**MULTITHREADING**

**THREADS:**

* A thread is a separate flow of execution. This means that your program will have two things happening at once.
* The threads may be running on different processors, but they will only be running one at a time.
* Threads allows a program to operate more efficiently by doing multiple things at the same time.
* Threads can be used to perform complicated tasks in the background without interrupting the main program.
* Threads sometimes called light-weight processes and they do not require much memory.

MULTI TASKING:

Executing several tasks simultaneously.

Example:

In Our Class Room: we are doing browsing, program executions, writing Notes, listening class, Checking Mobiles, Discussing some topics etc.…

They are two types of multi taskings:

1.Process based multitasking.

2.Thread based multitasking.

* Process based multitasking:

It is one type of multitasking. It executes several tasks simultaneously. Each task is separate independent process. This type of multitasking is called process-based multitasking.

Example:

1.Browsing something in system.

2.Downloading some files from the browser.

3.Listening songs in the same system.

4.Using some text editors in the system

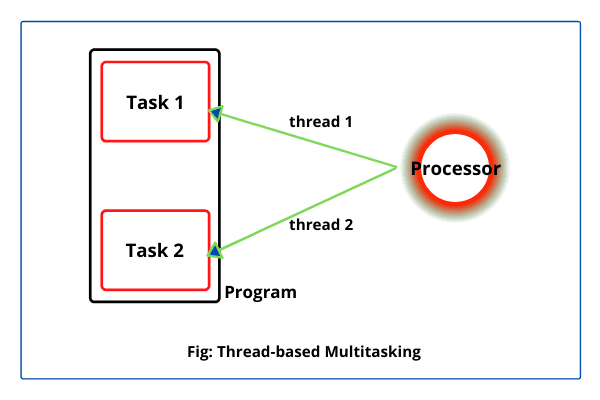
All the activities are running in our system simultaneously. But these are independent each other such type of multitasking is by default called as process-based multitasking.

Most of the times these process- based multitasking is applicable at operating system.

* Thread based multitasking:

Thread based have an only one process with in the process of multiple parts are required to execute simultaneously.

Example:



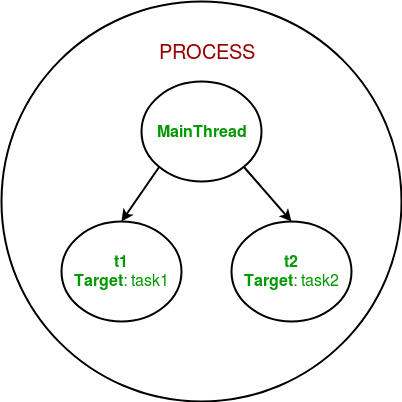
1.process means with in the same program multiple threads by default consider as process- based multitasking.

2.Thread means an independent part of a program.it is python object. every thread has independent job is available.

**MULTITHREADING IN PYTHON:**

**Multithreading** is defined as the ability of a processor to execute multiple threads in simultaneously. In Python, or any programming language, a thread is used to execute a task where some waiting is expected. So that the main program does not wait for the task to complete, but the thread can take care of it simultaneously

Like:



Program on multithreading:

Create a thread:

Import threading

Class Thread:

def run(self):

print (thread function)

for x in range (3):

t= Thread ()

start a thread:

A thread is started by applying start () method on the thread object.

from import threading \*

import time

class Hello:

def start (Thread):

for x in range (5):

print(“Hello”)

time. Sleep ()

s1=Hello ()

s1. start

output:

Hello

Hello

Hello

Hello

When to USE multithreading in python:

Multithreading in python can be used:

1.Multiple tasks need to achieved.

2.Task do not have interdependency.

Multithreading is use to saving the time and improving performance. But it cannot be applicable anywhere.in the previous example like process based.

HOW TO CREATE A THREADS IN PYTHON:

They are three methods to creating multithreading.

1.Without creating a class.

2.By Extending thread class.

3.Without Extending thread class.

* Without creating a class:

from import threading \*

def fun ():

for x in range (5):

print (“executing fun1…”)

t1=Thread (target =new)

t1. start ()

print(“hello”)

output:

executing fun1… hello

executing fun1…

executing fun1…

executing fun1…

* Before creating a thread in python, you have to import threading module.

Import threading module command in python is “from import threading\* “. Here, “every process has executing one thread that is main thread”. Here defines a function with function name like fun. In that creating a block of code whereas define for loop range of 5, then printed after that creating child thread using a thread class which is present in the threading module, this child thread specified with target(fun) function is defined. after that (t1=Thread ()) this is used to executed new function not the main thread. Following that t1. Start () is used to start the child thread. after that print statement is define.

* Coming to output child thread is executed but main thread is not waiting for completing the child thread. that is “bye”. Now we want to wait for the main thread we are using join () function. like this…

From import the threading \*

def fun ():

for x in range (5):

print (“executing fun1…”)

t1=Thread(target=fun)

t1. Start ()

t1. Join ()

print(“hello”)

output:

executing fun1…

executing fun1…

executing fun1…

executing fun1…

hello

* Here, join () function is used main thread is waiting for the until finished of the child thread task. after that final print statement has been executed by the main thread.

from import the threading \*

def fun ():

for x in range (5):

print (“executing fun1…”, current\_ thread (). get Name ())

t1=Thread(target=fun)

t1. Start ()

t1. Join ()

print (“hello”, current \_ thread (). get Name ())

output:

executing fun1... Thread-1 (fun)

executing fun1... Thread-1 (fun)

executing fun1... Thread-1 (fun)

executing fun1... Thread-1 (fun)

hello Main Thread

* Here print statement with in the function using current thread and get name () is printed to current thread execution. Above the output Thread-1 is not a main thread it is child thread.in the last print statement using same function like current thread. get name () it executes the main thread as see in output.
* BY EXTENDING THREAD CLASS:

Used by extending thread class to create a thread using only two functions that is run and \_\_in it \_\_function. every python function that is define by class using self -parameter has to be specified.

Example:

Import threading

Class A:

def run(self):

for x in range (4):

print (“child =”, current \_thread (). get Name ())

object = A ()

object. start ()

object. join ()

print (“control return to”, current \_ thread. get Name ())

output:

child = Thread-1

child = Thread-1

child = Thread-1

child = Thread-1

control to Main Thread

* In the above program create a class A.by inheriting the thread class present in the threading module. after that over ridding the run function and by default must specified by self -parameter after that use for loop of the range of 4 executing the print statement. then create an object for the class A and start the executing child thread. then run the program. executing child class in according to the range. Then control will be in main thread.

2.WITHOUT EXTENDING THE THREAD CLASS:

Class Example:

def B(self):

l= [1,2,3,0.3,45]

for x in l:

print (“child thread”, x)

object = Example ()

t1=Thread (target=object. B)

t1. Start ()

t1. Join ()

print(“hello”)

output:

child thread 1

child thread 2

child thread 3

child thread 0.3

child thread 45

hello

* In the above program create a class with the name Example then define a function with function name B, self is the default parameter in the python.

In this function we have given a list of elements to print the child thread one after one by using for loop. following that creating object with class name then creating a thread by inheriting the threading class as specified by object. B start () function used to execute the thread and join () function is used to wait the executing the finishing the child thread. output of this program should be child thread with the elements one after another. After that it is going to main thread then print the print statement hello.

ADVANTAGES OF MULTI THREADING:

* Enhance the performance and by decreasing the development time.
* Simultaneously and parallelised occurrences in the tasks.
* Use CPU resources in better manner.

In simple words multithreading is used reducing the time and improving the performance.

Example with using thread:

from import threading \*

import time

start \_time=time. Perf \_counter ()

class Hello:

def start (Thread):

for x in range (5):

print(“Hello”)

time. Sleep (1)

end \_time= time. Perf \_counter ()

class Hi:

def start (Thread):

for x in range (5):

print(“Hi”)

time. Sleep (1)

s1=Hello ()

s2=Hi ()

print(“Bye”)

print (f’ finished in{round(end\_time-start\_time,2) second(s)’})

s1. Join ()

s2. Join ()

output:

Hello

Hello

Hello

Hello

Hi

Hi

Hi

Hi

Bye

finished in 0.0 second(s)

In the above program using import time module for execution time in the process.

TKINTERS

TKINTER: Tkinter is the standard library for python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

What is a GUI:

GUI is nothing but a desktop application which help to interact with computers. They used to perform different tasks in desktops computers and any other electronic devices. In generally we are using daily GUI apps.

GUI apps like:

1.Text editors

2.Games

3.Apps

* Text editors: Text editors is use to create, read, write and update and delete different type of files.
* Games: GUI Games like sudoku, chess other games which we can play.
* Apps: Apps like chrome, Microsoft Edge etc...

They are different types GUI apps we are using daily on desktops and laptops. we have to learn how to create those types of apps.

Python libraries for GUI:

* Python libraries are used to design own graphical user interface. Python has lot of libraries and these are main four libraries

1.Kivy

2.python Qt

3.Wx python

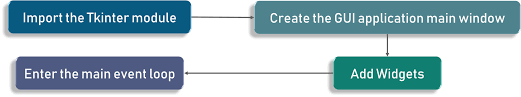
4.Tkinter.

All of these Tkinter is preferred by a lot of developers and learners because it is a simple and essay.

WHAT IS TKINTER:

Tkinter is a pre-defined module in python. which is used to create a simple GUI (Graphical user interface) app.

FUNDAMENTALS OF TKINTER:

[](https://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.edureka.co%2Fblog%2Fwp-content%2Fuploads%2F2018%2F12%2F1.png&imgrefurl=https%3A%2F%2Fwww.edureka.co%2Fblog%2Ftkinter-tutorial%2F&tbnid=MRk2SKGRRQjiyM&vet=12ahUKEwiajfT7zPD6AhWVidgFHVqFCHIQMygCegUIARDJAQ..i&docid=A052Bp3Vkth7RM&w=2048&h=381&q=first%20window%20flow%20using%20tkinter%20python&ved=2ahUKEwiajfT7zPD6AhWVidgFHVqFCHIQMygCegUIARDJAQ)

[Python Tkinter for Python GUI ...](https://www.edureka.co/blog/tkinter-tutorial/" \o "Python Tkinter Tutorial | Python GUI Programming Using Tkinter | Edureka" \t "_blank)

This is how to execute the application actually, first import the Tkinter module then create the Gui application main window. This window performing operations and displaying everything. Followed by that adding the widgets and lastly enter the main event loop. Here, Event loop is nothing but a telling the code to keep displaying the window until manually close it.

Example:

import tkinter

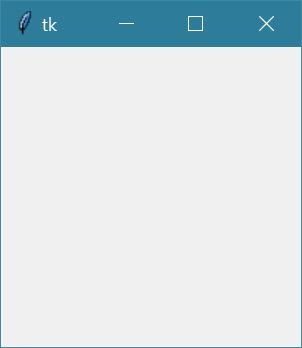
window= tkinter. Tk () # to rename the title of the window

#Pack is used to show the objects in the window

label=tkinter. Label (window). Pack ()

window. main loop ()

output:



Here, we are creating a window by importing the tkinter module and add any text on the window.

Label is nothing but a what output need to be shown on the window.

Adding widgets to the applications:

* A Widget is an element of a Graphical user interface that displays the information or provide a specific way for user to interact with the operating system or an application. Widgets are something like elements in the HTML, we find the different types of the widgets to the different types of html in tkinter. Those are label, Button, Entry, combo box, check button, Radio, scrolled text, spin Box, menu bar, Note book.
* **Button -**widget is used to place the buttons in the tkinter and then canvas is used to draw the shapes in the GUI.
* **Check Button** **-**check button we have to use the create check button in the application. we can select more than one option at a time.
* **Entry-**Entry is used to create input fields in the Graphical user interface (GUI).
* **Label-** label is used to create single line widgets like text, images etc...
* **Menu Bars-** Menu bars are used to create menus in the GUI applications.

**Label**: you can use set of label fonts so you can make it bigger and maybe bold.it is a single line definition.

Example:

import tkinter

window= tkinter. Tk () # to rename the title of the window

label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

label. grid (column=0, row=0)

window. main loop ()

output:



We can set the default window size using geometry function set it as per the requirement. Grid function is use to place the widget on the window.

Example:

import tkinter

window= tkinter. Tk () # to rename the title of the window

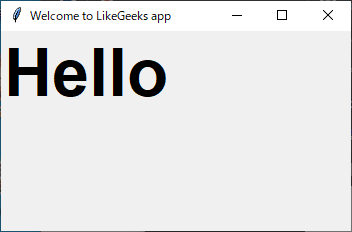
label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

window. geometry=(“352x232”)

label. grid (column=0, row=0)

window. main loop ()

output:



Here, window width is 352 pixels and height are 232 pixels.

BUTTON: Adding button to the window, the button is created and added to the new window the same the label.

Example:

import tkinter

window= tkinter. Tk () # to rename the title of the window

label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

window. geometry=(“352x232”)

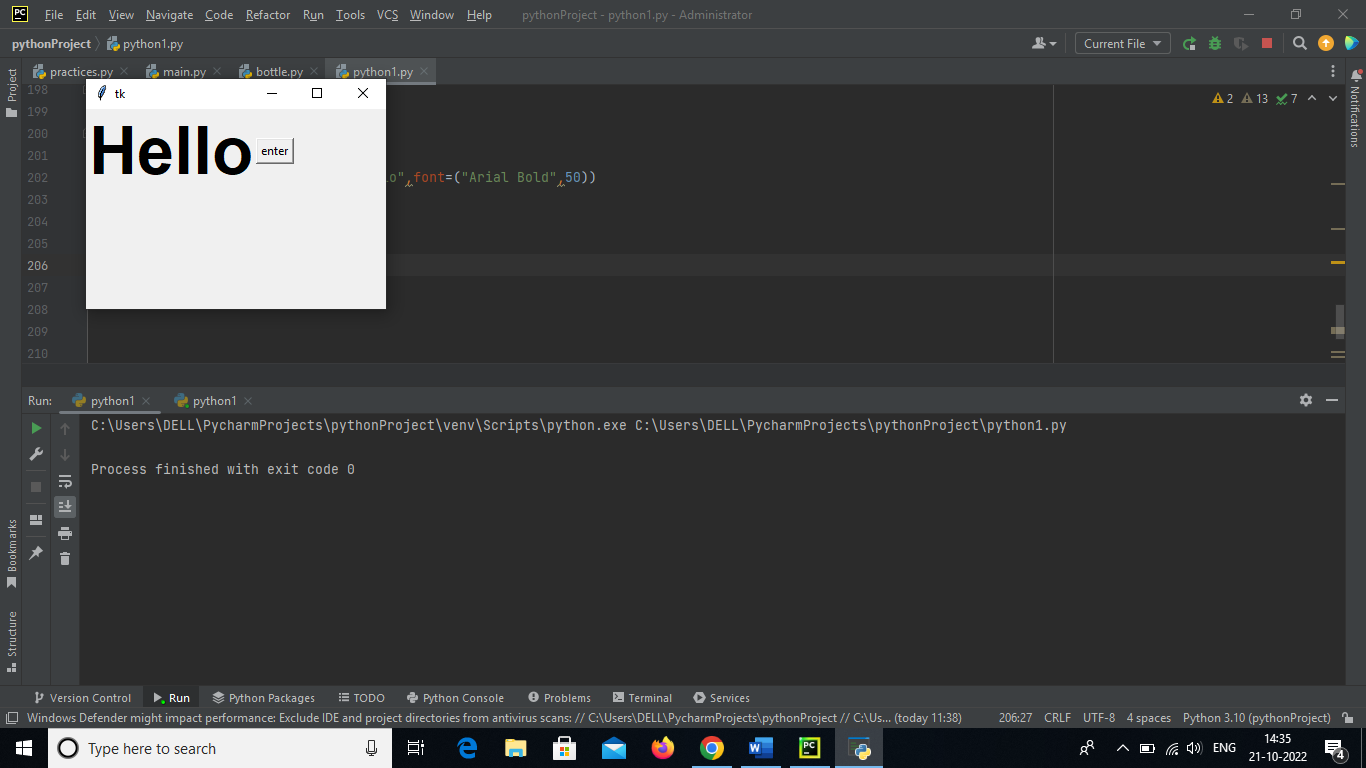
bt =Button (window, text=” enter”)

bt. grid (column=1, row=0)

label. grid (column=0, row=0)

window. main loop ()

output:



Here, using the grid function to set the button position on are particular window.

* We can change foreground for a button or any other widgets using fg property.
* Also, we can the background colour for any widget using bg property.

Example:

import tkinter

window= tkinter. Tk () # to rename the title of the window

label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

window. geometry=(“352x232”)

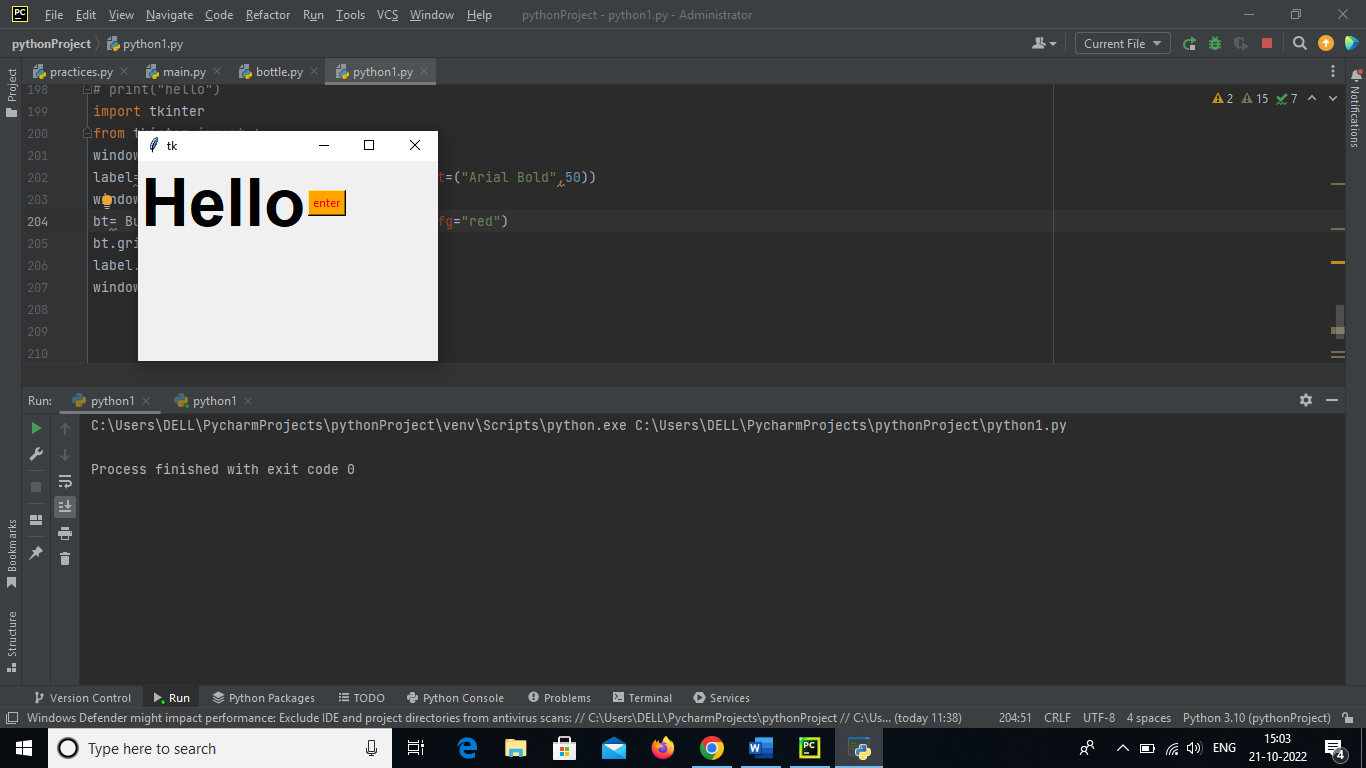
bt =Button (window, text=” enter”, bg=” orange”, fg=” red”)

bt. grid (column=1, row=0)

label. grid (column=0, row=0)

window. main loop ()

output:



Here, bg and fg properties are changing the background and foreground colours. Orange is background colour and foreground is the text that is enter will be in red.

* Now adding the click event button. First, we will write the function that we need to execute when the button is clicked.

Example:

import tkinter

from tkinter import \*

def clicked ():

window= tkinter. Tk () # to rename the title of the window

label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

label. configure (text=” button clicked”)

bt =Button (window, text=” enter”, command=clicked)

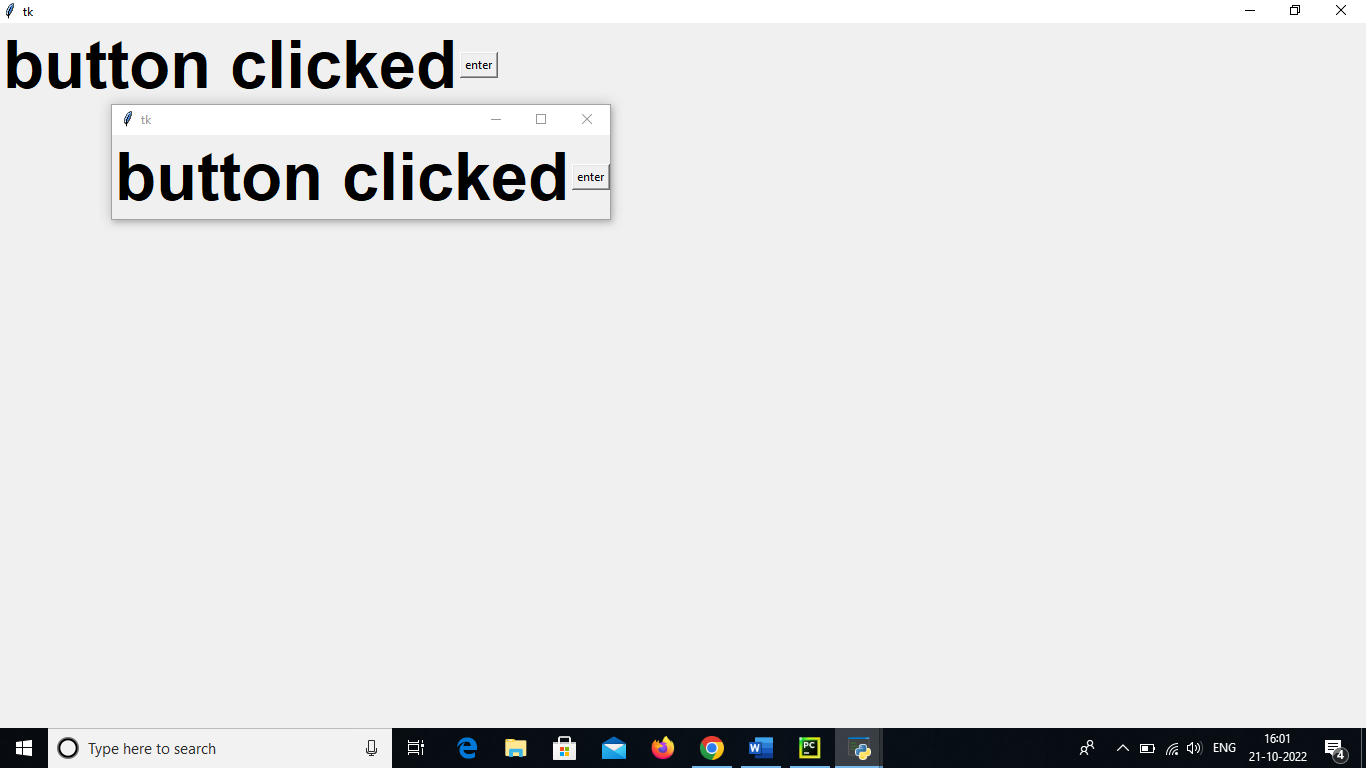
bt. grid (column=1, row=0)

label. grid (column=0, row=0)

window. main loop ()

clicked ()

output:



Here function that will execute the button click event and writing the button with in the function.

ENTRY: It is used create input fields in the GUI. In the previous python GUI examples, we saw the simple widgets. now try getting to the user input using Tkinter Entry class (Tkinter textbook).

Example:

import tkinter

from tkinter import \*

window= tkinter. Tk ()

txt=Entry (window, width=10)

txt. grid (column=1, row=0)

def clicked ():

result=” welcome to” + txt.get ()

label=tkinter. Label (window, text=” Hello”, font= (“Arial Bold”,50))

label. configure (text=” button clicked”)

bt =Button (window, text=” enter”, command=clicked)

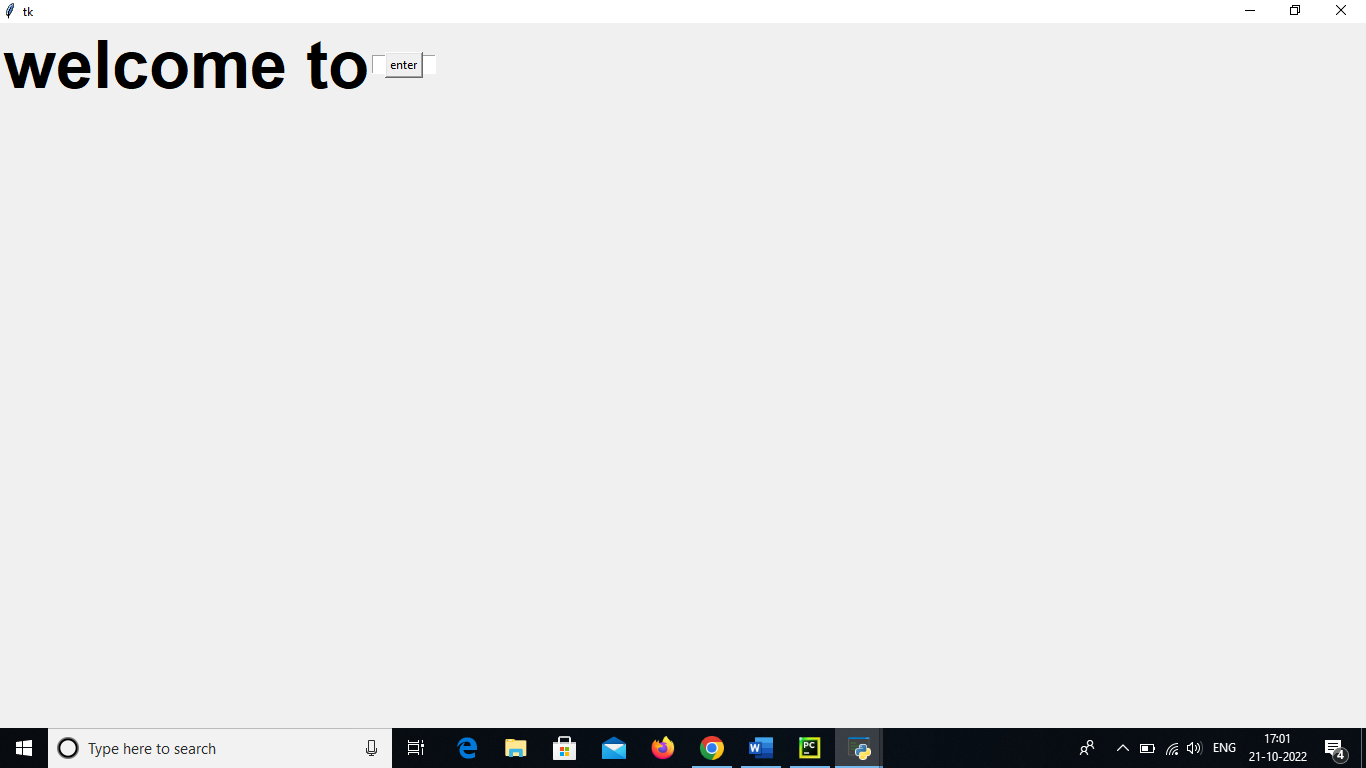
bt. grid (column=1, row=0)

label. grid (column=0, row=0)

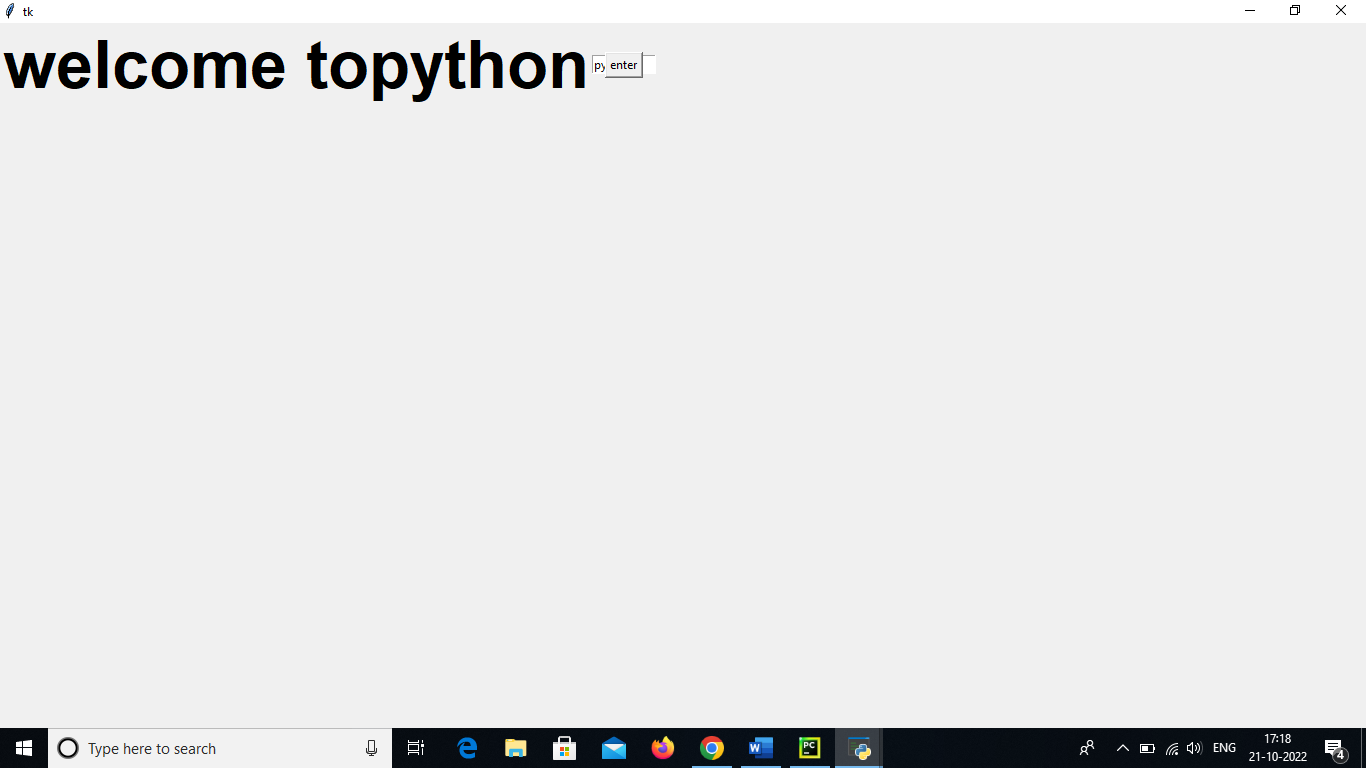
window. main loop ()

clicked ()

output:



Here creating a text box using Tkinter Entry class. Once button is clicked show “welcome to “concatenated with the whatever entered into the text area like this…



COMBOBOX: It is nothing but drop down the menu with the options. Combo box widgets are very easy to use and widely used as well.

Example:

import tkinter

from tkinter import \*

from tkinter.ttt import combobox

window=tkinter. Tk ()

combo=combobox(window)

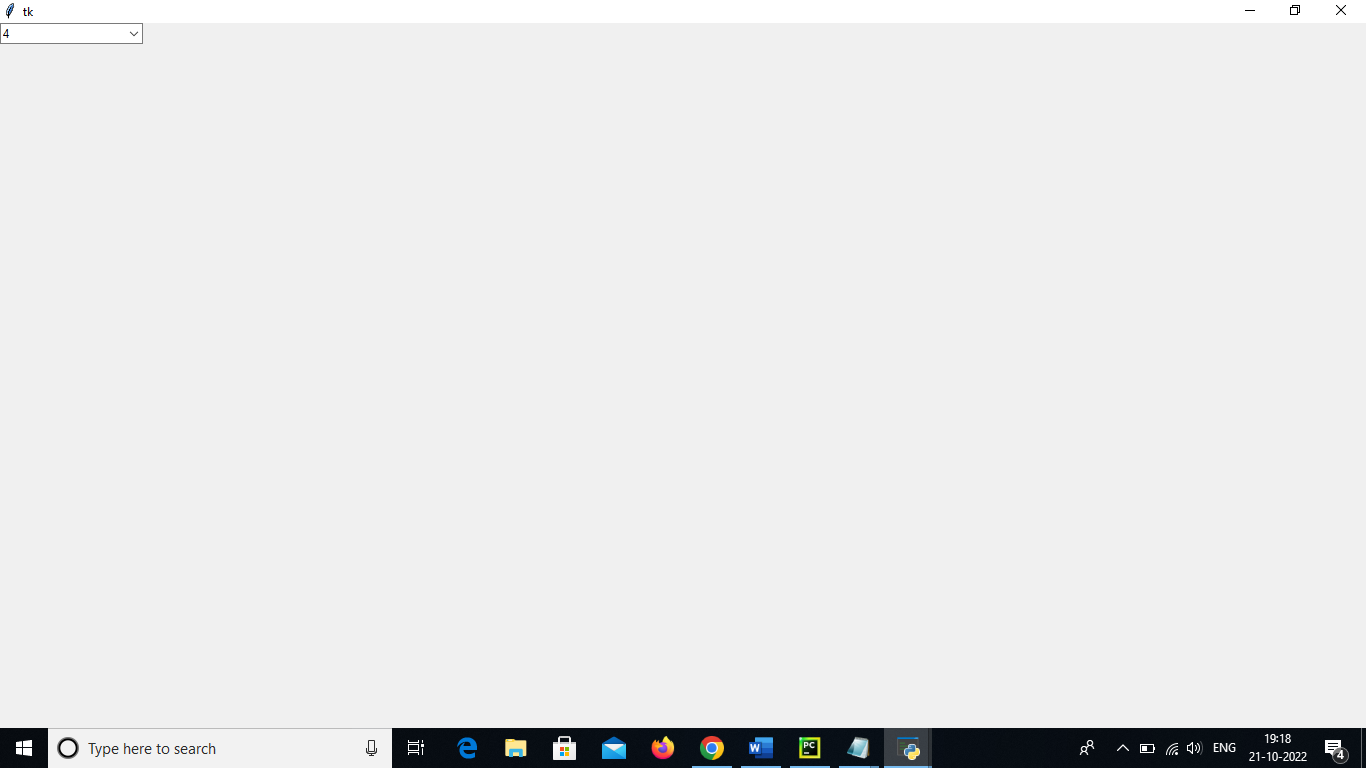
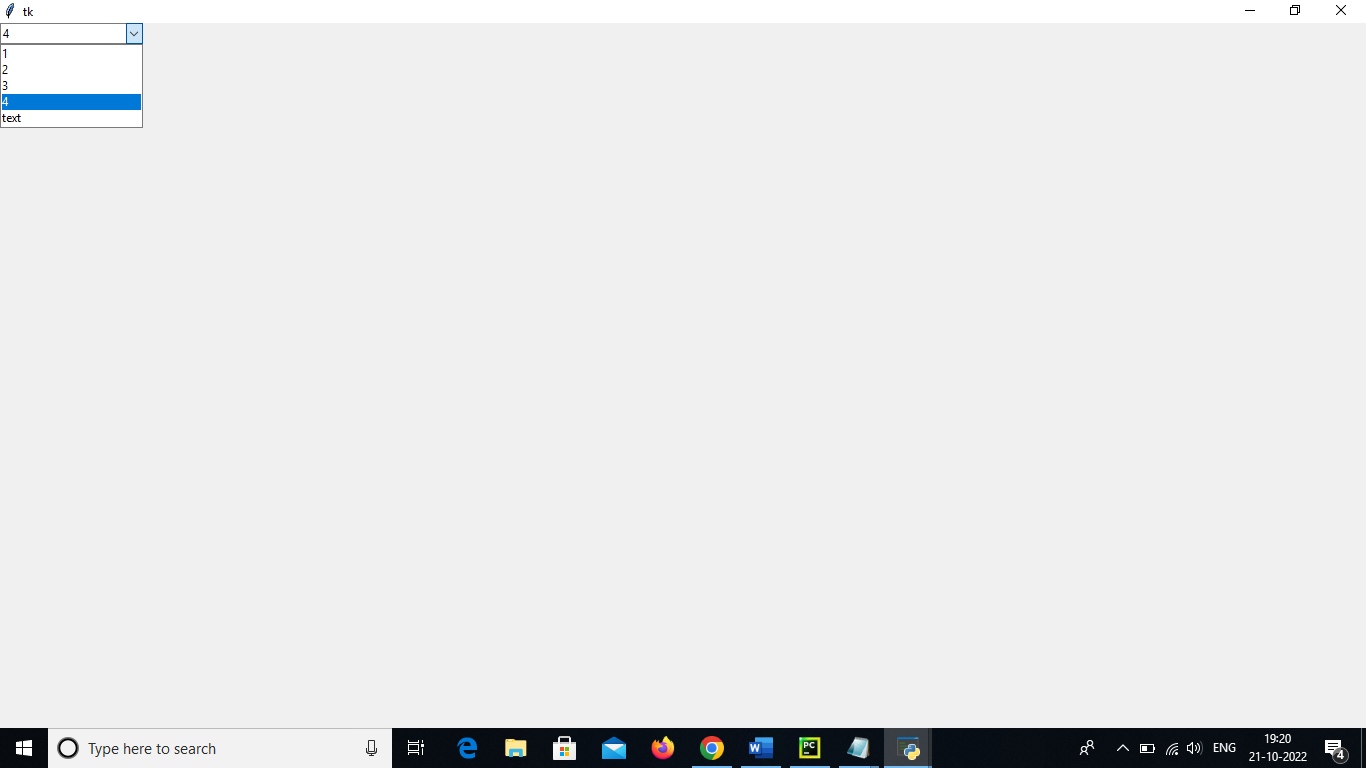
combo[“values”] =(1,2,3,4,”text”) # Adding combo box items using the tuple.

combo. Current (3) # setting the selected items.

combo. grid (column=0, row=0)

window. main loop ()

output:

* Here, they are no parameters in the combo box definition except window after define a set of values such as ranging from one to five and some text. Where one to five is numeric inputs but we have textual input as well. Finally use grid function to place the widget on the window.
* In the output have to drop down the menu and display all have defined in the code.

Radio button: To add the radio buttons, simply we can use radio button class.

Example:

import tkinter

from tkinter import \*

window=tkinter. Tk ()

rad1=Radio button (window, text=” HELLO”, value=1) # set the values for every radio button with a different value, otherwise, they won’t work.

rad2=Radio button (window, text=” python”, value=2)

rad3=Radio button (window, text=” world”, value=3)

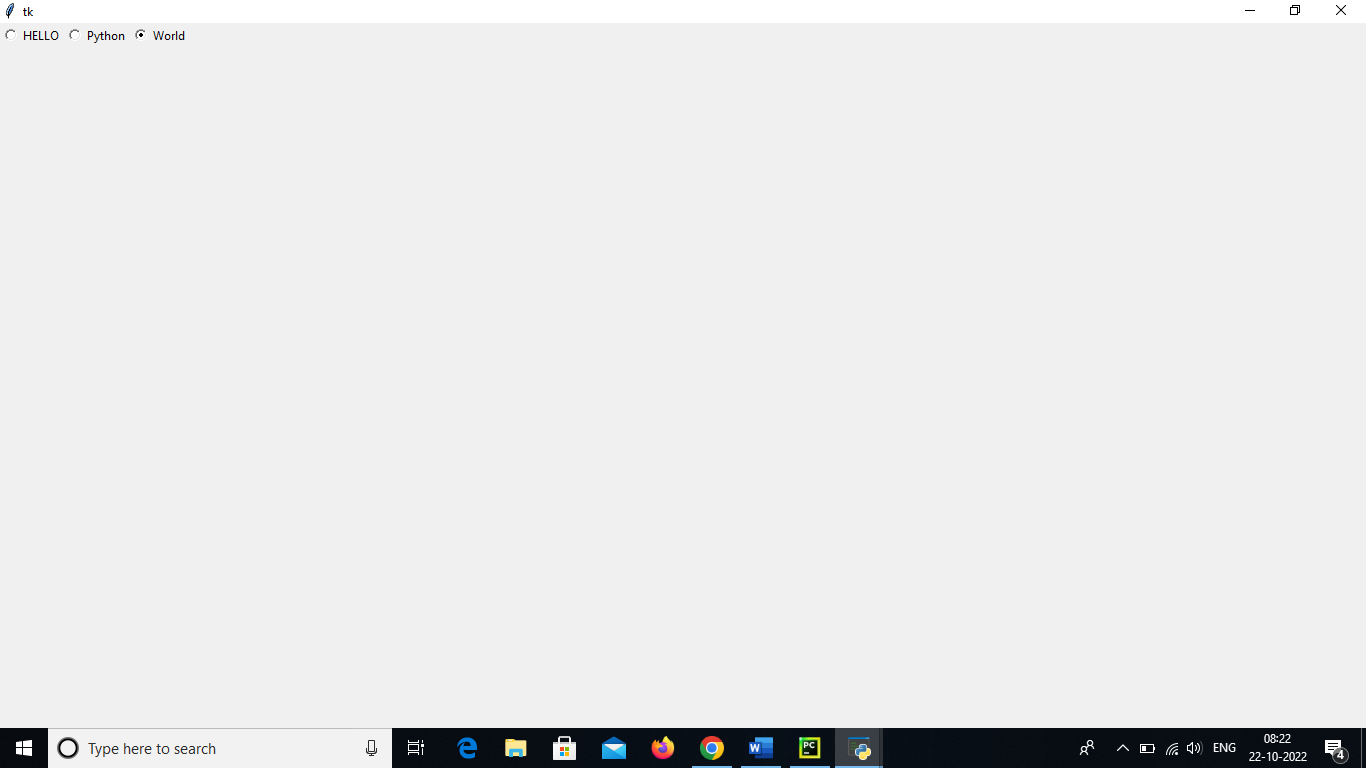
rad1.grid(column=0, row=0)

rad2.grid(column=1, row=0)

rad3.grid(column=2, row=0)

window. main loop ()

output:



* Here value parameters are different one, two and three, unique value is used to address the radio button. Value should be unique but text is same whatever you want the text, and grid function is used place the widget on the window.
* In the output of radio button, we have multiple grids but we can select one option at a time.
* All tkinter widgets have geometric measurements.

Geometry measurement class are:

1.pack ()

2.grid ()

3.place ()

* Pack (): it organizes the widgets in the block, which means it occupies entire available width. This method is showing the entire the window.
* Grid ():it organizes the widgets in the table like structures (rows and columns).
* Place ():it is used to place the widgets at a specific position you want.

Organizing layout and widgets: we use frame class to arrange layout in a window.

Frame: Frame is used to create the divisions in the window. You can align the frames as you like with side parameter of pack () method.

Button: Button is used to create a button in the window. It takes several parameters like text (value of the button), fg (colour of the text), bg (back ground colour of the text)

Example:

import tkinter

from tkinter import \*

window=tkinter. Tk ()

window. tittle(“python”)

top frame=tkinter. Frame(window). pack ()

bottom frame=tkinter. Frame(window). pack (side=” bottom”)

bt1=tkinter. Button (top frame, text=” Button1”, fg=” red”). pack ()

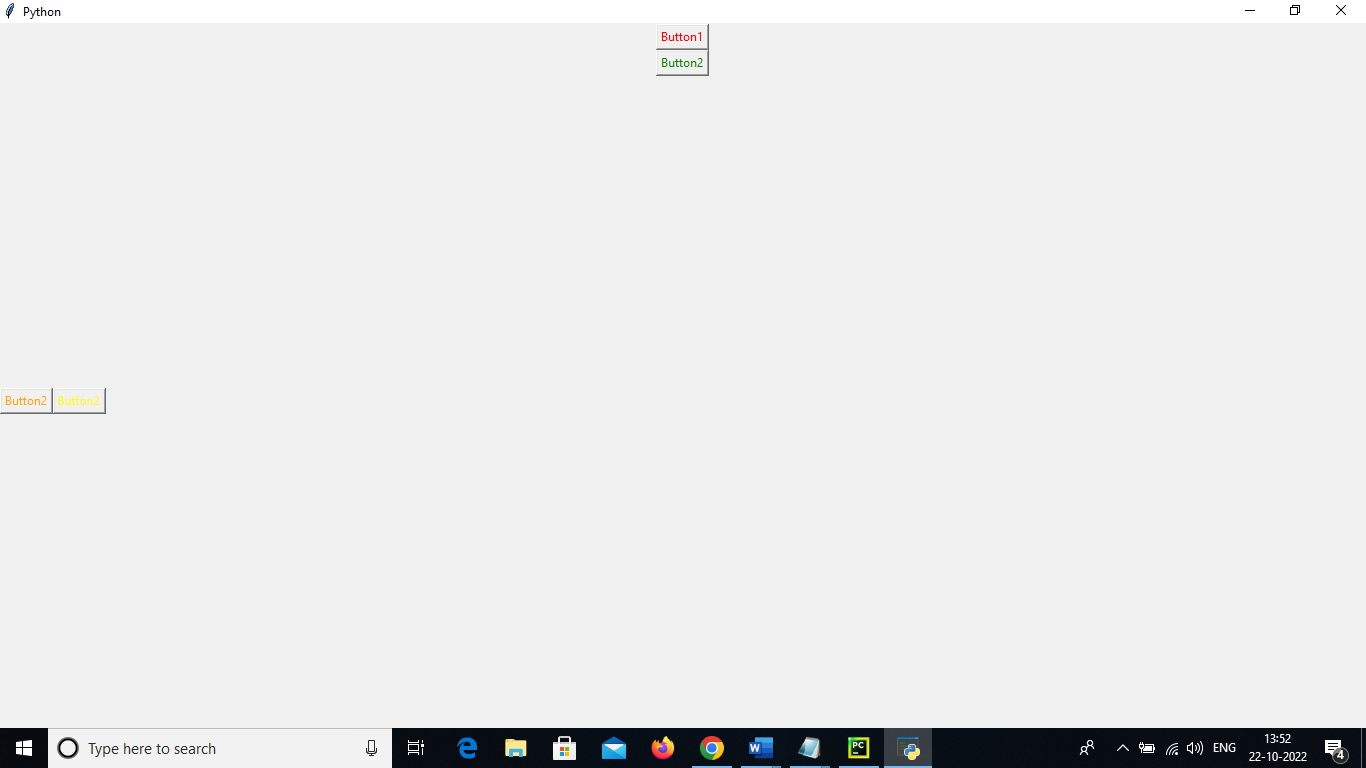
bt2=tkinter. Button (top frame, text=” Button2”, fg=” green”). pack ()

bt3=tkinter. Button (top frame, text=” Button2”, fg=” orange”). pack (side=” left”)

bt4=tkinter. Button (top frame, text=” Button2”, fg=” yellow”). pack (side=” left”)

window main loop ()

output:



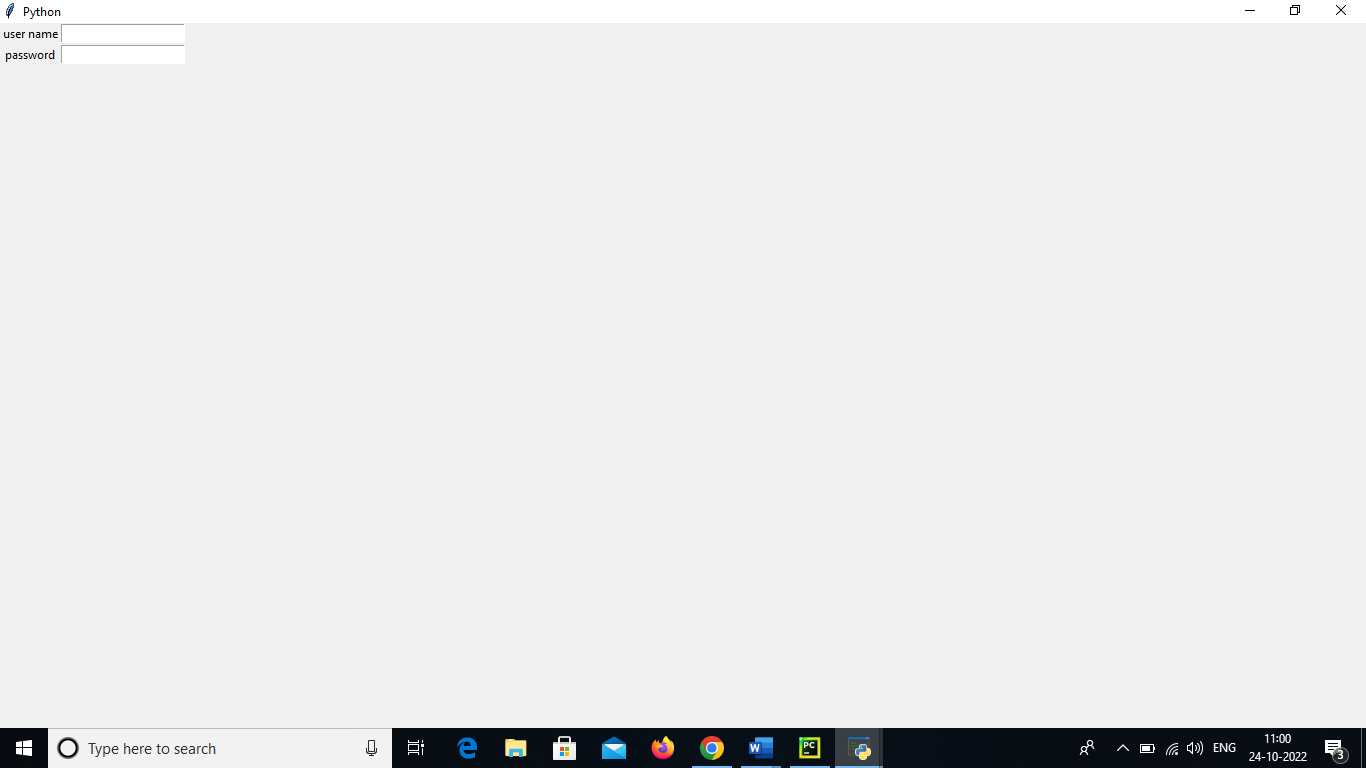
* Here import the tkinter and entered tittle called python and defining two frames (top frame and bottom frame) and creating some widgets in the top frame and bottom frame.in output button 1 fore ground colour will be red and next button will be green these are in top frame and button3, button 4 will be in bottom frame and colour should be what we define in the program. Here, when click the button nothing is happen because no code using button in the program.

GRID: grid is another way to organize the widgets.it uses in rows columns concepts.

Example:

**import** tkinter  
**from** tkinter **import** \*  
window=tkinter. Tk ()  
window. Title(**"Python"**)  
tkinter. Label (window, text=**"user name"**). grid(row=0)  
tkinter. Entry(window). grid (row=0, column=1)  
tkinter. Label (window, text=**"password"**). grid(row=1)  
tkinter. Entry(window). grid (row=1, column=1)window. main loop ()

output:



* In this program importing the tkinter and window title is python and then we have a label, text to be written as user name. This is the put in the place of (zero, zero) i.e., Top left and then we have an entry widget put in any text and here row 0 and column 1. which is actually right side of the user’s name. Next, we have a password again same as label, this is placed on one and zero it is right below the user’s name. Another entry widget as similarly row 1 and column1.in this program we observed we are placing at (0,0), (0,1),(1,0)and(1,1) in the form of matrix. Hence it is called as the grid.