(height=4)

fruits = ["Apple", "Pear", "Orange", "Banana"]

**for** item **in** fruits:

    listbox**.insert**(fruits**.index**(item), item)

listbox**.bind**("<<ListboxSelect>>", listbox\_used)

listbox**.pack**()

window**.mainloop**()

#*python tkinter\_more\_other.py*

**27.5 Details on Layout managers pack(), grid() and place()**

* Place is precise positioning:

Widget.palce(x=, y=)

* Grid is grid-like positioning:

button**.grid**(column=1,row=1)

**import** tkinter

wiNdw = tkinter**.Tk**()

#*Title*

wiNdw**.title**("First GUI ????")

#*size*

wiNdw**.minsize**(width = 500, height = 300)

# Place - layout: Absolute positioning

my\_label\_2 = tkinter**.Label**(text="label", font=("Arial", 24, "italic"))

my\_label\_2**.place**(x=300, y=0)

button\_2 = tkinter**.Button**(text="Click")

button\_2**.place**(x=100, y=200)

input\_2 = tkinter**.Entry**(width=10)

input\_2**.place**(x=200, y=200)

# Grid - layout: It is relative. Relative to first widget

my\_label = tkinter**.Label**(text="label", font=("Arial", 24, "italic"))

my\_label**.grid**(column=0,row=0)

button = tkinter**.Button**(text="Click")

button**.grid**(column=1,row=1)

input = tkinter**.Entry**(width=10)

input**.grid**(column=3,row=2)

#*challenge*

new\_button = tkinter**.Button**(text = "New Button")

new\_button**.grid**(column = 2, row = 0)

# add padding to windows and widgets

wiNdw**.config**(padx = 20, pady = 20)

new\_button**.config**(padx = 30, pady = 30)

wiNdw**.mainloop**()

#*python tkinter\_layout\_positioning.py*

* Exercise 27.1: Project : Kilometer Converter

Practice version

**import** tkinter

kms = 0

winDo = tkinter**.Tk**()

winDo**.minsize**(height = 200, width = 300)

inPut = tkinter**.Entry**(width = 10)

inPut**.grid**(column = 0, row = 0, padx = 10, pady = 50)

label\_1 = tkinter**.Label**(text = "Miles", font = ("Arial", 24, "normal"))

label\_1**.grid**(column = 1, row = 0)

label\_2 = tkinter**.Label**(text = f"is equal to {kms} km", font = ("Arial", 24, "normal"))

label\_2**.grid**(column = 0, row = 1)

**def** **calc\_miles**():

    miles = **int**(inPut**.get**())

    label\_2**.config**(text = f"is equal to {miles\*1.609} km")

calculate = tkinter**.Button**(text = "calculate", command = calc\_miles)

calculate**.grid**(column = 0, row = 2)

winDo**.mainloop**()

#*python miles\_to\_km\_GUI.py*

Instructor version

**import** tkinter

**def** **calc\_miles**():

    miles = **float**(miles\_inPut**.get**())

**print**(miles\*1.609)

    km\_result\_label**.config**(text = f"{**str**(miles\*1.609)}")

winDo = tkinter**.Tk**()

winDo**.title**("Miles to Kilometer converter")

winDo**.minsize**(height = 200, width = 300)

winDo**.config**(padx = 20, pady =20)

miles\_inPut = tkinter**.Entry**(width = 7)

miles\_inPut**.grid**(column = 1, row = 0, padx = 10, pady = 50)

miles\_label = tkinter**.Label**(text = "Miles", font = ("Arial", 24, "normal"))

miles\_label**.grid**(column = 2, row = 0)

isequal\_label = tkinter**.Label**(text = f"is equal to", font = ("Arial", 24, "normal"))

isequal\_label**.grid**(column = 0, row = 1)

km\_label = tkinter**.Label**(text = "km", font = ("Arial", 24, "normal"))

km\_label**.grid**(column = 2, row = 1)

calculate = tkinter**.Button**(text = "calculate", command = calc\_miles)

calculate**.grid**(column = 1, row = 2)

km\_result\_label = tkinter**.Label**(text = "0", font = ("Arial", 24, "normal"))

km\_result\_label**.grid**(column = 1, row = 1)

winDo**.mainloop**()

#*python miles\_to\_km\_GUI\_soln.py*