

A PROJECT REPORT ON *CalculatorVerse* **USING PYTHON**

Submitted to:
Chitkara University ,Punjab

for
Bachelor of Engineering program
Computer Science and Engineering

by

Tannishtha Jain
of Group-29
Roll Number – 2210990903



Under the supervision of
Ms. Shilpi Garg
Associate Professor

INDEX

S.No	Topic
1	Abstract
2	Introduction
2	Advantages and Features
3	Project Coding
4	Project
5	Conclusion

Abstract

This report introduces the process of creating an easy to use multi calculator and converter used by instructors and students. The implementation uses a tool called Tkinter which is the only framework that is built into the Python standard library. The major part of this report will introduce how to use Tkinter to create a user interface and inner logic to handle user request by going through the implementation process.

Introduction

This project is built using Tkinter and has four major components each of which has different functionality but similar architecture. In the project report I will demonstrate details of using Tkinter to build major component of this system. Also the technique and process which is showed here can be applied to build the other three components in this project.

The interface is very User-friendly.

Python is the language used to build the Tkinter framework. It is a dynamic scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

Front End Description

Language:- Python

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 system more efficiently. Python is dynamically typed and garbage

collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is as a "batteries included" language due to its comprehensive standard library.

Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's *de facto* standard GUI. It is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. name *Tkinter* comes from *Tk interface*. Tkinter was written by Fredrik Lundh.

Significance

The new system will integrate all existing functionalities of various calculating systems as well as other new functions. Thus the technical team only needs to maintain one system instead of many. The scalability of the new system will allow them to add more features to the system in future. Students and professors only need to log into one system to accomplish whatever they need. This will be more convenient and significantly improve their efficiency.

Project overview

1)Importing the necessary modules

To use the Tkinter we need to import the Tkinter module. We are also going to import the functions from math module.

2)Making a window for our calculator

Drafting the window for our calculator which will accommodate the buttons.

3)Designing the buttons

To design the buttons for our calculator and put them on our application window.

In this calculator program in python, the “Entry” function helps in making a text input field and we use .grid() method to define the positioning associated with the button or input field. We use the button method to display a button on our application window.

- **root** – the name with which we refer to our window
- **text** – text to be displayed on the button
- **row** – row index of the grid
- **column** – column index of the grid
- **columnspan** – spans or combines the number of columns
- **sticky** – If the resulting cell is larger than the widget then sticky defines how to expand the widget. The combination of constants used S, N, E, and W, or NW, NE, SW, and SE are analogous to the directions in compass. N+E+W+S means that the widget should be expanded in all directions

To connect the **digit button** to the **get_variable()** function, we use the “command” parameter.

4) Mapping the buttons to their functionalities

Mapping the digits

The **get_variable()** function receives the digit as parameter. The digit is inserted to the input field with **.insert()** method with parameters ‘i’ and ‘num’.

- **i** – the position to insert the digit
- **num** – the digit

Mapping the operator buttons

The `get_operation()` function receives the operator as a parameter which is then inserted to the text field at the `ith` position of python calculator.

Mapping the AC button

We use the `.delete()` method to remove characters in the text field. It accepts the start and end position as the parameter.

- 0 – start position
- **END** – end position

Mapping the '=' button

We fetch the string present on the input field using the `.get()` method. We now use the parse module to scan the string with the help of `.expr()` method which accepts the string as a parameter. We basically leave it to the parser to build an abstract syntax tree of the string which is evaluated using the `eval()` function.

Once we get the result, we push it to the input field after clearing it.

Source Code

```
1 from tkinter import Tk,Radiobutton,Button,Label,StringVar,IntVar,Entry
2
3 class TipCalculator():
4     def __init__(self):
5         window=Tk()
6         window.title("Tip Calculator App")
7         window.configure(background="sky blue")
8         window.geometry("350x250")
9         window.resizable(width=False,height=False)
10
11         self.meal_cost=StringVar()
12         self.tip_percent=IntVar()
13         self.tip= StringVar()
14         self.total_cost=StringVar()
15
16         tip_percents=Label(window,text="Tip Percentages",bg="purple",fg="white")
17         tip_percents.grid(column=0,row=0,padx=15,pady=14)
18
19         bill_amount=Label(window,text="Bill Amount",bg="black",fg="white")
20         bill_amount.grid(column=1,row=0,padx=15)
21
22         bill_amount_entry=Entry(window,textvariable=self.meal_cost,width=14)
23         bill_amount_entry.grid(column=2,row=0,padx=15)
24
25         one_percent_tip= Radiobutton(window,text="1%",variable=self.tip_percent,value=1)
26         one_percent_tip.grid(column=0,row=1)
27         five_percent_tip= Radiobutton(window,text="5%",variable=self.tip_percent,value=5)
28         five_percent_tip.grid(column=0,row=2)
29         ten_percent_tip= Radiobutton(window,text="10%",variable=self.tip_percent,value=10)
30         ten_percent_tip.grid(column=0,row=3)
31         fifteen_percent_tip= Radiobutton(window,text="15%",variable=self.tip_percent,value=15)
32         fifteen_percent_tip.grid(column=0,row=4)
33         twenty_percent_tip= Radiobutton(window,text="20%",variable=self.tip_percent,value=20)
34         twenty_percent_tip.grid(column=0,row=5)
35         twentyfive_percent_tip= Radiobutton(window,text="25%",variable=self.tip_percent,value=25)
36         twentyfive_percent_tip.grid(column=0,row=6)
```

38

39

```
tip_amount_lbl=Label(window,text="Tip Amount",bg="brown",fg="white")
```

40

```
tip_amount_lbl.grid(column=1,row=2,padx=15)
```

41

```
tip_amount_entry=Entry(window,textvariable=self.tip,width=14)
```

42

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

43

44

```
bill_total_lbl = Label (window, text ="Bill Total", bg="blue", fg="white")
```

45

```
bill_total_lbl.grid(column=1, row=4, padx=15)
```

46

```
bill_total_entry=Entry(window,textvariable=self.total_cost,width =14)
```

47

```
bill_total_entry.grid(column=2, row=4)
```

48

49

```
calculate_btn = Button(window, text ="Calculate", bg="green", fg="white",command=self.calculate)
```

50

```
calculate_btn.grid(column=1, row=7, padx=15)
```

51

52

```
clear_btn= Button(window, text="Clear", bg="black", fg="white",command=self.clear)
```

53

```
clear_btn.grid(column=2, row=7)
```

54

55

```
window.mainloop()
```

56

57

```
def calculate(self):
```

58

```
    pre_tip=float(self.meal_cost.get())
```

59

```
    percentage=self.tip_percent.get()/100
```

60

```
    tip_amount_entry=pre_tip*percentage
```

61

```
    self.tip.set(tip_amount_entry)
```

62

```
    final_bill=pre_tip+tip_amount_entry
```

63

```
    self.total_cost.set(final_bill)
```

64

```
def clear(self):
```

65

```
    self.total_cost.set("")
```

66

```
    self.meal_cost.set("")
```

67

```
    self.tip.set("")
```

68

```
Tipcalculator()
```

```

class LoanCalculator:
    def __init__(self):
        window=Tk()
        window.title("Loan Calculator")
        window.configure(background="light pink")

        Label(window,font="Helvetica 12 bold",bg="light pink",text="Annual Interest Rate").grid(row=1,column=1,sticky=W)
        Label(window,font="Helvetica 12 bold",bg="light pink",text="Number of Years").grid(row=2,column=1,sticky=W)
        Label(window,font="Helvetica 12 bold",bg="light pink",text="Loan Amount").grid(row=3,column=1,sticky=W)
        Label(window,font="Helvetica 12 bold",bg="light pink",text="Monthly Payment").grid(row=4,column=1,sticky=W)
        Label(window,font="Helvetica 12 bold",bg="light pink",text="Total Payment").grid(row=5,column=1,sticky=W)

        self.annualInterestRateVar= StringVar()
        Entry(window, textvariable=self.annualInterestRateVar,justify=RIGHT).grid(row=1, column=2)
        self.numberofYearsVar= StringVar()
        Entry(window, textvariable=self.numberofYearsVar,justify=RIGHT).grid(row=2, column=2)
        self.loanAmountVar= StringVar()
        Entry(window, textvariable=self.loanAmountVar,justify=RIGHT).grid(row=3, column=2)
        self.monthlyPaymentVar= StringVar()
        lblMonthlyPayment=Label(window,font="Helvetica 12 bold",bg="light pink", textvariable=self.monthlyPaymentVar).grid(row=4,column=2,sticky=E)
        self.totalPaymentVar=StringVar()
        lblTotalPayment=Label(window,font="Helvetica 12 bold",bg="light pink", textvariable=self.totalPaymentVar).grid(row=5,column=2,sticky=E)

        btComputePayment=Button(window, text="Compute Payment",bg="red",fg="white",font="Helvetica 14 bold", command=self.computePayment).grid(row=6,column=2,sticky=E)
        btClear=Button(window, text="Clear",bg="light green",fg="white",font="Helvetica 14 bold", command=self.delete_all).grid(row=6,column=8,padx=20,pady=20,sticky=E)

        window.mainloop()

    def computePayment(self):
        monthlyPayment=self.getMonthlyPayment(
            float(self.loanAmountVar.get()),
            float(self.annualInterestRateVar.get()) / 1200,
            int(self.numberofYearsVar.get()))

        self.monthlyPaymentVar.set(format(monthlyPayment,'10.2f'))
        totalPayment=float(self.monthlyPaymentVar.get()) *12 \
            *int(self.numberofYearsVar.get())
        self.totalPaymentVar.set(format(totalPayment,"10.2f"))

    def getMonthlyPayment(self,loanAmount,monthlyInterestRate,numberofYears):
        monthlyPayment=loanAmount*monthlyInterestRate/(1-1/(1+monthlyInterestRate)**(numberofYears*12))
        return monthlyPayment

    def delete_all(self):
        self.monthlyPaymentVar.set("")
        self.loanAmountVar.set("")
        self.annualInterestRateVar.set("")
        self.numberofYearsVar.set("")
        self.totalPaymentVar.set("")

LoanCalculator()

```

```

1 from tkinter import Tk, Button, Label, DoubleVar, Entry
2
3 window=Tk()
4 window.title('Feet to Meter Conversion App')
5 window.configure(background="light blue")
6 window.geometry("320x220")
7 window.resizable(width=False, height=False)
8
9 def convert():
10     value=float(ft_entry.get())
11     meter=value*0.3048
12     mt_value.set("%.4f" % meter)
13
14 def clear():
15     ft_value.set("")
16     mt_value.set("")
17
18 ft_lbl=Label(window,text="Feet",background="purple",foreground="white",width=14)
19 ft_lbl.grid(column=0,row=0,padx=15,pady=15)
20
21 ft_value=DoubleVar()
22 ft_entry=Entry(window,textvariable=ft_value,width=14)
23 ft_entry.grid(column=1,row=0)
24 ft_entry.delete(0,"end")
25
26 mt_lbl=Label(window,text="Meter",bg='brown',fg="white",width=14)
27 mt_lbl.grid(column=0,row=1)
28
29 mt_value=DoubleVar()
30 mt_entry=Entry(window,textvariable=mt_value,width=14)
31 mt_entry.grid(column=1,row=1,pady=30)
32 mt_entry.delete(0,"end")
33
34 convert_btn=Button(window,text="Convert", bg="blue", fg="white",width=14,command=convert)
35 convert_btn.grid(column=0,row=3,padx=15)
36
37 clear_btn=Button(window,text="Clear",bg="black",fg="white",width=14,command=clear)
38 clear_btn.grid(column=1,row=3,padx=15)
39
40 window.mainloop()
41

```

```

1 from tkinter import *
2 class Application(Frame):
3     def __init__(self, master):
4
5         super(Application, self).__init__(master)
6         self.task=""
7         self.UserIn=StringVar()
8         self.grid()
9         self.create_widgets()
10    def create_widgets(self):
11
12        self.user_input=Entry(self,bg="#5BC8AC", bd=29,
13            insertwidth=4,width = 24,
14            font=("Helvetica", 20,"bold"), textvariable=self.UserIn,justify=RIGHT)
15        self.user_input.grid(columnspan=4)
16
17        self.user_input.insert(0,"0")
18
19        self.button1=Button(self,bg="#98D8C6", bd=12,
20            text="7",padx=33,pady=25, font=("Helvetica", 20, "bold"),
21            command=lambda: self.buttonClick(7))
22        self.button1.grid(row=2, column= 0, sticky=W)
23
24        self.button2=Button(self,bg="#98D8C6", bd=12,
25            text="8",padx=35,pady=25, font=("Helvetica", 20, "bold"),
26            command=lambda: self.buttonClick(8))
27        self.button2.grid(row=2, column= 1, sticky=W)
28
29        self.button3=Button(self,bg="#98D8C6", bd=12,
30            text="9",padx=33,pady=25, font=("Helvetica", 20, "bold"),
31            command=lambda: self.buttonClick(9))
32        self.button3.grid(row=2, column= 2, sticky=W)
33
34        self.button4=Button(self,bg="#98D8C6", bd=12,
35            text="4",padx=33,pady=25, font=("Helvetica", 20, "bold"),
36            command=lambda: self.buttonClick(4))
37        self.button4.grid(row=3, column= 0, sticky=W)
38
39        self.button5=Button(self,bg="#98D8C6", bd=12,
40            text="5",padx=35,pady=25, font=("Helvetica", 20, "bold"),
41            command=lambda: self.buttonClick(5))
42        self.button5.grid(row=3, column= 1, sticky=W)
43
44        self.button6=Button(self,bg="#98D8C6", bd=12,
45            text="6",padx=33,pady=25, font=("Helvetica", 20, "bold"),
46            command=lambda: self.buttonClick(6))
47        self.button6.grid(row=3, column= 2, sticky=W)
48
49        self.button7=Button(self,bg="#98D8C6", bd=12,
50            text="1",padx=33,pady=25, font=("Helvetica", 20, "bold"),
51            command=lambda: self.buttonClick(1))
52        self.button7.grid(row=4, column= 0, sticky=W)
53
54        self.button8=Button(self,bg="#98D8C6", bd=12,
55            text="2",padx=35,pady=25, font=("Helvetica", 20, "bold"),
56            command=lambda: self.buttonClick(2))
57        self.button8.grid(row=4, column= 1, sticky=W)
58
59        self.button9=Button(self,bg="#98D8C6", bd=12,
60            text="3",padx=33,pady=25, font=("Helvetica", 20, "bold"),
61            command=lambda: self.buttonClick(3))
62        self.button9.grid(row=4, column= 2, sticky=W)
63
64        self.button0=Button(self,bg="#98D8C6", bd=12,
65            text="0",padx=33,pady=25, font=("Helvetica", 20, "bold"),
66            command=lambda: self.buttonClick(0))

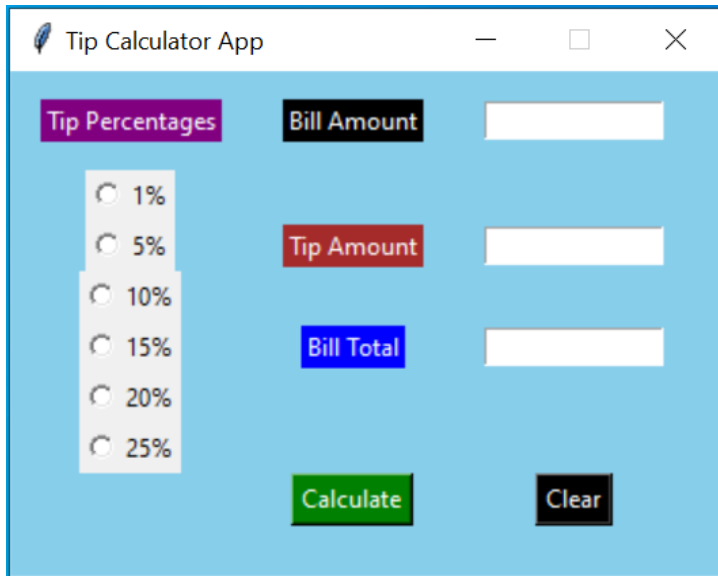
```

```

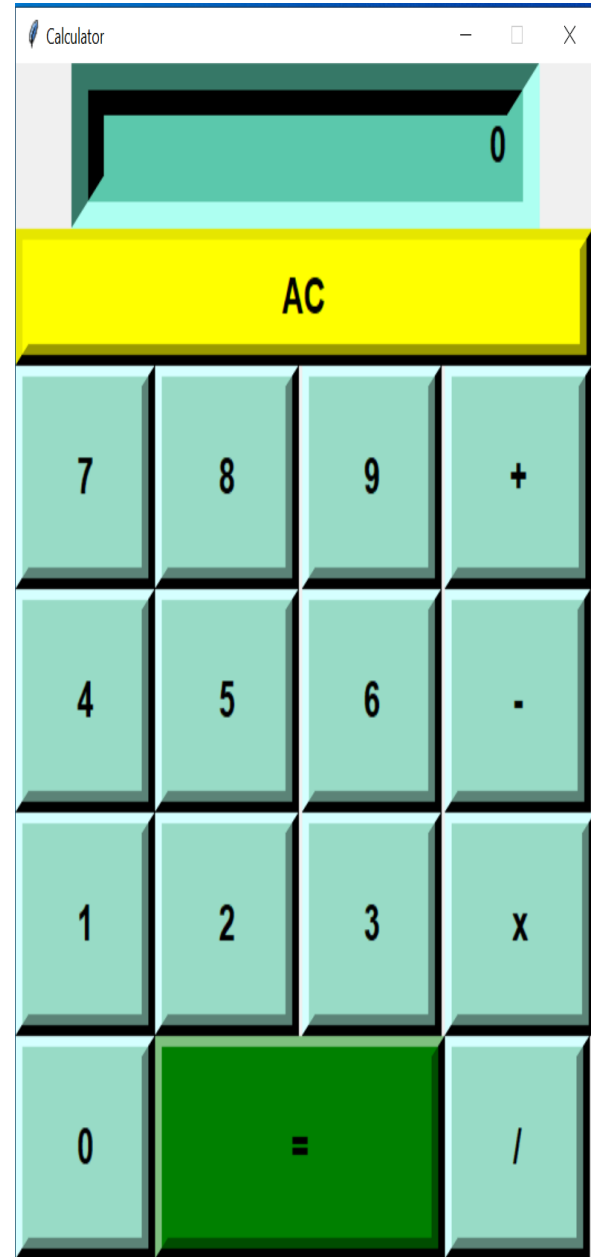
64         self.button0=Button(self,bg="#98DBC6", bd=12,
65         text="0",padx=33,pady=25, font=("Helvetica", 20, "bold"),
66         command=lambda: self.buttonClick(0))
67         self.button0.grid(row=5, column= 0, sticky=W)
68
69         self.Addbutton=Button(self,bg="#98DBC6", bd=12,
70         text="+",padx=36,pady=25, font=("Helvetica", 20, "bold"),
71         command=lambda: self.buttonClick("+"))
72         self.Addbutton.grid(row=2, column= 3, sticky=W)
73
74         self.Subbutton=Button(self,bg="#98DBC6", bd=12,
75         text="-",padx=39,pady=25, font=("Helvetica", 20, "bold"),
76         command=lambda: self.buttonClick("-"))
77         self.Subbutton.grid(row=3, column= 3, sticky=W)
78
79         self.Multbutton=Button(self,bg="#98DBC6", bd=12,
80         text="x",padx=38,pady=25, font=("Helvetica", 20, "bold"),
81         command=lambda: self.buttonClick("*"))
82         self.Multbutton.grid(row=4, column= 3, sticky=W)
83
84         self.Divbutton=Button(self,bg="#98DBC6", bd=12,
85         text="/",padx=39,pady=25, font=("Helvetica", 20, "bold"),
86         command=lambda: self.buttonClick("/"))
87         self.Divbutton.grid(row=5, column= 3, sticky=W)
88
89         self.Equalbutton=Button(self,bg="green", bd=12,
90         text="=",padx=100,pady=25, font=("Helvetica", 20, "bold"),
91         command=self.CalculateTask)
92         self.Equalbutton.grid(row=5, column= 1, sticky=W,columnspan=2)
93
94         self.Clearbutton=Button(self,bg="yellow", bd=12,
95         text="AC",width=28,padx=7, font=("Helvetica", 20, "bold"),
96         command=self.ClearDisplay)
97         self.Clearbutton.grid(row=1,columnspan=4, sticky=W)
98
99     def buttonClick(self, number):
100         self.task=str(self.task)+str(number)
101         self.UserIn.set(self.task)
102
103     def CalculateTask(self):
104         self.data=self.user_input.get()
105         try:
106             self.answer= eval(self.data)
107             self.displayText(self.answer)
108             self.task=self.answer
109         except SyntaxError as e:
110             self.displayText("Invalid Syntax!")
111             self.task= ""
112
113     def displayText(self, value):
114         self.user_input.delete(0, END)
115         self.user_input.insert(0, value)
116
117     def ClearDisplay(self):
118         self.task=""
119         self.user_input.delete(0,END)
120         self.user_input.insert(0,"0")
121
122 calculator=Tk()
123 calculator.title("Calculator")
124
125 calculator.resizable(width=False,height=False)
126 app=Application(calculator)
127 calculator.mainloop()
128

```

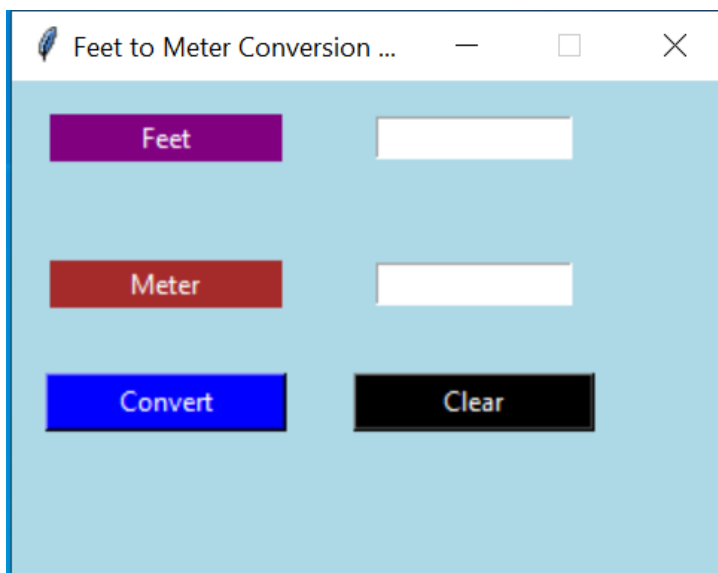
Snapshots of Project



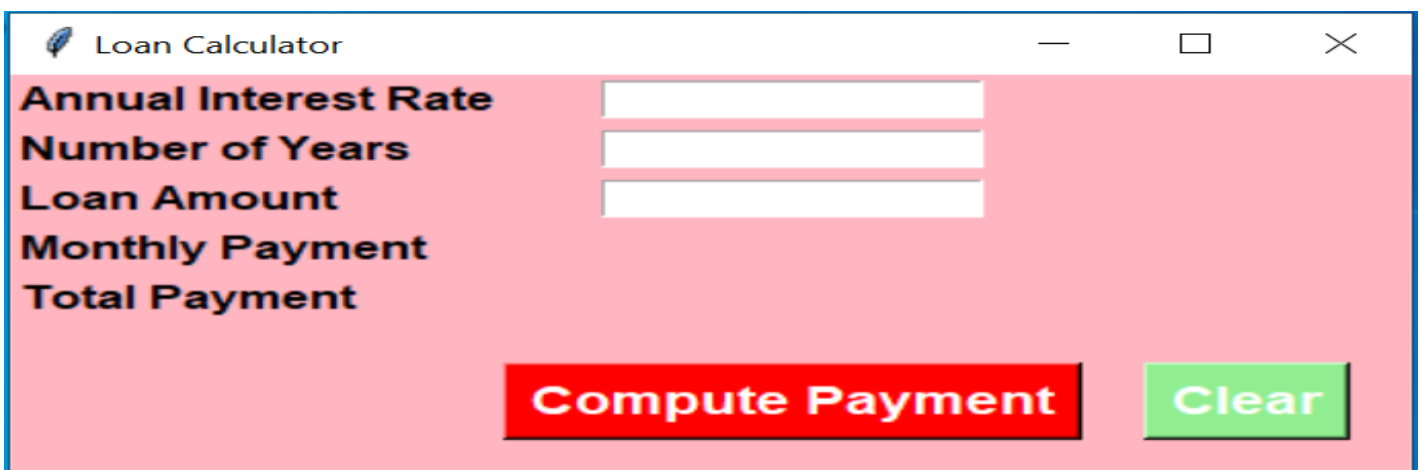
The screenshot shows a window titled "Tip Calculator App". On the left, there is a "Tip Percentages" section with a vertical list of radio buttons for 1%, 5%, 10%, 15%, 20%, and 25%. To the right of this list are three input fields: "Bill Amount", "Tip Amount", and "Bill Total". At the bottom right, there are two buttons: a green "Calculate" button and a black "Clear" button.



The screenshot shows a standard calculator application window titled "Calculator". It features a display at the top showing the number "0". Below the display is a yellow "AC" (All Clear) button. The main area contains a grid of buttons for digits 0-9, decimal point, and arithmetic operators (+, -, x, /). The equals button (=) is highlighted in green.



The screenshot shows a window titled "Feet to Meter Conversion ...". It has two input fields: "Feet" and "Meter". Below the "Feet" field is a blue "Convert" button, and below the "Meter" field is a black "Clear" button.



The screenshot shows a window titled "Loan Calculator". It has four input fields for "Annual Interest Rate", "Number of Years", "Loan Amount", and "Monthly Payment". Below these fields are two buttons: a red "Compute Payment" button and a green "Clear" button.

Future Scope and Enhancement

The scalability of this new system will allow to add more features to the system in future.

- Addition of more features such as graphing and scientific calculator, etc.
- To add a currency converter that calculates at runtime.
- To add health based calculating systems such as BMI calculator, period tracker, etc.
- To include a calculator catering to matrix and determinant calculations.
- To modernize the interface.

Conclusion

We developed a multi calculator application using Tkinter and various widgets of Tkinter about which we have covered in this project report.

Tkinter gives us a simple and reliable way to create the calculating system. It provides powerful functionalities and concise syntax to help programmers deal with the the web page and the inner logic. The experience of developing various components in the system also helped me learning a lot about user interface with Tkinter. Within the Tkinter framework, we have successfully accomplished the requirements of the system. Once this system passes the testing phase, it can be used to serve students and instructors and substitute several systems currently in service. It will make the work for instructors and students to manage their needs much easier. It also can simplify the operations for students with step wise explanation all in one system. In short, this system will bring great user experience to both instructors and students. The flexibility of Tkinter would provide an easy way to add new features into the system.