

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in
Information Technology

By

DINESH KUMAR A

Reg. No. 37120087



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

AUGUST 2020



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabamauniversity.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **DINESH KUMAR A (Reg. No. 37120087)** who underwent the professional training in "**RESTAURANT MANAGEMENT SYSTEM**" under our supervision from Jan 2020 to April 2020.

Internal Guide

Ms.Jithina Jose,M.E(Ph.D)

Head of the Department

Dr. R.SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **DINESH KUMAR A** (Reg No. 37120087) hereby declare that the Professional Training Report on “**RESTAURANT MANAGEMENT SYSTEM**” done by me under the guidance of **Ms. JITHINA JOSE.,M.E(Ph.D)** at **Sathyabama Institute of science and Technology** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Information Technology.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean, School of Computing** and **Dr. R.SUBHASHINI M.E., Ph.D., Head of the Department, Dept. of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.Jithina Jose.,M.E(Ph.D)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

The businesses in restaurants are now growing constantly. At the same time, the need for managing its operations and tasks arises. The best way to optimize these activities is growing the business online as well. Today's generation encourages high-tech services especially over the Internet.

The project "Restaurant Management System" is implemented to reduce the manual work and enhances the accuracy of work in a restaurant.

RESTAURANT MANAGEMENT SYSTEM is Desktop Application to restaurant management. Hence the project is developed proficiently to help restaurant. This system wake to provide service facility to restaurant and owners automate their BILLING OPERATIONS.

This system entirely reduces the unnecessary time waste inside the hotel as well as it reduces unnecessary calculation.

As per the new rule Goods and Service Tax (GST) levy on both AC and non-AC restaurants to 5%. Every restaurant charges 5% GST breakup as 2.5% State GST (SGST) and 2.5 % Central GST (CGST) without any other service tax and any other VAT charges.. We have developed a RESTAURANT MANAGEMENT SYSTEM with inbuilt GST calculation in the bill, as well as it shows the amount in rupees how much GST is applied.

RESTAURANT MANAGEMENT SYSTEM project fully developed in python language which is currently demanded in market using python GUI Tkinter. Python is a programming language that lets you work quickly and integrate systems more efficiently.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	vi
	1 INTRODUCTION	
	1.1 PYTHON LANGUAGE INTRODUCTION	1
	1.2 HISTORY OF PYTHON	2
	1.3 PYTHON FEATURES	2
	1.4 PYTHON GUI's	3
	2 METHODS – PYTHON TKINTER GUI	
	2.1 TKINTER WIDGETS	4
	2.2 GEOMETRY MANAGEMENT	6
	2.3 MODULAR DESIGN	6
	2.4 PYTHON INTELLIJ IDE	8
	3 SYSTEM IMPLEMENTATION	
	3.1 TECHNOLOGIES USED	8
	3.2 LANGUAGE USED	8
	3.3 REQUIREMENTS	9
	3.4 SUMMARY	10
	4 CONCLUSION AND FUTURE WORK	
	4.1 SUMMARY	12
	4.2 FUTURE WORK	12
	REFERENCES	
	APPENDIX	
	A.SOURCE CODE	14
	B.SCREEN SHOTS	24

CHAPTER 1

INTRODUCTION

1.1 PYTHON LANGUAGE INTRODUCTON:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted**

Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive**

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented –**

Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language –**

Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

1.1.1 HISTORY OF PYTHON:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

1.3 PYTHON FEATURES:

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

1.4 PYTHON GUI's :

Tkinter –

Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.

- **wxPython –**

This is an open-source Python interface for wxWindows <http://wxpython.org>.

- **JPython –**

JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine <http://www.jython.org>.

There are many other interfaces available, which you can find them on the net.

CHAPTER 2

METHODS - PYTHON INTELLIJ IDE

2.1 TKINTER WIDGETS:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

Sr.No.	Operator & Description
1	<u>Button</u> The Button widget is used to display buttons in your application.
2	<u>Canvas</u> The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	<u>Checkbutton</u> The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	<u>Entry</u> The Entry widget is used to display a single-line text field for accepting values from a user.
5	<u>Frame</u> The Frame widget is used as a container widget to organize other widgets.
6	<u>Label</u> The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7	<u>Listbox</u> The Listbox widget is used to provide a list of options to a user.
8	<u>Menubutton</u> The Menubutton widget is used to display menus in your application.
9	<u>Menu</u> The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	Message The Message widget is used to display multiline text fields for accepting values from a user.
11	Radiobutton The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	Scale The Scale widget is used to provide a slider widget.
13	Scrollbar The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	Text The Text widget is used to display text in multiple lines.
15	Toplevel The Toplevel widget is used to provide a separate window container.
16	Spinbox The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

17	PanedWindow A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	LabelFrame A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	TkMessageBox This module is used to display message boxes in your applications.

2.2 GEOMETRY MANAGEMENT

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method – This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

2.3 MODULAR DESIGN :

TKINTER PROGRAMMING :



Tkinter is the standard GUI 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.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python

importtkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

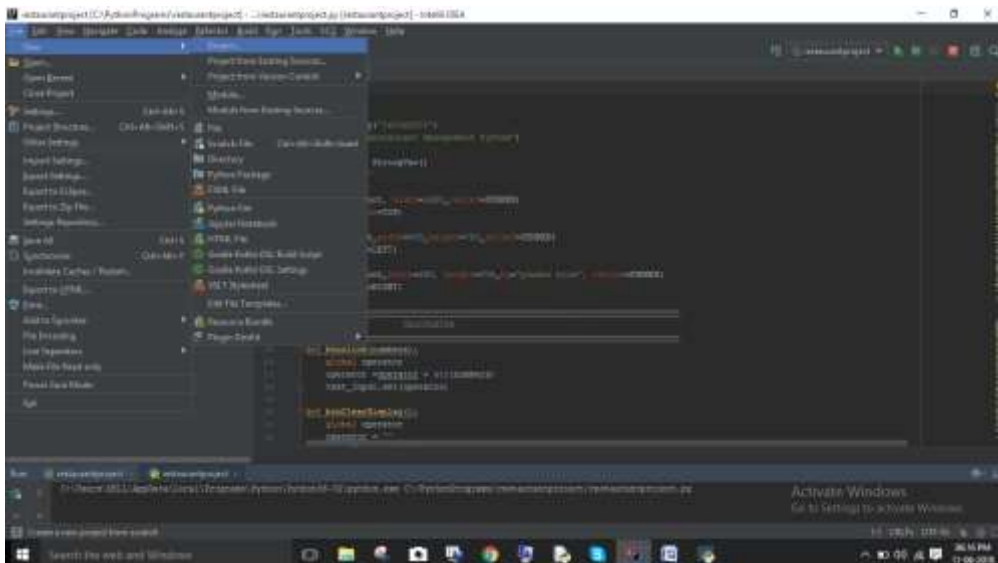
This would create a following window –



2.4 Python IntelliJ IDE :



IntelliJ IDEA is a Java integrated development environment for developing computer software. It is developed by JetBrains, and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition.



CHAPTER 3

SYSTEM IMPLEMENTATION

3.1 TECHNOLOGIES USED:

Python 3.6.

Python Tkinter GUI

IntelliJ IDE

3.2 LANGUAGES USED:

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure.^[1] This feature is also sometimes termed the off-side rule.

3.3 REQUIREMENTS:

	WINDOWS	Mac OS LINUX VERSION	
OS Version	Windows 10/8/7 x64	Mac OS 10.8 or later GNOME or KDE desktop	

RAM	1 GB minimum; 4 GB or more recommended for Android development And commercial production.
Disk space	300 MB hard disk space + at least 1 GB for caches
JDK Version	JDK 1.8 since 2016.
Screen resolution	1024×768 minimum screen resolution

The IDE provides certain features like code completion by analyzing the context, code navigation which allows jumping to a class or declaration in the code directly, code refactoring and options to fix inconsistencies via suggestions.

3.4 SUMMARY:

Python can serve as a scripting language for web applications, e.g, via Apache web server With Web Server Gateway Interface, a standard API has evolved to facilitate these applications.

Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as data mapper to a relational database.

Libraries such as NumPy, SciPy and allow the effective use of Python in scientific computing and provides domain-specific functionality.

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D animation packages such as 3ds Max, Cinema 4D, Light wave, Motion Builder, Soft image and musical notation programs like score writer and capella. It has also been used in several video games and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go. Python is also used in algorithmic trading and quantitative finance. Python can also be implemented in APIs of online brokerages that run on other languages by using wrappers.

Python has been used in artificial intelligence projects. Many operating systems include Python as a standard component. It ships with most Linux distributions, macOS, and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Python is used extensively in the information security industry, including in exploit development.

CHAPTER 4

CONCLUSION

4.1 SUMMARY:

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.

As well as standard desktop integrated development environments (see Wikipedia's "Python IDE" article), there are Web browser-based IDEs; SageMath (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

4.2 FUTURE WORK:

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a userfriendly application and IntelliJ IDE as a platform of carrying codes.

The fast growing use of internet confirms the good future and scope of the proposed project.

Finally I learnt a valuable lifelong lesson about the improvements and working on python .

REFERENCES

www.google.com

www.python.org

www.jetbrains.com

www.wikipedia.com

www.projectlombok.com

APPENDIX

A. SOURCE CODE:

```
from tkinter import*
import random
import time
import datetime

root=Tk()
root.geometry("1600x8000")
root.title("Restaurant Management System")

text_Input = StringVar()
operator = ""

Tops=Frame(root, width=1600,relief=SUNKEN)
Tops.pack(side=TOP)

f1=Frame(root,width=800,height=700,relief=SUNKEN)
f1.pack(side=LEFT)

f2 = Frame(root,width=300, height=700,bg="powder blue", relief=SUNKEN)
f2.pack(side=RIGHT)

#=====
====

#                                CALCULATOR

#=====
====

defbtnclick(numbers):
global operator
```

```
operator =operator + str(numbers)
```

```
text_Input.set(operator)
```

```
defbtnClearDisplay():
```

```
global operator
```

```
    operator = ""
```

```
text_Input.set("")
```

```
defbtnEqualsInput():
```

```
global operator
```

```
sumup= str(eval(operator))
```

```
text_Input.set(sumup)
```

```
    operator = ""
```

```
txtDisplay = Entry(f2,font=('arail', 20, 'bold'), textvariable=text_Input, bd=30,
```

```
insertwidth=4, bg="powder blue", justify='right')
```

```
txtDisplay.grid(columnspan=4)
```

```
btn7=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="7",
```

```
bg="powder blue", command=lambda: btnclick(7))
```

```
btn7.grid(row=2,column=0)
```

```
btn8=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="8",
```

```
bg="powder blue", command=lambda: btnclick(8))
```

```
btn8.grid(row=2,column=1)
```

```
btn9=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="9",
```

```
bg="powder blue", command=lambda: btnclick(9))
```

```
btn9.grid(row=2,column=2)
```

```
Addition=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="+",
```

```
bg="powder blue", command=lambda: btnclick("+"))
```

```
Addition.grid(row=2,column=3)
```

```

btn4=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="4",
bg="powder blue", command=lambda: btnclick(4))
btn4.grid(row=3,column=0)
btn5=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="5",
bg="powder blue", command=lambda: btnclick(5))
btn5.grid(row=3,column=1)
btn6=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="6",
bg="powder blue", command=lambda: btnclick(6))
btn6.grid(row=3,column=2)
Subtraction=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="-",
bg="powder blue", command=lambda: btnclick("-"))
Subtraction.grid(row=3,column=3)

```

```

btn1=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="1",
bg="powder blue", command=lambda: btnclick(1))
btn1.grid(row=4,column=0)
btn2=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="2",
bg="powder blue", command=lambda: btnclick(2))
btn2.grid(row=4,column=1)
btn3=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="3",
bg="powder blue", command=lambda: btnclick(3))
btn3.grid(row=4,column=2)
Multiply=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="*",
bg="powder blue", command=lambda: btnclick("*"))
Multiply.grid(row=4,column=3)

```

```

btn0=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="0",
bg="powder blue", command=lambda: btnclick(0))
btn0.grid(row=5,column=0)
btnClear=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="C",
bg="powder blue", command=btnClearDisplay)
btnClear.grid(row=5,column=1)

```

```

btnEquals=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text "=",
bg="powder blue", command=btnEqualsInput)
btnEquals.grid(row=5,column=2)
Division=Button(f2,padx=16,pady=16, fg="black", font=('arail',20,'bold'),text="/",
bg="powder blue", command=lambda: btnclick("/"))
Division.grid(row=5,column=3)

```

```

#=====
=====
#                               TIME AND HEADING NAME
#=====
=====

```

```

localtime=time.asctime(time.localtime(time.time()))

```

```

lblInfo=Label(Tops,font=('arial',50,'bold'),text="MAHA RESTAURANT ",fg="Steel
Blue",bd=10,anchor='w')
lblInfo.grid(row=0,column=0)

```

```

lblInfo=Label(Tops,font=('arial',20,'bold'),text=localtime,fg="Steel
Blue",bd=10,anchor='w')
lblInfo.grid(row=1,column=0)

```

```

#=====
=====
#                               BILL CALCULATIONS
#=====
=====

```

```

def Ref():

```

```
x=random.randint(10908,500876)
randomRef=str(x)
rand.set(randomRef)
```

```
if (Idly.get()==""):
    Coldly=0
else:
    Coldly=float(Idly.get())
```

```
if (Dosa.get()==""):
    CoDosa=0
else:
    CoDosa=float(Dosa.get())
```

```
if (IceCream.get()==""):
    ColceCream=0
else:
    ColceCream=float(IceCream.get())
```

```
if (Pulav.get()==""):
    CoPulav=0
else:
    CoPulav=float(Pulav.get())
```

```
if (Tea.get()==""):
    CoTea=0
else:
    CoTea=float(Tea.get())
```

```

if (Drinks.get()==""):
    CoD=0
else:
    CoD=float(Drinks.get())

```

```

Costofldly = Coldly * 25
CostofDrinks= CoD * 20
CostofDosa = CoDosa* 25
CostofIceCream = ColceCream * 30
CostPulav = CoPulav* 50
CostTea = CoTea * 5

```

```

Central_GST=
(((Costofldly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+CostTea)*
2.5)/100)

```

```

State_GST=(((Costofldly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav
+CostTea)* 2.5)/100)

```

```

Total_cost =
(Costofldly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+CostTea)

```

```

CostofMeal= "Rs", str('%0.2f' %
(Costofldly+CostofDrinks+CostofDosa+CostofIceCream+CostPulav+CostTea))
C_gst = "Rs", str ('%0.2f' % Central_GST)
S_gst = "Rs", str ('%0.2f' % State_GST)
OverAllCost ="Rs", str ('%0.2f' % (Total_cost+Central_GST+State_GST))

```

```
Sgst.set(S_gst)
Cost.set(CostofMeal)
Cgst.set(C_gst)
Total.set(OverAllCost)
```

```
defqExit():
root.destroy()
```

```
def Reset():
Tea.set("")
Idly.set("")
Dosa.set("")
IceCream.set("")
Pulav.set("")
Drinks.set("")
```

```
rand.set("")
```

```
Total.set("")
Sgst.set("")
Cgst.set("")
Cost.set("")
```

```
#=====
=====
```

```
#          RESTAURANT MENU
```

```
#=====
=====
```

```
Tea=StringVar()
Idly=StringVar()
Dosa=StringVar()
```

IceCream=StringVar()

Pulav=StringVar()

Drinks=StringVar()

rand = StringVar()

Cost=StringVar()

Sgst=StringVar()

Cgst=StringVar()

Total=StringVar()

lblTea= Label(f1, font=('arial', 16, 'bold'),text="Tea",bd=16,anchor="w")

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

lblTea=Entry(f1,

font=('arial',16,'bold'),textvariable=Tea,bd=10,insertwidth=4,bg="white",justify='right')

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

lblDrinks= Label(f1, font=('arial', 16, 'bold'),text="Drinks",bd=16,anchor="w")

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

txtDrinks=Entry(f1,

font=('arial',16,'bold'),textvariable=Drinks,bd=10,insertwidth=4,bg="white",justify='right')

txtDrinks.grid(row=1,column=1)

lblIceCream= Label(f1, font=('arial', 16, 'bold'),text="Ice-Cream",bd=16,anchor="w")

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

lblIceCream=Entry(f1,

font=('arial',16,'bold'),textvariable=IceCream,bd=10,insertwidth=4,bg="white",justify='right')

lblIceCream.grid(row=2,column=1)

```

lblIdly= Label(f1, font=('arial', 16, 'bold'),text="Idly",bd=16,anchor="w")
lblIdly.grid(row=3, column=0)
txtIdly=Entry(f1,
font=('arial',16,'bold'),textvariable=Idly,bd=10,insertwidth=4,bg="white",justify='right
')
txtIdly.grid(row=3,column=1)

```

```

lblDosa= Label(f1, font=('arial', 16, 'bold'),text="Dosa",bd=16,anchor="w")
lblDosa.grid(row=4, column=0)
txtDosa=Entry(f1,
font=('arial',16,'bold'),textvariable=Dosa,bd=10,insertwidth=4,bg="white",justify='right')
txtDosa.grid(row=4,column=1)

```

```

lblPulav= Label(f1, font=('arial', 16, 'bold'),text="Rice-Plate",bd=16,anchor="w")
lblPulav.grid(row=5, column=0)
txtPulav=Entry(f1,
font=('arial',16,'bold'),textvariable=Pulav,bd=10,insertwidth=4,bg="white",justify='right')
txtPulav.grid(row=5,column=1)

```

```

#=====
=====
#          RESTAURANT BILL INFO
#=====
=====

```

```

lblReference= Label(f1, font=('arial', 16,
'bold'),text="Reference",bd=16,anchor="w")
lblReference.grid(row=0, column=2)

```

```
txtReference=Entry(f1,  
font=('arial',16,'bold'),textvariable=rand,bd=10,insertwidth=4,bg="powder  
blue",justify='right')  
txtReference.grid(row=0,column=3)
```

```
lblCost= Label(f1, font=('arial', 16, 'bold'),text="Cost of Meal",bd=16,anchor="w")  
lblCost.grid(row=1, column=2)  
txtCost=Entry(f1,  
font=('arial',16,'bold'),textvariable=Cost,bd=10,insertwidth=4,bg="powder  
blue",justify='right')  
txtCost.grid(row=1,column=3)
```

```
lblSgst= Label(f1, font=('arial', 16, 'bold'),text="SGST",bd=16,anchor="w")  
lblSgst.grid(row=2, column=2)  
txtSgst=Entry(f1,  
font=('arial',16,'bold'),textvariable=Sgst,bd=10,insertwidth=4,bg="powder  
blue",justify='right')  
txtSgst.grid(row=2,column=3)
```

```
lblCgst= Label(f1, font=('arial', 16, 'bold'),text="CGST",bd=16,anchor="w")  
lblCgst.grid(row=3, column=2)  
txtCgst=Entry(f1,  
font=('arial',16,'bold'),textvariable=Cgst,bd=10,insertwidth=4,bg="powder  
blue",justify='right')  
txtCgst.grid(row=3,column=3)
```

```
lblTotalCost= Label(f1, font=('arial', 16, 'bold'),text="Total Cost",bd=16,anchor="w")  
lblTotalCost.grid(row=4, column=2)  
txtTotalCost=Entry(f1,  
font=('arial',16,'bold'),textvariable=Total,bd=10,insertwidth=4,bg="powder  
blue",justify='right')
```

```
txtTotalCost.grid(row=4,column=3)
```

```
#=====
=====
```

```
#          BUTTONS
```

```
#=====
=====
```

```
btnTotal=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=
10,text="Total",bg="powder blue",command=Ref).grid(row=7,column=1)
```

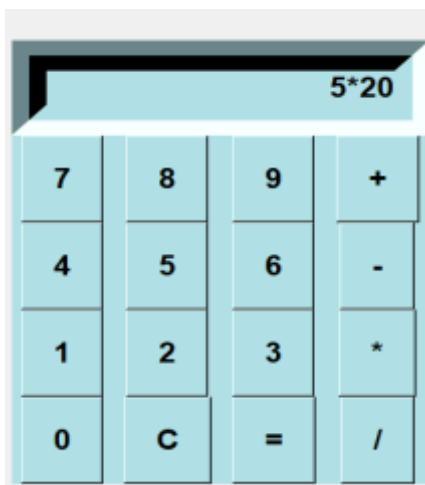
```
btnReset=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width
=10,text="Reset",bg="powder blue",command=Reset).grid(row=7,column=2)
```

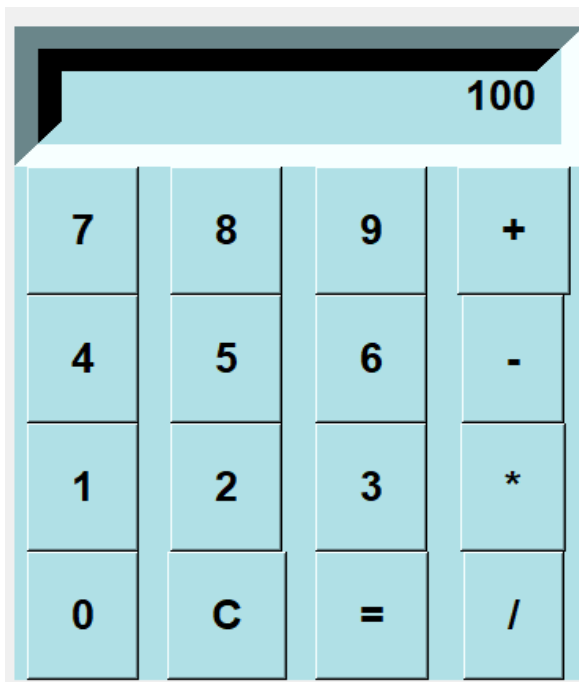
```
btnExit=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=1
0,text="Exit",bg="powder blue",command=qExit).grid(row=7,column=3)
```

```
root.mainloop()
```

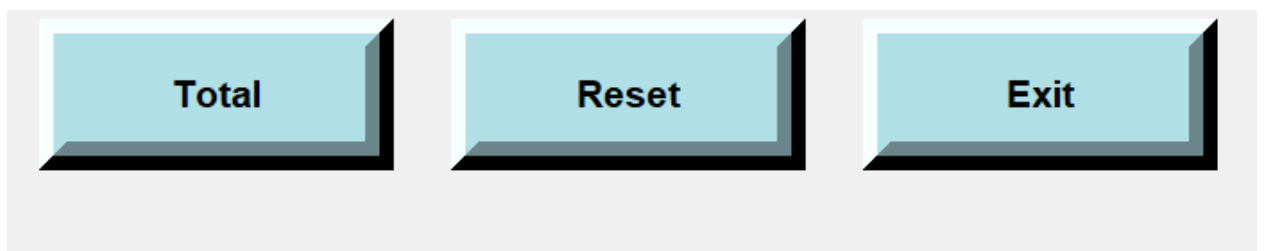
B. SCREENSHOTS

I have added a calculator for instant calculation -





Buttons –



↑
Total button
gives the
calculation of Bill

↑
Reset button
resets all the
values

↑
Exit button
closes the
window

We need to insert number of menu items taken by customer ,after inserting no. of items.

Tea	1
Drinks	2
Ice-Cream	1
Idly	1
Dosa	0
Rice-Plate	2

Generated Bill

/===== **BILL NO**

/===== **COST OF MEAL**

/===== **TOTAL SGST**

/===== **TOTAL CGST**

/===== **TOTAL COST with**

GST