A

SEMINAR REPORT

ON

Scientific Calculator by python

SUBMITTED TO

JAGANNATH UNIVERSITY JAIPUR

OF

BACHLOR OF TECHNOLOGY

OF **COMPUTER SCIENCE**

SESSION 2023-2027

SUBMITTED TO:

Dr. Renu bagoria

(Dept. Of CS)

SUBMITTED BY:

Hritik Singh

Enroll No : 0201230163

B.Tech. 3nd Yr. CS

JAGANNATH UNIVERSITY JAIPUR

(Chaksu Campus)

ACKNOWLEDGEMENT

This is opportunity to express my heartfelt words for the people who were part of this seminar in numerous ways. People who gave me unending support right from beginning of the seminar.

I want to give sincere thanks to the principal Dr. Vaishali Sharma for her valuable support.

I extend my thanks to Dr. Renu Bagoria Head of the Department for her constant support.

Your Sincerely,

Hritik Singh

Enroll No: 201230163

Candidate's Declaration

I Hritik Singh hereby declare that the work presented in this report in partial fulfilled of the requirement for the award of Degree of Bachelor of Technology, submitted in the Department of COMPUTER SCIENCE at JU.

Hritik Singh **Dr. Renu Bagoria**

(Name & Signature of Candidate) (Counter signed by)

Certificate

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.

Table Of Contents

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'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.

● **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.

A part 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.

2.Tkinter Programming

Tkinter (short for *Tk Interface*) is the **standard GUI (Graphical User Interface) library** for Python.
It allows you to build windows, buttons, labels, text boxes, menus, and full desktop applications.

Tkinter comes **pre-installed** with Python, so no separate installation is required.

Why Use Tkinter?

- Simple and beginner-friendly
- Comes built into Python
- Large collection of widgets
- Works on Windows, macOS, and Linux
- Good for small and medium desktop applications

Basic Concepts in Tkinter

To build applications using Tkinter, you need to understand the following concepts:

a) Tk() – The main window

example.py

```
from tkinter import *
win = Tk()
win.mainloop()
```

.

b) Widgets

Widgets are components used to build the GUI interface. Examples include:

Widget	Description
Label	Displays text or images
Button	A clickable button
Entry	Single-line text input
Text	Multi-line text box
Canvas	For drawing shapes or images
Frame	Container to organize other widgets
Checkbutton	Checkbox
Radiobutton	Option selector
Listbox	List of selectable items

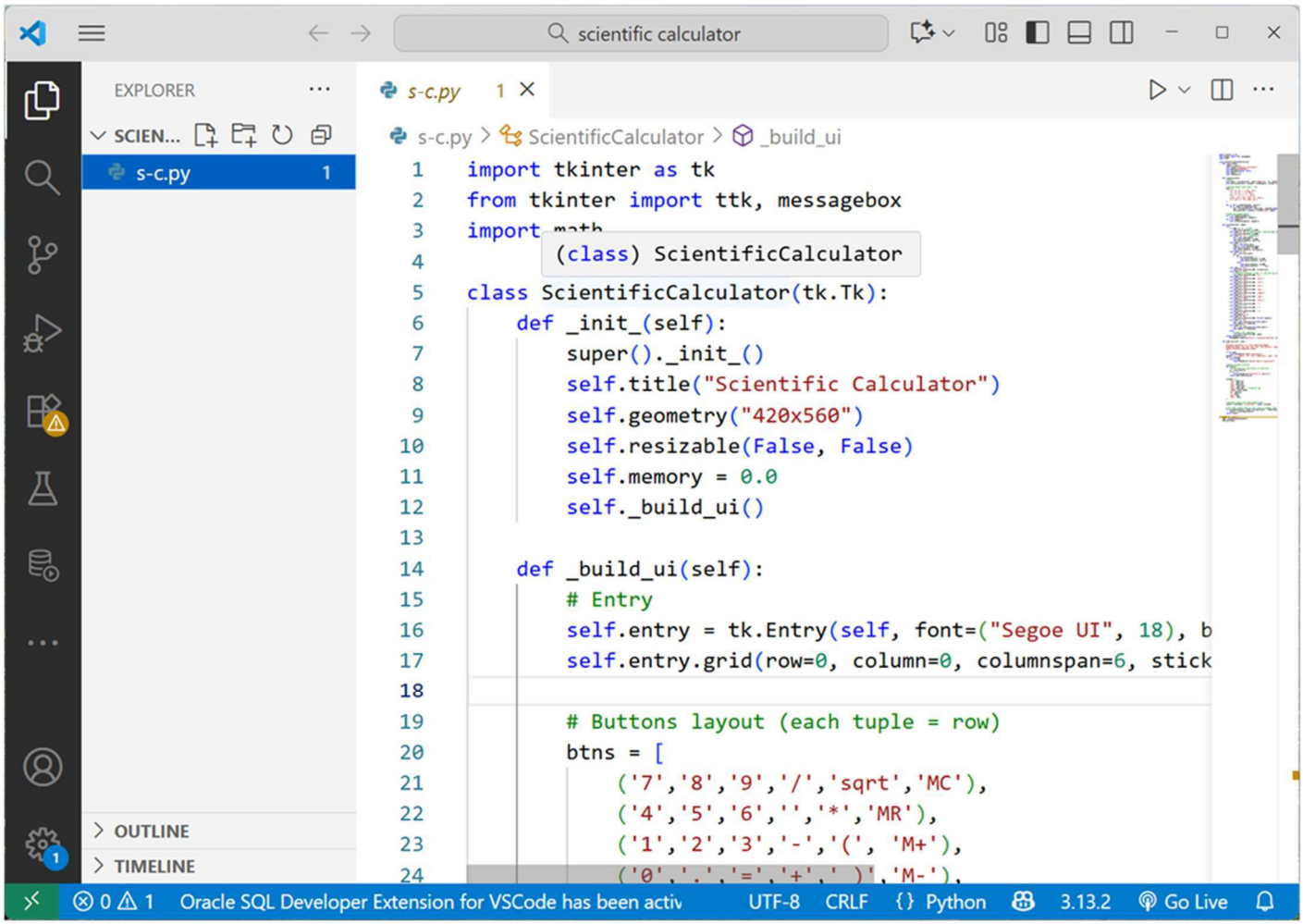
c) Geometry Managers

Geometry managers position widgets in the window.

- pack() places widgets automatically in blocks
- grid() uses rows and columns
- place() positions widgets using x, y coordinates

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.



4.Source Code

The code for the, Scientific Calculator is as follows:

```
import tkinter as tk
from tkinter import ttk, messagebox
import math


class ScientificCalculator(tk.Tk):
    def _init_(self):
        super()._init_()
        self.title("Scientific Calculator")
        self.geometry("420x560")
        self.resizable(False, False)
        self.memory = 0.0
        self._build_ui()


    def _build_ui(self):
        # Entry
        self.entry = tk.Entry(self, font=("Segoe UI", 18), borderwidth=2, relief="groove", justify="right")
        self.entry.grid(row=0, column=0, columnspan=6, sticky="nsew", padx=8, pady=10, ipady=8)


        # Buttons layout (each tuple = row)
        btns = [
            ('7','8','9','/','sqrt','MC'),
            ('4','5','6','','*','MR'),
            ('1','2','3','-','(', 'M+'),
            ('0','.','=','+',' )','M-'),
            ('sin','cos','tan','log','ln','fact'),
```

```
        ('C','CE','<-','±','exp','pi')
    ]

    for r, row in enumerate(btns, start=1):
        for c, label in enumerate(row):
            btn = ttk.Button(self, text=label, command=lambda x=label: self.on_button(x))
            btn.grid(row=r, column=c, sticky="nsew", padx=4, pady=4)

    # make grid expand evenly
    for i in range(len(btns)+1):
        self.rowconfigure(i, weight=1)
    for j in range(6):
        self.columnconfigure(j, weight=1)

def on_button(self, label):
    try:
        if label == 'C':  # clear
            self.entry.delete(0, tk.END)
        elif label == 'CE': # same as clear (could be made to clear last entry)
            self.entry.delete(0, tk.END)
        elif label == '<-':  # backspace
            cur = self.entry.get()
            self.entry.delete(0, tk.END)
            self.entry.insert(0, cur[:-1])
        elif label == '=':
            expr = self.entry.get()
            result = self.safe_eval(expr)
            self.entry.delete(0, tk.END)
            self.entry.insert(0, str(result))
```

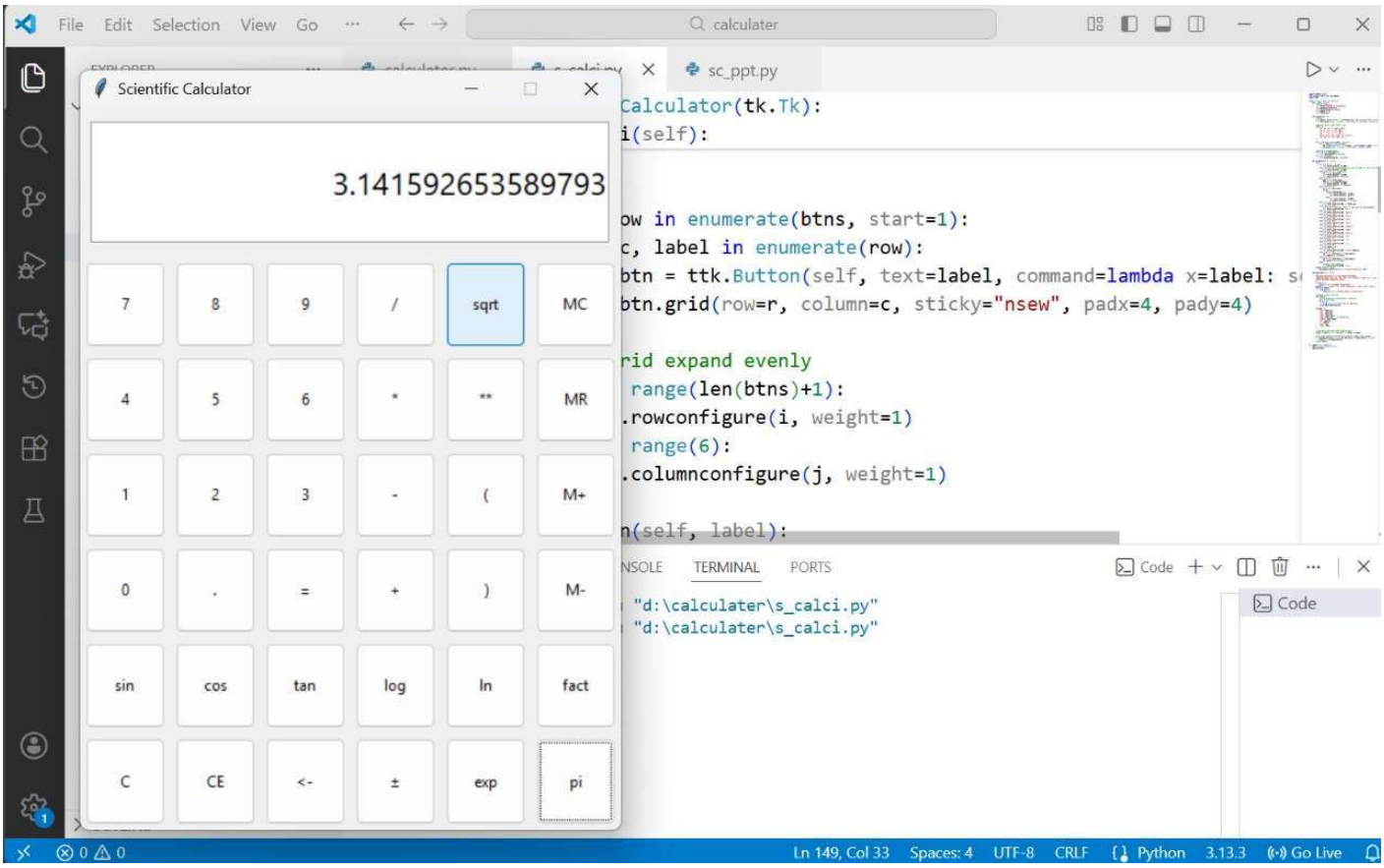
```
elif label == '±':
    cur = self.entry.get()
    if cur:
        if cur.startswith('-'):
            self.entry.delete(0, tk.END)
            self.entry.insert(0, cur[1:])
        else:
            self.entry.delete(0, tk.END)
            self.entry.insert(0, '-' + cur)
elif label == 'pi':
    self.entry.insert(tk.END, str(math.pi))
elif label == 'exp':
    # scientific notation: insert 'e' and user can type exponent
    self.entry.insert(tk.END, 'e')
elif label == 'sqrt':
    self.entry.insert(tk.END, 'sqrt(')
elif label == 'sin':
    self.entry.insert(tk.END, 'sin(')
elif label == 'cos':
    self.entry.insert(tk.END, 'cos(')
elif label == 'tan':
    self.entry.insert(tk.END, 'tan(')
elif label == 'log':
    self.entry.insert(tk.END, 'log10(')
elif label == 'ln':
    self.entry.insert(tk.END, 'log(')
elif label == 'fact':
    self.entry.insert(tk.END, 'fact(')
elif label == '':
```

```
            self.entry.insert(tk.END, '')
        elif label == '(':
            self.entry.insert(tk.END, '(')
        elif label == ' )':
            self.entry.insert(tk.END, ')')
        elif label == 'MC':
            self.memory = 0.0
        elif label == 'MR':
            self.entry.insert(tk.END, str(self.memory))
        elif label == 'M+':
            val = self.safe_eval(self.entry.get())
            self.memory += float(val)
        elif label == 'M-':
            val = self.safe_eval(self.entry.get())
            self.memory -= float(val)
        else:
            # digits, dot, operators
            self.entry.insert(tk.END, label)
    except Exception as e:
        messagebox.showerror("Error", f"Invalid operation: {e}")


def safe_eval(self, expr):
    """
    Evaluate expression in a restricted environment.
    Supported names: sin, cos, tan, sqrt, log (natural), log10, pi, e, pow, fact
    Blocks obviously dangerous tokens.
    """
    if not expr:
        raise ValueError("Empty expression")
```

```
banned = ['','import','os','sys','subprocess','open','eval','exec']
for b in banned:
    if b in expr:
        raise ValueError("Unsafe token in expression")

# define allowed functions
def fact(x):
    # allow factorial like fact(5) or fact(5.0)
    xi = int(float(x))
    if xi < 0:
        raise ValueError("Factorial of negative")
    return math.factorial(xi)

allowed = {
    'sin': math.sin,
    'cos': math.cos,
    'tan': math.tan,
    'sqrt': math.sqrt,
    'log': math.log,     # natural log
    'log10': math.log10,
    'pi': math.pi,
    'e': math.e,
    'pow': pow,
    'fact': fact,
}

# Evaluate using restricted globals/locals
result = eval(expr, {"__builtins__": None}, allowed)
```

```
        # For float results that are very close to integer, show integer
        if isinstance(result, float) and abs(result - round(result)) < 1e-12:
            result = int(round(result))
        return result

if _name_ == "_main_":
    app = ScientificCalculator()
    app.mainloop()
```

5 Output


Scientific Calculator
3.141592653589793
7
8
9
/
sqrt
MC
4
5
6
*
**
MR
1
2
3
-
(
M+
0
.
=
+
)
M-
sin
cos
tan
log
ln
fact
C
CE
<-
±
exp
pi
sc_ppt.py
TERMINAL
PORTS
"d:\calculater\s_calci.py"
Ln 149, Col 33
Spaces: 4
UTF-8
CRLF
Python
3.13.3
Go Live

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