Calculator_Project.ipynb

September 26, 2025

1 Mini Calculator Project

Course: Python Final Project
Group: Group 3
Team Members: - Reihaneh Niknahad - Thi Ngoc Hanh Nguyen

1.1 Project Description

Our project is a **Mini Calculator with a simple UI built using Gradio**.
It supports basic arithmetic operations (`+`, `-`, `*`, `/`) and also advanced ones (x^2, $\sqrt{x}$).
The program is divided into multiple files for better structure and reusability: - **simple_calculator.py** – first console-based version - **math_core.py** – core calculation functions - **calculator.py** – UI with Gradio

1.2 Peer Evaluation (for teacher & teammates)

- Comments from team members will be added here.

[1]:
```
"""
=====================================================
 Mini Calculator Project - math_core.py
-----------------------------------------------------
 Purpose:
   - Contains all core math functions (add, subtract, multiply, divide, sqrt,␣
 ↪power).
   - Designed to be reusable for different UIs.
==================================
"""
import math

def calculate(a, b, op):
    if op == '+':
        return a + b
    if op == '-':
        return a - b
    if op == '*':
        return a * b
    if op == '/':
```

```
        if b == 0:
            raise ZeroDivisionError("Cannot divide by 0")
        return a / b
    if op == '^2':
        return a ** 2
    if op == 'sqrt':
        if a < 0:
            raise ValueError("Cannot take sqrt of negative number")
        return math.sqrt(a)
    raise ValueError("Does not support this operator")
```

[2]:
```
"""
=====================================================
 Mini Calculator Project - calculator.py
-----------------------------------------------------
 Purpose:
   - Provides the User Interface using Gradio.
   - Connects UI buttons to functions in math_core.py.
=====================================================
"""

import gradio as gr
# from math_core import calculate

# Map pretty labels -> internal tokens
PRETTY_TO_TOKEN = {
    "√ ": "√",
    " ²": "^2",
}

def on_click(btn, state, stopped):
    # Map pretty button labels to internal tokens
    btn = PRETTY_TO_TOKEN.get(btn, btn)

    if stopped:
        return state, gr.update(value="Stop the application",␣
 ↪elem_classes="display-err"), True

    # Reset
    if btn == "C":
        return "", gr.update(value="", elem_classes="display-num"), False

    # Exit
    elif btn == "Exit":
        return "", gr.update(value="The application was stopped",␣
 ↪elem_classes="display-err"), True
```

```
    # Perform calculation
    elif btn == "=":
        try:
            op = None
            for o in ["+", "-", "*", "/", "^2", "√"]:
                if o in state:
                    op = o
                    break

            if op is None:
                return state, gr.update(value="Err: Invalid expression",␣
↪elem_classes="display-err"), False

            if op == "√":
                try:
                    a = float(state.replace("√", ""))
                    if a < 0:
                        return state, gr.update(value="Err: Negative number␣
↪under square root", elem_classes="display-err"), False
                    result = calculate(a, 0, "sqrt")
                except ValueError:
                    return state, gr.update(value="Error: Invalid input",␣
↪elem_classes="display-err"), False

            elif op == "^2":
                try:
                    a = float(state.replace("^2", ""))
                    result = calculate(a, 0, "^2")
                except ValueError:
                    return state, gr.update(value="Error: Invalid input",␣
↪elem_classes="display-err"), False

            else:
                parts = state.split(op)
                if len(parts) != 2:
                    return state, gr.update(value="Err: Invalid expression",␣
↪elem_classes="display-err"), False
                a = float(parts[0].strip())
                b = float(parts[1].strip())
                result = calculate(a, b, op)

            return state, gr.update(value=str(result),␣
↪elem_classes="display-num"), False

        except Exception as e:
```

```
            return state, gr.update(value=f"Err: {e}",␣
↪elem_classes="display-err"), False

    else:
        # Prevent invalid cases
        if state.endswith("√") and btn == "-":
            return state, gr.update(value="Err: Cannot take sqrt of negative␣
↪number", elem_classes="display-err"), False
        if state == "" and btn == "-":
            return state, gr.update(value="Err: Cannot start with minus",␣
↪elem_classes="display-err"), False
        if "√-" in state + btn or "√(-" in state + btn:
            return state, gr.update(value="Err: Cannot take sqrt of negative␣
↪number", elem_classes="display-err"), False

        new_state = state + btn
        return new_state, gr.update(value=new_state,␣
↪elem_classes="display-num"), False


def launch_ui():
    with gr.Blocks(css="""
        .btn { width:60px !important; height:60px !important; font-size:18px;␣
↪flex:none !important; }
        .display-num textarea { width:400px !important; height:40px !important;␣
↪font-size:20px; text-align:right; }
        .display-err textarea { width:400px !important; height:40px !important;␣
↪font-size:18px; text-align:left; color:#ff5555; }
    """) as demo:
        gr.Markdown("###  Mini Calculator")

        state = gr.State("")
        stopped = gr.State(False)

        with gr.Row():
            with gr.Column(scale=0):
                display = gr.Textbox(
                    label="Result",
                    value="",
                    interactive=False,
                    lines=2,
                    max_lines=2,
                    elem_classes="display-num"   # Display numbers with right␣
↪align
                )
```

```
        buttons = [
            ["7", "8", "9", "/"],
            ["4", "5", "6", "*"],
            ["1", "2", "3", "-"],
            ["0", ".", "=", "+"],
            ["√", " ²", "C", "Exit"]
        ]

        for row in buttons:
            with gr.Row():
                for label in row:
                    btn = gr.Button(label, elem_classes="btn")
                    internal = PRETTY_TO_TOKEN.get(label, label)  # map sqrt &␣
↪^2 labels -> token
                    btn.click(
                        on_click,
                        inputs=[gr.Textbox(value=internal, visible=False),␣
↪state, stopped],
                        outputs=[state, display, stopped],
                        show_progress=False
                    )

    demo.launch()

if __name__ == "__main__":
    launch_ui()
```

```
* Running on local URL:  http://127.0.0.1:7861
* To create a public link, set `share=True` in `launch()`.
```

```
<IPython.core.display.HTML object>
```

[ ]: