CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

PROJECT REPORT ON

“Scientific Calculator”

Submitted By:

Ayush Kumar Thakur, UID – 24MCA20322

Under The Guidance of:

Mrs. Deepali Saini

October, 2024

University Institute of Computing

Chandigarh University,

Mohali, Punjab

CERTIFICATE

This is to certify that Ayush Kumar Thakur(UID- 24MCA20322) have successfully completed the project title **"Scientific Calculator"** at University Institute of Computing under my supervision and guidance in the fulfilment of requirements of fourth semester, **Master of Computer Application.** Of Chandigarh University, Mohali, Punjab.

Dr. Abdullah

Head of the Department

University Institute of Computing

Mrs. Deepali Saini

Project Guide Supervisor

University Institute of Computing

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **Mrs. Deppali Saini** under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by **Dr. Abdullah (H.O.D, University Institute of Computing)** who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to Dr. Manisha Malhotra, Additional Director, University Institute of Technology, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication.

Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Date: 30.10.2024

Place: Chandigarh University, Mohali, Punjab

ABSTRACT

A **scientific calculator** is a type of electronic calculator, usually but not always handheld, designed to calculate problems in science, engineering, and mathematics. They have completely replaced slide rules in traditional applications, and are widely used in both education and professional. The python calculator was implemented using tkinter to make the calculation of mathematical functions easier. The application consists of scientific and standard functions. The standard is used to solve scientific notation type math functions like sin, cos, tan, log etc.

Introduction

A scientific calculator developed using Python is a powerful tool designed to perform a wide range of mathematical operations beyond basic arithmetic. This project typically includes functionalities such as trigonometric, logarithmic, and exponential functions, which are essential for scientific and engineering calculations. Python, with its rich libraries like `math` and `tkinter` for GUI, allows for a versatile and user-friendly calculator interface. By building this project, users can gain hands-on experience with Python programming, learn to handle complex mathematical computations, and understand how to create an interactive application. This calculator can serve as a foundation for further customization and learning in Python development.

Report Index

1.Python

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.

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.

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-tomaintain.
- **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 bytecode 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.

2.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 −

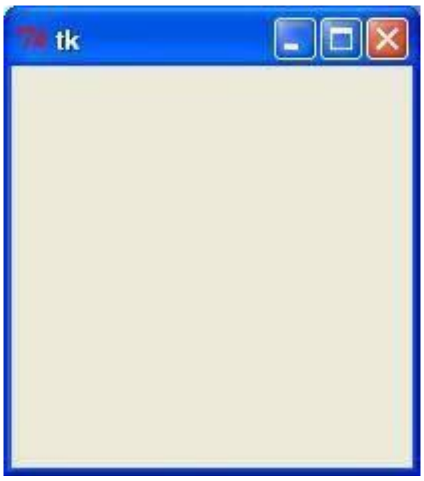
- Import the *Tkinter* module.
- Create the GUI application main window.

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

This would create a following window −



Tkinter Widgets

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

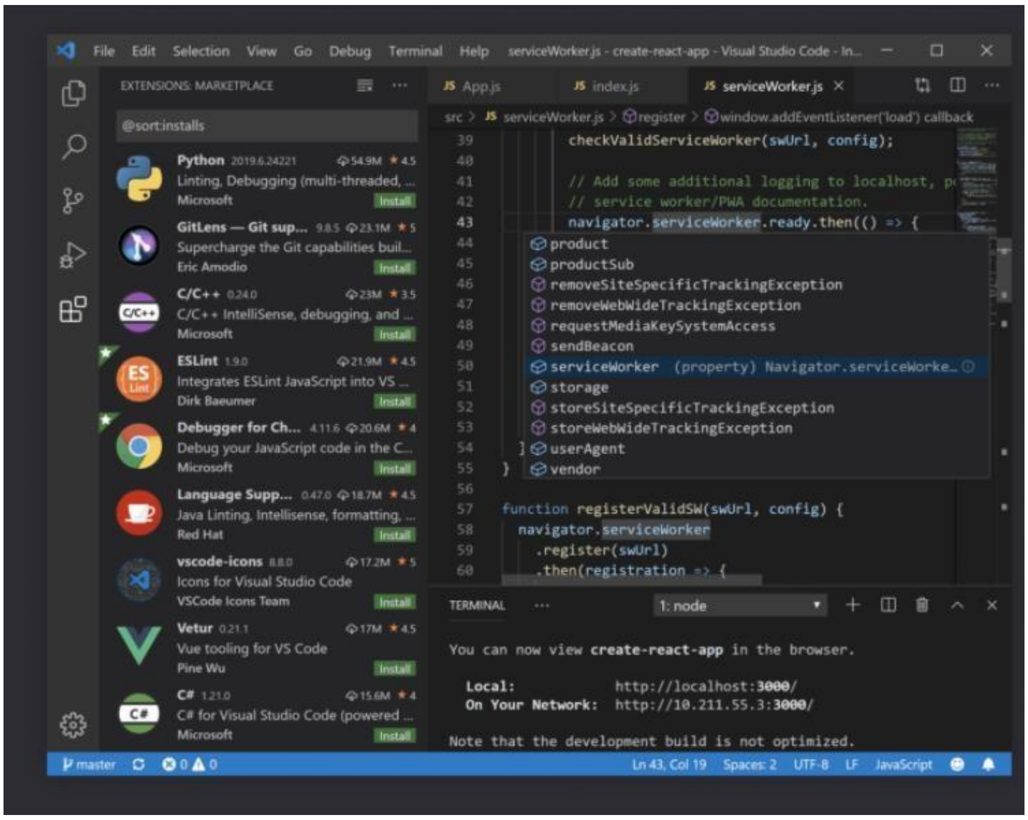
There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

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

3.Visual Studio Code

Visual Studio Code is a free source code editor, made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The python extension in Visual Studio Code makes it an excellent video editor.


File Edit Selection View Go Debug Terminal Help
serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
@sort:installs
Python
GitLens — Git sup...
C/C++
ESLint
Debugger for Ch...
Language Supp...
vscode-icons
Vetur
C#
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback
product
productSub
removeSiteSpecificTrackingException
removeWebWideTrackingException
requestMediaKeySystemAccess
sendBeacon
serviceWorker (property) Navigator.serviceWorke...
storage
storeSiteSpecificTrackingException
storeWebWideTrackingException
userAgent
vendor
TERMINAL
1: node
master
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript

```
        checkValidServiceWorker(swUrl, config);

        // Add some additional logging to localhost, p
        // service worker/PWA documentation.
        navigator.serviceWorker.ready.then(() => {
      }
  }

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

Fig1: Visual Studio Code Platform

4.Source Code

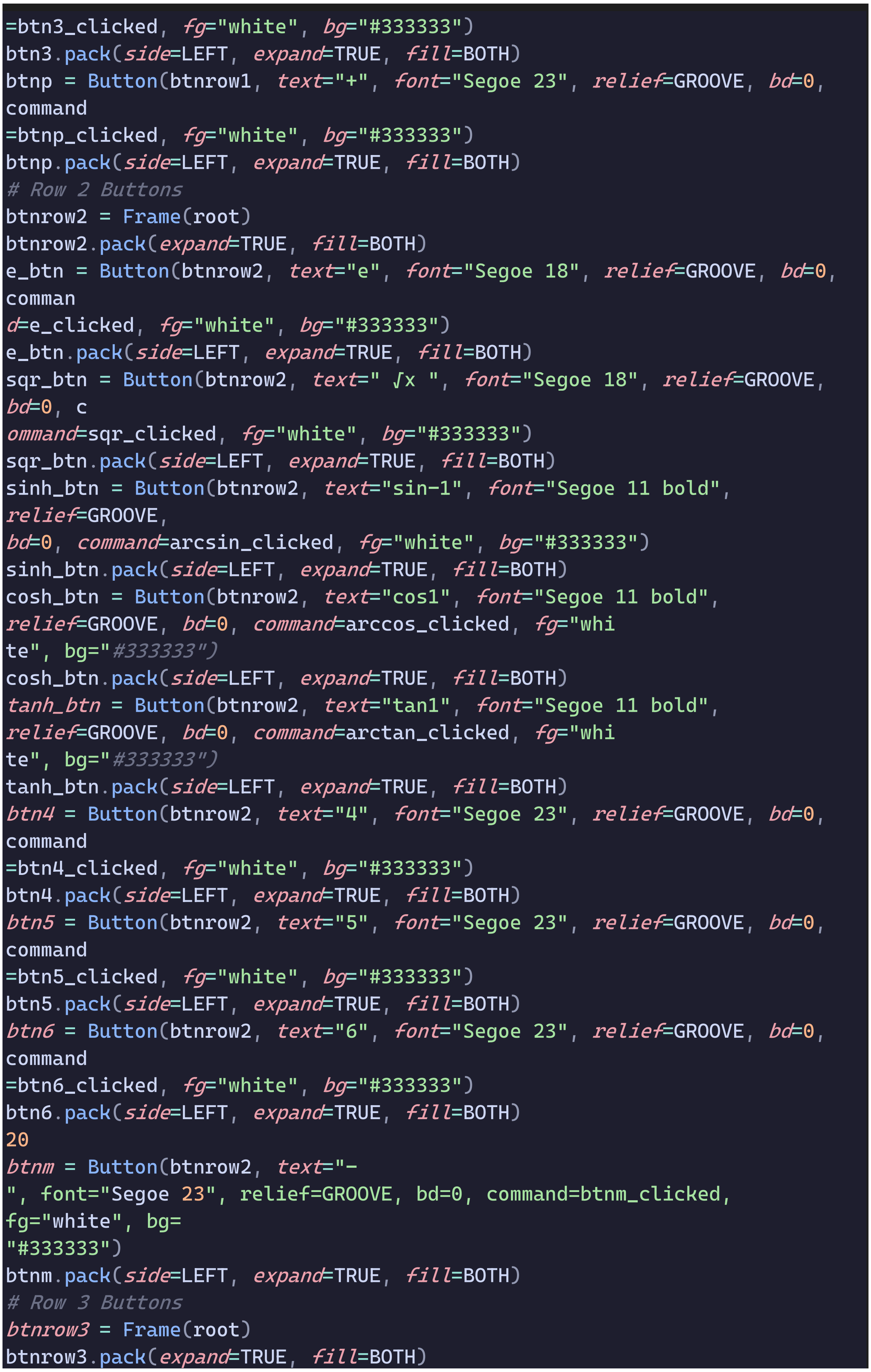
The code for the, Scientific Calculator is as follows:

```
   from tkinter import *
import
tkinter.messagebox
import math


root = Tk() root.geometry("650x400+300+300")
 root.title("Scientific Calculator by
Pramoth")
 switch =
None

# Button on press
 def btn1_clicked():
if disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '1')


def btn2_clicked():
    if disp.get() == '0':
disp.delete(0, END)     pos =
len(disp.get())
disp.insert(pos, '2')


def btn3_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '3')


def btn4_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '4')
```

```
def btn5_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '5')


def btn6_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '6')


def btn7_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '7')


def btn8_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '8')


def btn9_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '9')


def btn0_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
disp.insert(pos, '0')


def key_event(*args):
if disp.get() == '0':
disp.delete(0, END)
 def
btnp_clicked():
    pos = len(disp.get())
disp.insert(pos, '+')


def btnm_clicked():
    pos = len(disp.get())
disp.insert(pos, '-')


def btnml_clicked():
    pos = len(disp.get())
disp.insert(pos, '*')


def btnd_clicked():
    pos = len(disp.get())
disp.insert(pos, '/')


def btnc_clicked(*args):
disp.delete(0, END)
disp.insert(0, '0')


def sin_clicked():
try:
        ans = float(disp.get())
if switch is True:
            ans = math.sin(math.radians(ans))
else:
            ans = math.sin(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def cos_clicked():
    try:
        ans = float(disp.get())
if switch is True:
            ans = math.cos(math.radians(ans))
else:
            ans = math.cos(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def tan_clicked():
try:
        ans = float(disp.get())
if switch is True:
            ans = math.tan(math.radians(ans))
else:
            ans = math.tan(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def arcsin_clicked():
try:
        ans = float(disp.get())
if switch is True:
            ans = math.degrees(math.asin(ans))
else:
            ans = math.asin(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def arccos_clicked():
try:
        ans = float(disp.get())
if switch is True:
            ans = math.degrees(math.acos(ans))
else:
            ans = math.acos(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def arctan_clicked():
    try:
        ans = float(disp.get())
        if switch is True:
            ans = math.degrees(math.atan(ans))
else:
            ans = math.atan(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def pow_clicked():
    pos = len(disp.get())
disp.insert(pos, '**')


def round_clicked():
    try:
        ans = float(disp.get())
ans = round(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def logarithm_clicked():
try:
        ans = float(disp.get())
ans = math.log10(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def fact_clicked():
try:
        ans = float(disp.get())
ans = math.factorial(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")
```

```
def sqr_clicked():
try:
        ans = float(disp.get())
ans = math.sqrt(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def dot_clicked():
    pos = len(disp.get())
disp.insert(pos, '.')


def pi_clicked():
    if disp.get() == '0':
disp.delete(0, END)     pos =
len(disp.get())
    disp.insert(pos, str(math.pi))


def e_clicked():     if
disp.get() == '0':
disp.delete(0, END)
pos = len(disp.get())
    disp.insert(pos, str(math.e))


def bl_clicked():
    pos = len(disp.get())
disp.insert(pos, '(')


def br_clicked():
    pos = len(disp.get())
disp.insert(pos, ')')


def del_clicked():
    pos = len(disp.get())
display = str(disp.get())
if display == '':
        disp.insert(0, '0')
elif display == ' ':
disp.insert(0, '0')     elif
display == '0':
        pass
    else:
        disp.delete(0, END)
disp.insert(0, display[0:pos-1])


def conv_clicked():     global
switch     if switch is None:
switch = True
conv_btn['text'] = "Deg"
else:
        switch = None
conv_btn['text'] = "Rad"


def ln_clicked():
try:
        ans = float(disp.get())
ans = math.log(ans)
disp.delete(0, END)
disp.insert(0, str(ans))     except
Exception:
        tkinter.messagebox.showerror("Value Error", "Check your values and ope
rators")


def mod_clicked():
    pos = len(disp.get())
disp.insert(pos, '%')


def btneq_clicked(*args):
try:
        ans = disp.get()
ans = eval(ans)
disp.delete(0, END)
disp.insert(0, ans)

except:
        tkinter.messagebox.showerror("Value Error", "Check your values and
ope rators")

# Label
data = StringVar()


disp = Entry(root, font="Verdana 20", fg="black", bg="mistyrose", bd=0,
justif y=RIGHT, insertbackground="#abbab1", cursor="arrow")
```

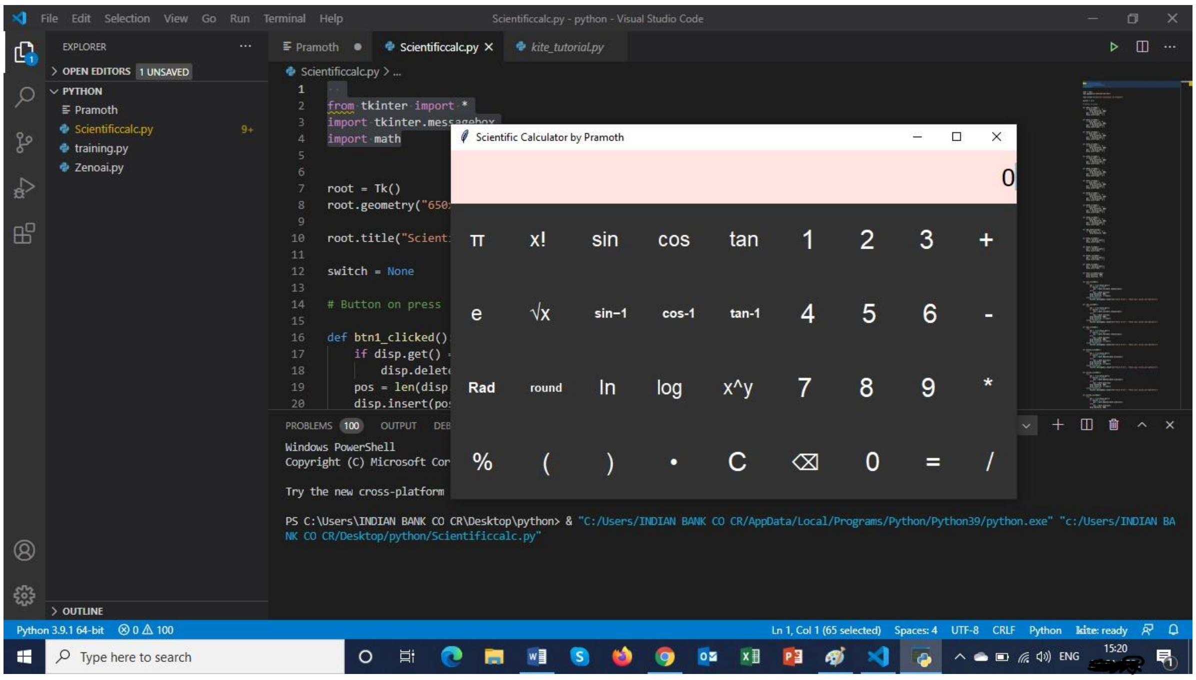
```
disp.bind("<Return>", btneq_clicked)
disp.bind("<Escape>", btnc_clicked)
disp.bind("<Key-1>", key_event)
disp.bind("<Key-2>", key_event)
disp.bind("<Key-3>", key_event)
disp.bind("<Key-4>", key_event)
disp.bind("<Key-5>", key_event)
disp.bind("<Key-6>", key_event)
disp.bind("<Key-7>", key_event)
disp.bind("<Key-8>", key_event)
disp.bind("<Key-9>", key_event)
disp.bind("<Key-0>", key_event)
disp.bind("<Key-.>", key_event)
disp.insert(0, '0')
disp.focus_set()
disp.pack(expand=TRUE, fill=BOTH)
# Row 1 Buttons
btnrow1 = Frame(root, bg="#000000")
btnrow1.pack(expand=TRUE, fill=BOTH)
pi_btn = Button(btnrow1, text="π", font="Segoe 18", relief=GROOVE, bd=0,
comma
nd=pi_clicked, fg="white", bg="#333333")
pi_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
fact_btn = Button(btnrow1, text=" x! ", font="Segoe 18", relief=GROOVE,
bd=0,
command=fact_clicked, fg="white", bg="#333333")
fact_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
sin_btn = Button(btnrow1, text="sin", font="Segoe 18", relief=GROOVE,
bd=0, co
mmand=sin_clicked, fg="white", bg="#333333")
sin_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
cos_btn = Button(btnrow1, text="cos", font="Segoe 18", relief=GROOVE,
bd=0, co
mmand=cos_clicked, fg="white", bg="#333333")
cos_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
tan_btn = Button(btnrow1, text="tan", font="Segoe 18", relief=GROOVE,
bd=0, co
mmand=tan_clicked, fg="white", bg="#333333")
tan_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn1 = Button(btnrow1, text="1", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn1_clicked, fg="white", bg="#333333")
btn1.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn2 = Button(btnrow1, text="2", font="Segoe 23", relief=GROOVE, bd=0,
comman
d=btn2_clicked, fg="white", bg="#333333")
19
btn2.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn3 = Button(btnrow1, text="3", font="Segoe 23", relief=GROOVE, bd=0,
command
```



```
=btn3_clicked, fg="white", bg="#333333")
btn3.pack(side=LEFT, expand=TRUE, fill=BOTH)
btnp = Button(btnrow1, text="+", font="Segoe 23", relief=GROOVE, bd=0,
command
=btnp_clicked, fg="white", bg="#333333")
btnp.pack(side=LEFT, expand=TRUE, fill=BOTH)
# Row 2 Buttons
btnrow2 = Frame(root)
btnrow2.pack(expand=TRUE, fill=BOTH)
e_btn = Button(btnrow2, text="e", font="Segoe 18", relief=GROOVE, bd=0,
comman
d=e_clicked, fg="white", bg="#333333")
e_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
sqr_btn = Button(btnrow2, text=" √x ", font="Segoe 18", relief=GROOVE,
bd=0, c
ommand=sqr_clicked, fg="white", bg="#333333")
sqr_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
sinh_btn = Button(btnrow2, text="sin-1", font="Segoe 11 bold",
relief=GROOVE,
bd=0, command=arcsin_clicked, fg="white", bg="#333333")
sinh_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
cosh_btn = Button(btnrow2, text="cos1", font="Segoe 11 bold",
relief=GROOVE, bd=0, command=arccos_clicked, fg="whi
te", bg="#333333")
cosh_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
tanh_btn = Button(btnrow2, text="tan1", font="Segoe 11 bold",
relief=GROOVE, bd=0, command=arctan_clicked, fg="whi
te", bg="#333333")
tanh_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn4 = Button(btnrow2, text="4", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn4_clicked, fg="white", bg="#333333")
btn4.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn5 = Button(btnrow2, text="5", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn5_clicked, fg="white", bg="#333333")
btn5.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn6 = Button(btnrow2, text="6", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn6_clicked, fg="white", bg="#333333")
btn6.pack(side=LEFT, expand=TRUE, fill=BOTH)
20
btnm = Button(btnrow2, text="-
", font="Segoe 23", relief=GROOVE, bd=0, command=btnm_clicked,
fg="white", bg=
"#333333")
btnm.pack(side=LEFT, expand=TRUE, fill=BOTH)
# Row 3 Buttons
btnrow3 = Frame(root)
btnrow3.pack(expand=TRUE, fill=BOTH)
```

```
conv_btn = Button(btnrow3, text="Rad", font="Segoe 12 bold",
relief=GROOVE, bd
=0, command=conv_clicked, fg="white", bg="#333333")
conv_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
round_btn = Button(btnrow3, text="round", font="Segoe 10 bold",
relief=GROOVE,
bd=0, command=round_clicked, fg="white", bg="#333333")
round_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
ln_btn = Button(btnrow3, text="ln", font="Segoe 18", relief=GROOVE, bd=0,
comm
and=ln_clicked, fg="white", bg="#333333")
ln_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
logarithm_btn = Button(btnrow3, text="log", font="Segoe 17",
relief=GROOVE, bd
=0, command=logarithm_clicked, fg="white", bg="#333333")
logarithm_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
pow_btn = Button(btnrow3, text="x^y", font="Segoe 17", relief=GROOVE,
bd=0, co
mmand=pow_clicked, fg="white", bg="#333333")
pow_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn7 = Button(btnrow3, text="7", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn7_clicked, fg="white", bg="#333333")
btn7.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn8 = Button(btnrow3, text="8", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn8_clicked, fg="white", bg="#333333")
btn8.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn9 = Button(btnrow3, text="9", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn9_clicked, fg="white", bg="#333333")
btn9.pack(side=LEFT, expand=TRUE, fill=BOTH)
btnml = Button(btnrow3, text="*", font="Segoe 23", relief=GROOVE, bd=0,
comman
d=btnml_clicked, fg="white", bg="#333333")
btnml.pack(side=LEFT, expand=TRUE, fill=BOTH)
# Row 4 Buttons
21
btnrow4 = Frame(root)
btnrow4.pack(expand=TRUE, fill=BOTH)
mod_btn = Button(btnrow4, text="%", font="Segoe 21", relief=GROOVE, bd=0,
comm
and=mod_clicked, fg="white", bg="#333333")
mod_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
bl_btn = Button(btnrow4, text=" ( ", font="Segoe 21", relief=GROOVE,
bd=0, com
mand=bl_clicked, fg="white", bg="#333333")
bl_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
br_btn = Button(btnrow4, text=" ) ", font="Segoe 21", relief=GROOVE,
bd=0, com
mand=br_clicked, fg="white", bg="#333333")
br_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
dot_btn = Button(btnrow4, text=" • ", font="Segoe 21", relief=GROOVE,
bd=0, co
mmand=dot_clicked, fg="white", bg="#333333")
dot_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
btnc = Button(btnrow4, text="C", font="Segoe 23", relief=GROOVE, bd=0,
command
=btnc_clicked, fg="white", bg="#333333")
btnc.pack(side=LEFT, expand=TRUE, fill=BOTH)
del_btn = Button(btnrow4, text="□", font="Segoe 20", relief=GROOVE, bd=0,
comm
and=del_clicked, fg="white", bg="#333333")
del_btn.pack(side=LEFT, expand=TRUE, fill=BOTH)
btn0 = Button(btnrow4, text="0", font="Segoe 23", relief=GROOVE, bd=0,
command
=btn0_clicked, fg="white", bg="#333333")
btn0.pack(side=LEFT, expand=TRUE, fill=BOTH)
btneq = Button(btnrow4, text="=", font="Segoe 23", relief=GROOVE, bd=0,
comman
d=btneq_clicked, fg="white", bg="#333333")
btneq.pack(side=LEFT, expand=TRUE, fill=BOTH)
btnd = Button(btnrow4, text="/", font="Segoe 23", relief=GROOVE, bd=0,
command
=btnd_clicked, fg="white", bg="#333333")
btnd.pack(side=LEFT, expand=TRUE, fill=BOTH)
root.mainloop()
```

5.Output

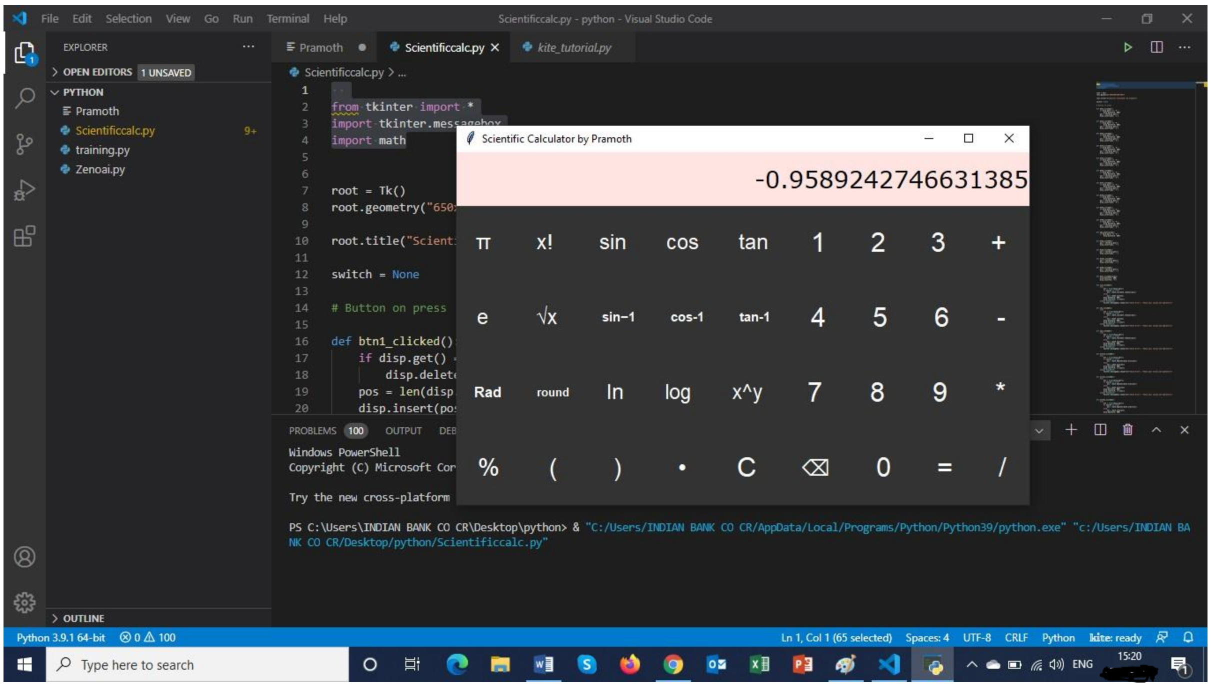

File Edit Selection View Go Run Terminal Help
Scientificcalc.py - python - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
PYTHON
Pramoth
Scientificcalc.py
training.py
Zenoai.py
OUTLINE
Scientific Calculator by Pramoth
0
π x! sin cos tan 1 2 3 +
e √x sin-1 cos-1 tan-1 4 5 6 -
Rad round ln log x^y 7 8 9 *
% ( ) • C ⌫ 0 = /
PROBLEMS 100 OUTPUT
Python 3.9.1 64-bit
Ln 1, Col 1 (65 selected) Spaces: 4 UTF-8 CRLF Python kite: ready
Type here to search
ENG 15:20

```
from tkinter import *
import tkinter.messagebox
import math

root = Tk()
root.geometry("650
root.title("Scient

switch = None

# Button on press

def btn1_clicked()
    if disp.get() =
        disp.delete
    pos = len(disp
    disp.insert(pos
```

```
Windows PowerShell
Copyright (C) Microsoft Cor

Try the new cross-platform

PS C:\Users\INDIAN BANK CO CR\Desktop\python> & "C:/Users/INDIAN BANK CO CR/AppData/Local/Programs/Python/Python39/python.exe" "c:/Users/INDIAN BA
NK CO CR/Desktop/python/Scientificcalc.py"
```

[redacted]

Fig2: Running the Python file in the terminal


File Edit Selection View Go Run Terminal Help
Scientificcalc.py - python - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
PYTHON
Pramoth
Scientificcalc.py
training.py
Zenoai.py
OUTLINE
Scientific Calculator by Pramoth
-0.9589242746631385
π x! sin cos tan 1 2 3 +
e √x sin-1 cos-1 tan-1 4 5 6 -
Rad round ln log x^y 7 8 9 *
% ( ) • C ⌫ 0 = /
PROBLEMS 100 OUTPUT
Python 3.9.1 64-bit
Ln 1, Col 1 (65 selected) Spaces: 4 UTF-8 CRLF Python kite: ready
Type here to search
ENG 15:20

```
from tkinter import *
import tkinter.messagebox
import math

root = Tk()
root.geometry("650
root.title("Scient

switch = None

# Button on press

def btn1_clicked()
    if disp.get() =
        disp.delete
    pos = len(disp
    disp.insert(pos
```

```
Windows PowerShell
Copyright (C) Microsoft Cor

Try the new cross-platform

PS C:\Users\INDIAN BANK CO CR\Desktop\python> & "C:/Users/INDIAN BANK CO CR/AppData/Local/Programs/Python/Python39/python.exe" "c:/Users/INDIAN BA
NK CO CR/Desktop/python/Scientificcalc.py"
```

[redacted]

Fig3: Successful Output (Answer for sin5)

6.Conclusion

The proposed system is error free. Trivial concepts of Python language are implemented into the system. As, the usage of Python Tkinter as the GUI provided various controls, such as buttons, labels, and text boxes to build a user friendly application.

The rapid expansion and use of the internet, confirms the splendid future and scope of the project.

7.References

[1] Al Sweigart - Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners – 2015.

[2] Python GUI Programming with Tkinter: Develop Responsive and Powerful GUI Applications with Tkinter – 2018.