Republic of the Philippines
**CAMARINES SUR POLYTECHNIC COLLEGES**
Nabua, Camarines Sur

**COLLEGE *of* COMPUTER STUDIES**

# Final Project and Final Exam

Defensive Programming
CSEC-311

## Modification of Python-based projects with Defensive Programming

Submitted by:

Justine Labios
Mae Julienn Mata
Marc Chrsitian Tumaneng

## I. Project Description

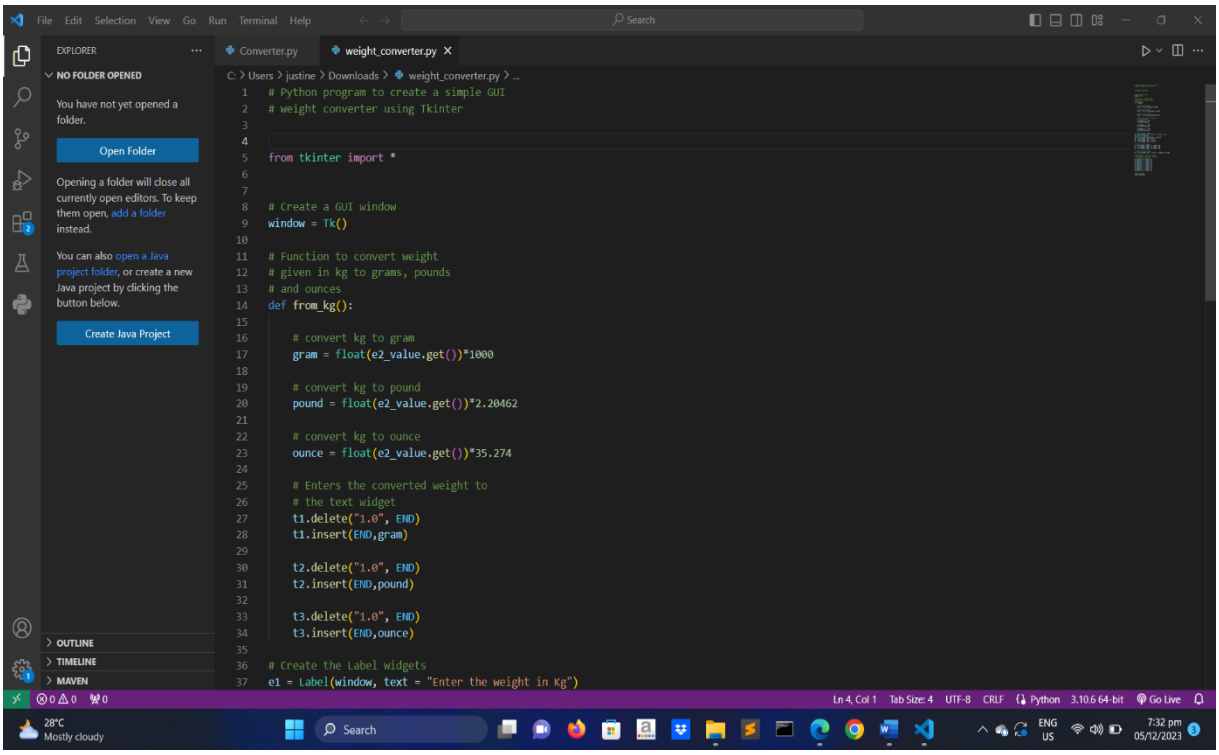
The Weight and Height Converter Application is a Python-based graphical user interface (GUI) program that enables users to easily convert weights and heights between different units. The application is an evolution from a basic weight converter to a more sophisticated tool that incorporates Object-Oriented Programming (OOP) principles, modularization, enhanced functionalities, and extended capabilities to handle both weight and height conversions, providing users with a versatile and user-friendly tool for converting weights and heights. The application's modular and organized codebase, along with its enhanced features, makes it a valuable resource for individuals needing quick and accurate unit conversions.

## II. What did you do? Original vs. Modified (code)

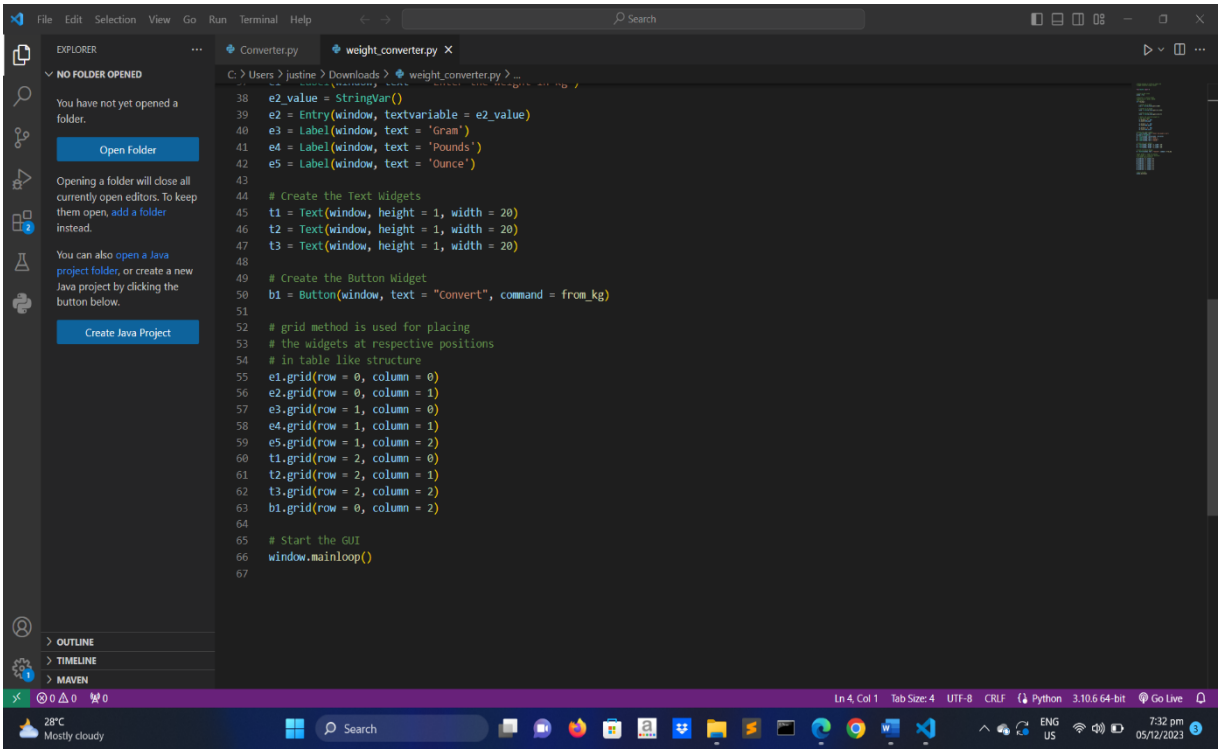
The changes from the original to the modified code involve a shift towards OOP, modularization, enhanced styling, improved input validation, and the addition of new features. The original code is a weight converter using the Tkinter library in Python. It exhibits a procedural structure, with all functionalities implemented in a linear sequence. The only purpose of the code is to convert a given weight in kilograms into grams, pounds, and ounces. The modified code on the other hand represents a substantial evolution from the original code, transforming a basic weight converter into a more sophisticated and user-friendly , height and weight converter. This iteration incorporates Object-Oriented Programming (OOP) principles, modularization, improved styling, and enhanced functionalities. We add new functionalities such as displayed instruction, height conversion, clear button and viewing recent conversion results.

## III. Screenshots of the Original vs Modified

### Original Code

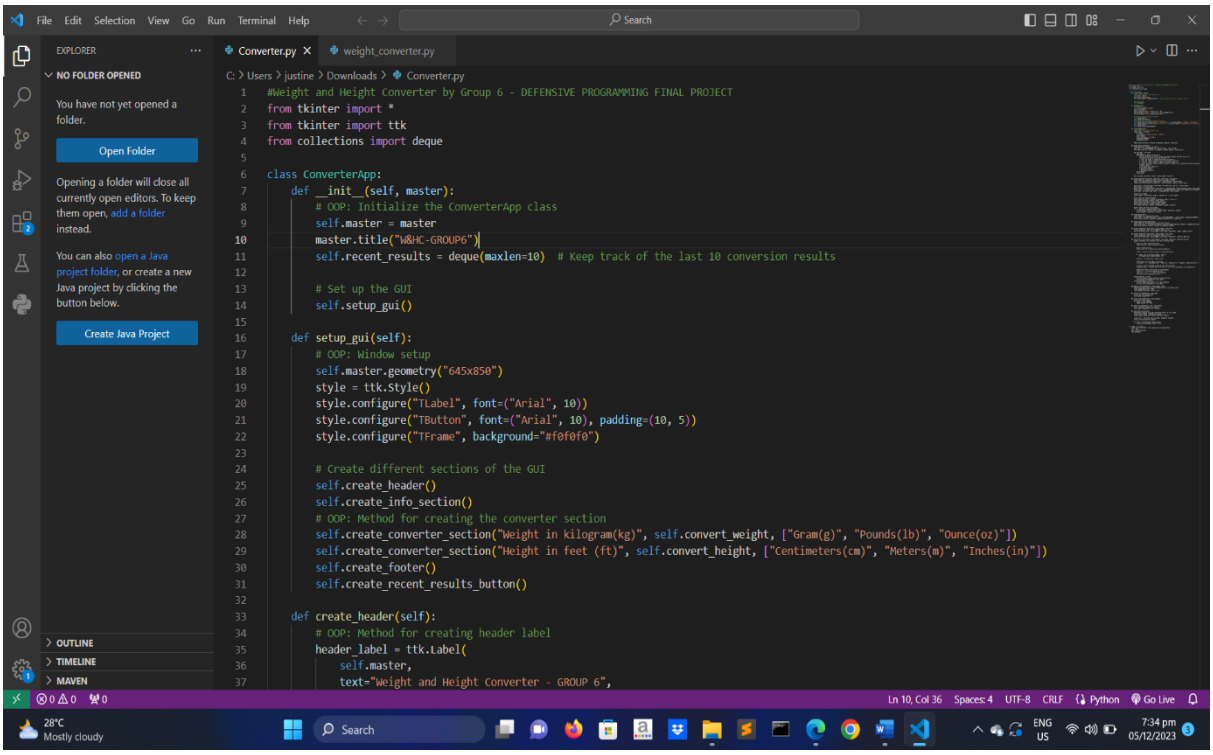


```
# Python program to create a simple GUI
# weight converter using Tkinter


from tkinter import *


# Create a GUI window
window = Tk()

# Function to convert weight
# given in kg to grams, pounds
# and ounces
def from_kg():

    # convert kg to gram
    gram = float(e2_value.get())*1000

    # convert kg to pound
    pound = float(e2_value.get())*2.20462

    # convert kg to ounce
    ounce = float(e2_value.get())*35.274

    # Enters the converted weight to
    # the text widget
    t1.delete("1.0", END)
    t1.insert(END,gram)

    t2.delete("1.0", END)
    t2.insert(END,pound)

    t3.delete("1.0", END)
    t3.insert(END,ounce)

# Create the Label Widgets
e1 = Label(window, text = "Enter the weight in Kg")
```

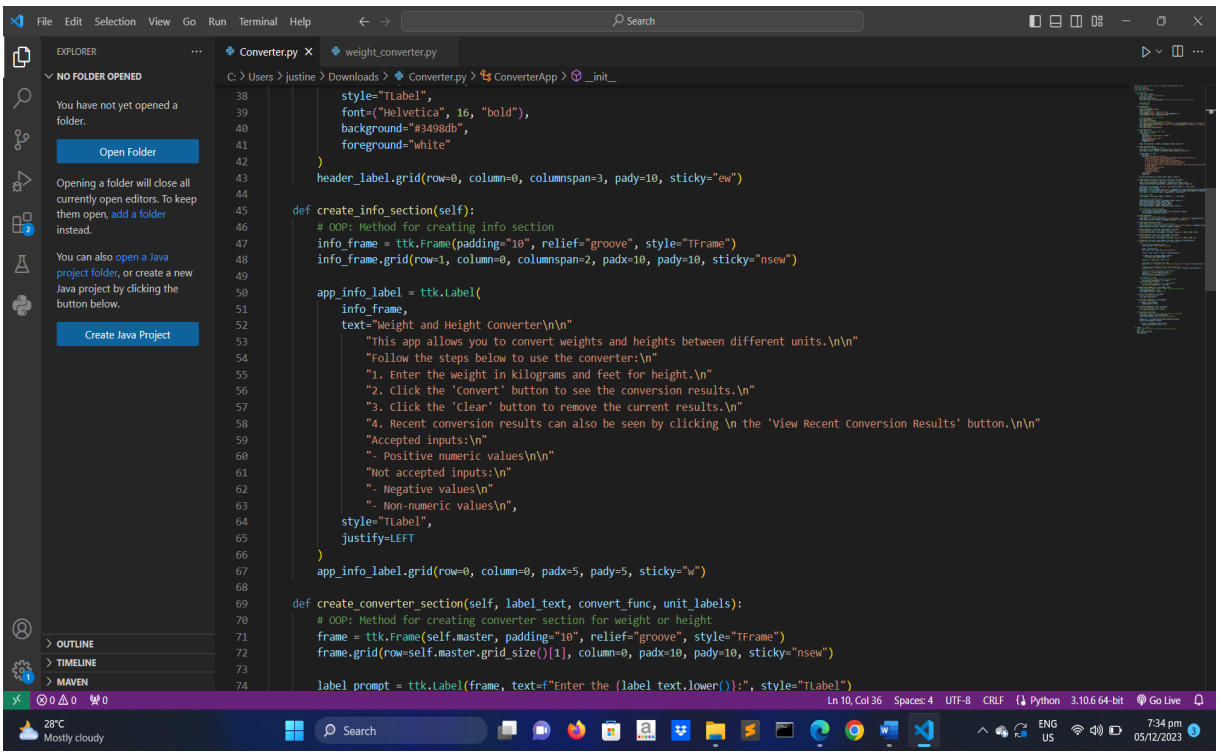


```
e2_value = StringVar()
e2 = Entry(window, textvariable = e2_value)
e3 = Label(window, text = 'Gram')
e4 = Label(window, text = 'Pounds')
e5 = Label(window, text = 'Ounce')

# Create the Text Widgets
t1 = Text(window, height = 1, width = 20)
t2 = Text(window, height = 1, width = 20)
t3 = Text(window, height = 1, width = 20)

# Create the Button Widget
b1 = Button(window, text = "Convert", command = from_kg)

# grid method is used for placing
# the widgets at respective positions
# in table like structure
e1.grid(row = 0, column = 0)
e2.grid(row = 0, column = 1)
e3.grid(row = 1, column = 0)
e4.grid(row = 1, column = 1)
e5.grid(row = 1, column = 2)
t1.grid(row = 2, column = 0)
t2.grid(row = 2, column = 1)
t3.grid(row = 2, column = 2)
b1.grid(row = 0, column = 2)

# Start the GUI
window.mainloop()
```

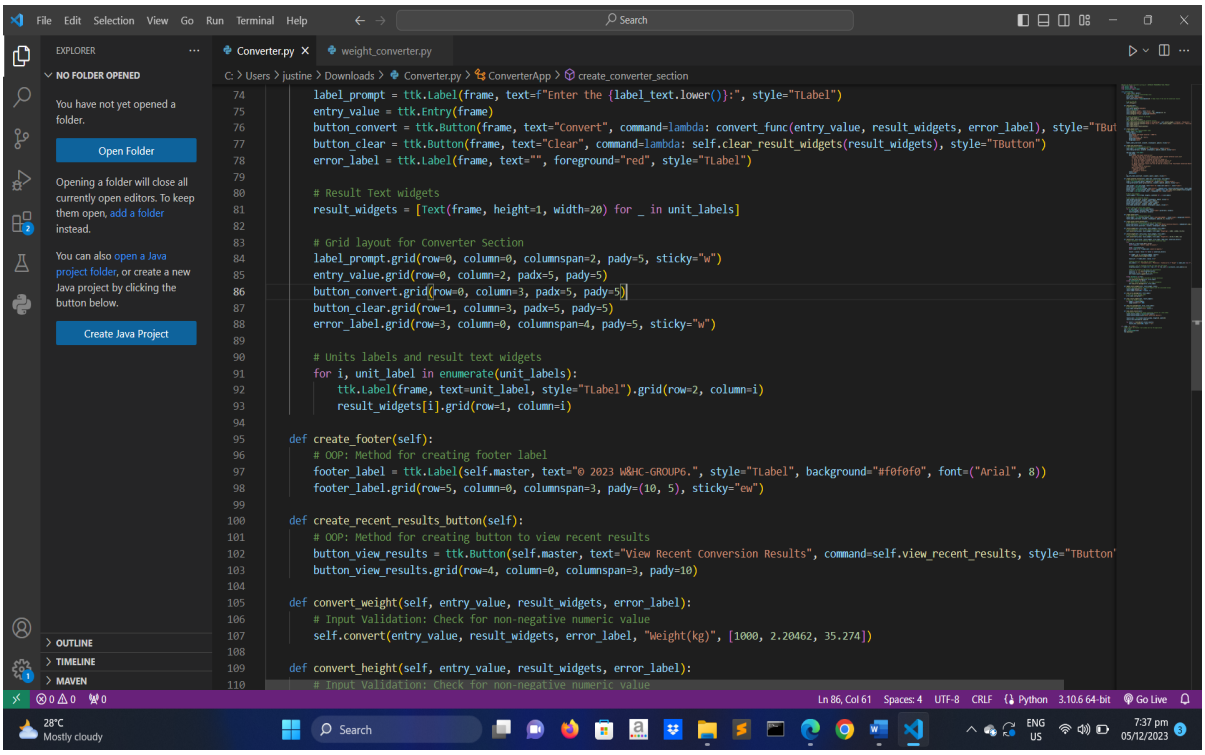
## Modified Code



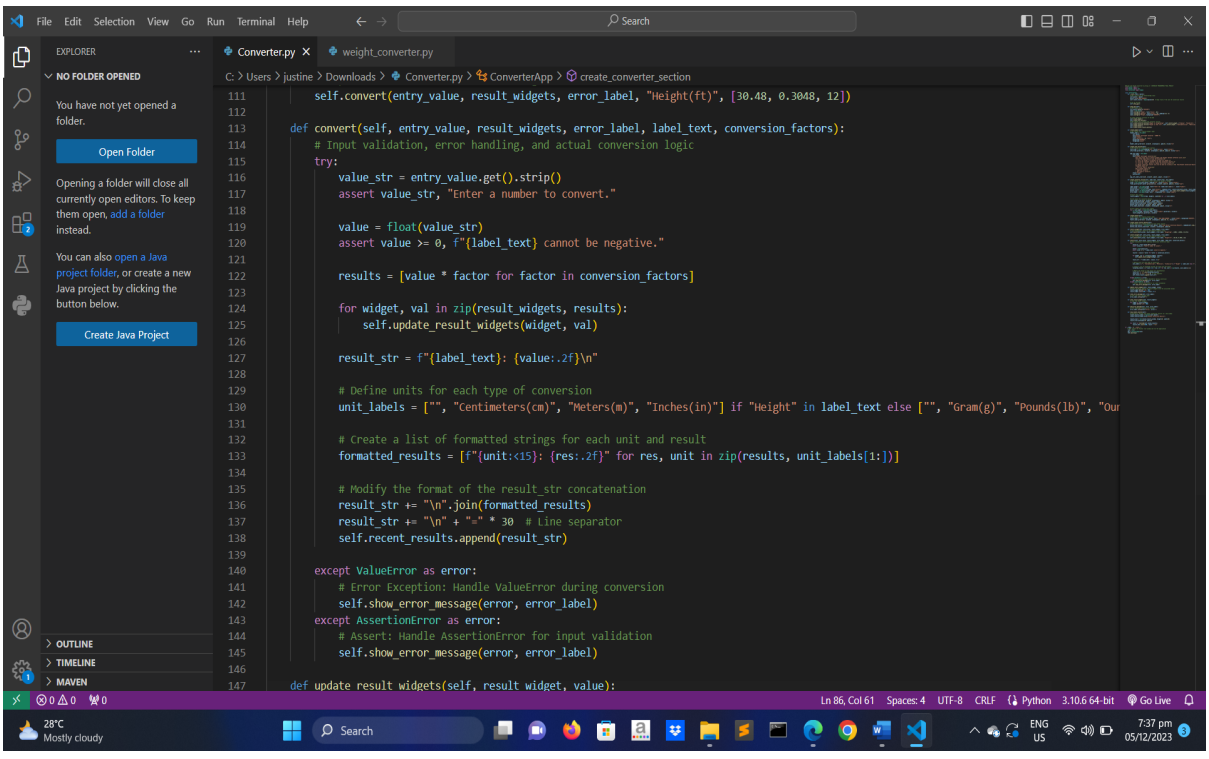
```python
#Weight and Height Converter by Group 6 - DEFENSIVE PROGRAMMING FINAL PROJECT
from tkinter import *
from tkinter import ttk
from collections import deque

class ConverterApp:
    def __init__(self, master):
        # OOP: Initializing the ConverterApp class
        self.master = master
        master.title("W&HC-GROUP6")
        self.recent_results = deque(maxlen=10)  # Keep track of the last 10 conversion results

        # Set up the GUI
        self.setup_gui()

    def setup_gui(self):
        # OOP: Window setup
        self.master.geometry("645x850")
        style = ttk.Style()
        style.configure("TLabel", font=("Arial", 10))
        style.configure("TButton", font=("Arial", 10), padding=(10, 5))
        style.configure("TFrame", background="#f0f0f0")

        # Create different sections of the GUI
        self.create_header()
        self.create_info_section()
        # OOP: Method for creating the converter section
        self.create_converter_section("Weight in kilogram(kg)", self.convert_weight, ["Gram(g)", "Pounds(lb)", "Ounce(oz)"])
        self.create_converter_section("Height in feet (ft)", self.convert_height, ["Centimeters(cm)", "Meters(m)", "Inches(in)"])
        self.create_footer()
        self.create_recent_results_button()

    def create_header(self):
        # OOP: Method for creating header label
        header_label = ttk.Label(
            self.master,
            text="Weight and Height Converter - GROUP 6",
```



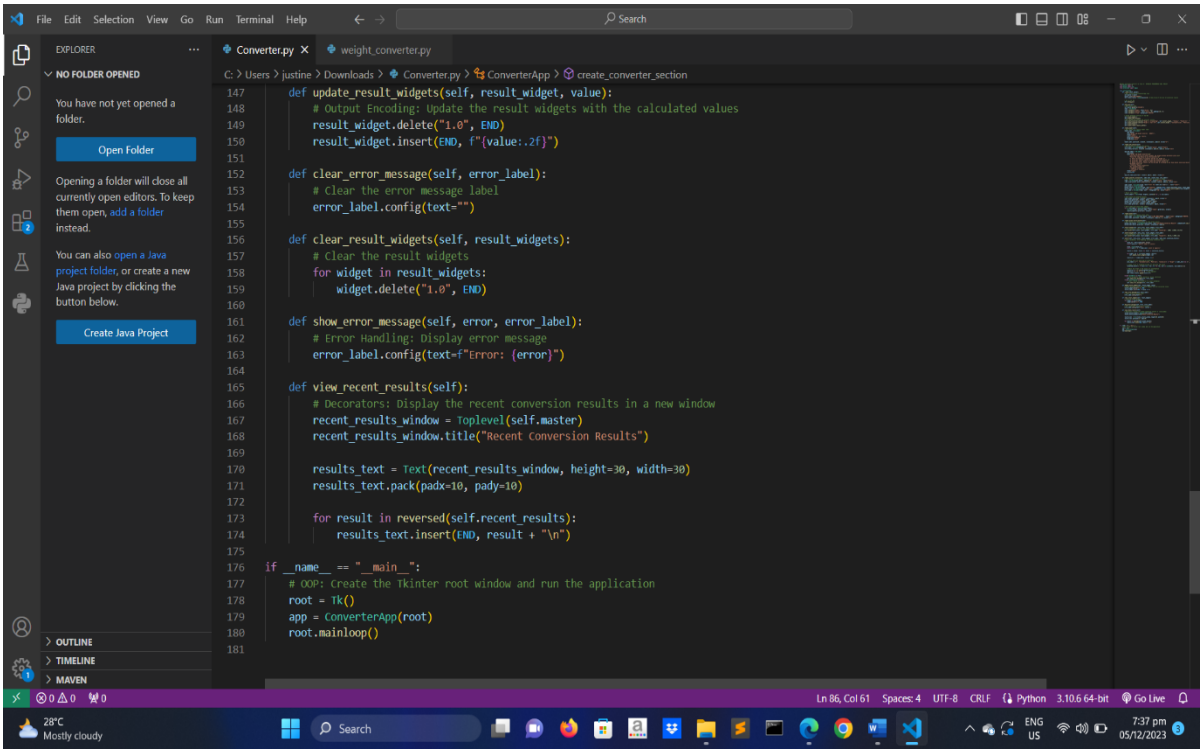
```python
            style="TLabel",
            font=("Helvetica", 16, "bold"),
            background="#3498db",
            foreground="white"
        )
        header_label.grid(row=0, column=0, columnspan=3, pady=10, sticky="ew")

    def create_info_section(self):
        # OOP: Method for creating info section
        info_frame = ttk.Frame(padding="10", relief="groove", style="TFrame")
        info_frame.grid(row=1, column=0, columnspan=2, padx=10, pady=10, sticky="nsew")

        app_info_label = ttk.Label(
            info_frame,
            text="Weight and Height Converter\n\n"
                 "This app allows you to convert weights and heights between different units.\n\n"
                 "Follow the steps below to use the converter:\n"
                 "1. Enter the weight in kilograms and feet for height.\n"
                 "2. Click the 'Convert' button to see the conversion results.\n"
                 "3. Click the 'Clear' button to remove the current results.\n"
                 "4. Recent conversion results can also be seen by clicking \n the 'View Recent Conversion Results' button.\n\n"
                 "Accepted inputs:\n"
                 "- Positive numeric values\n\n"
                 "Not accepted inputs:\n"
                 "- Negative values\n"
                 "- Non-numeric values\n",
            style="TLabel",
            justify=LEFT
        )
        app_info_label.grid(row=0, column=0, padx=5, pady=5, sticky="w")

    def create_converter_section(self, label_text, convert_func, unit_labels):
        # OOP: Method for creating converter section for weight or height
        frame = ttk.Frame(self.master, padding="10", relief="groove", style="TFrame")
        frame.grid(row=self.master.grid_size()[1], column=0, padx=10, pady=10, sticky="nsew")

        label_prompt = ttk.Label(frame, text=f"Enter the {label_text.lower()}:", style="TLabel")
```



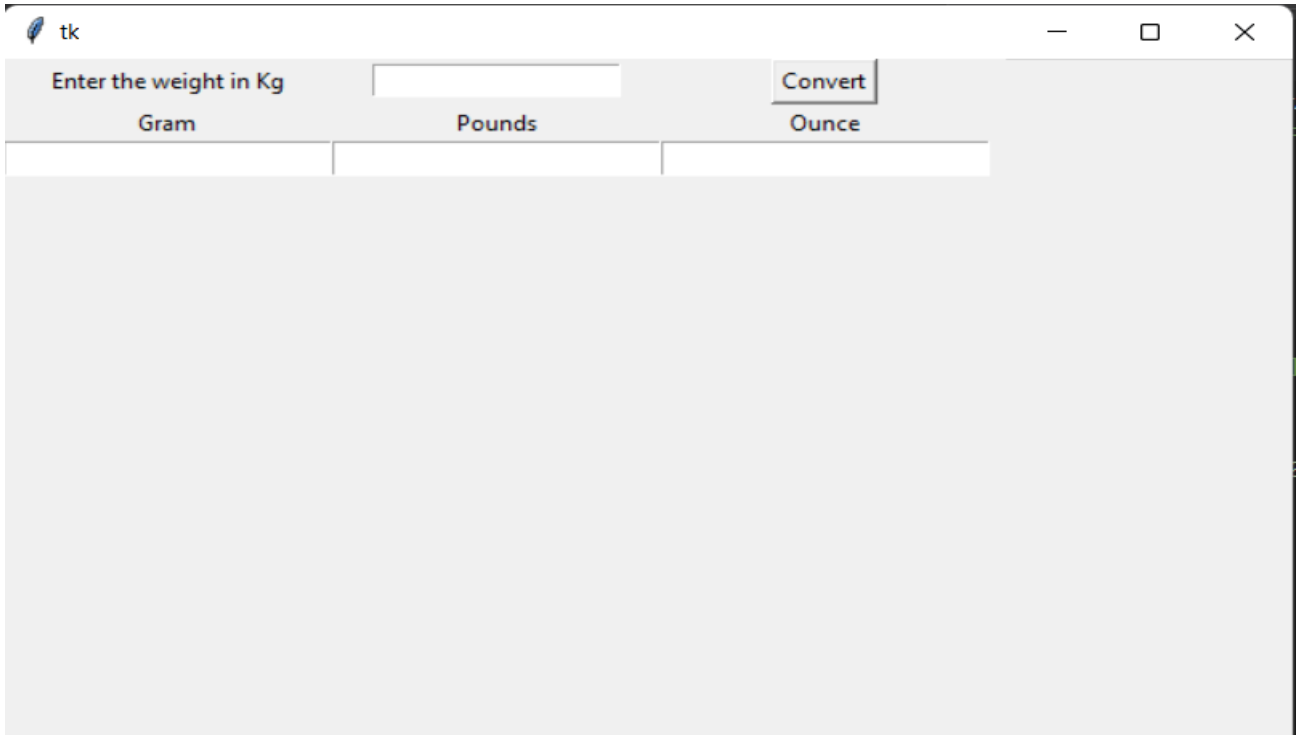

```
        label_prompt = ttk.Label(frame, text=f"Enter the {label_text.lower()}:", style="TLabel")
        entry_value = ttk.Entry(frame)
        button_convert = ttk.Button(frame, text="Convert", command=lambda: convert_func(entry_value, result_widgets, error_label), style="TBut
        button_clear = ttk.Button(frame, text="Clear", command=lambda: self.clear_result_widgets(result_widgets), style="TButton")
        error_label = ttk.Label(frame, text="", foreground="red", style="TLabel")

        # Result Text widgets
        result_widgets = [Text(frame, height=1, width=20) for _ in unit_labels]

        # Grid layout for Converter Section
        label_prompt.grid(row=0, column=0, columnspan=2, pady=5, sticky="w")
        entry_value.grid(row=0, column=2, padx=5, pady=5)
        button_convert.grid(row=0, column=3, padx=5, pady=5)
        button_clear.grid(row=1, column=3, padx=5, pady=5)
        error_label.grid(row=3, column=0, columnspan=4, pady=5, sticky="w")

        # Units labels and result text widgets
        for i, unit_label in enumerate(unit_labels):
            ttk.Label(frame, text=unit_label, style="TLabel").grid(row=2, column=i)
            result_widgets[i].grid(row=1, column=i)

    def create_footer(self):
        # OOP: Method for creating footer label
        footer_label = ttk.Label(self.master, text="© 2023 W&HC-GROUP6.", style="TLabel", background="#f0f0f0", font=("Arial", 8))
        footer_label.grid(row=5, column=0, columnspan=3, pady=(10, 5), sticky="ew")

    def create_recent_results_button(self):
        # OOP: Method for creating button to view recent results
        button_view_results = ttk.Button(self.master, text="View Recent Conversion Results", command=self.view_recent_results, style="TButton"
        button_view_results.grid(row=4, column=0, columnspan=3, pady=10)

    def convert_weight(self, entry_value, result_widgets, error_label):
        # Input Validation: Check for non-negative numeric value
        self.convert(entry_value, result_widgets, error_label, "Weight(kg)", [1000, 2.20462, 35.274])

    def convert_height(self, entry_value, result_widgets, error_label):
        # Input Validation: Check for non-negative numeric value
```




```
        self.convert(entry_value, result_widgets, error_label, "Height(ft)", [30.48, 0.3048, 12])

    def convert(self, entry_value, result_widgets, error_label, label_text, conversion_factors):
        # Input validation, error handling, and actual conversion logic
        try:
            value_str = entry_value.get().strip()
            assert value_str, "Enter a number to convert."

            value = float(value_str)
            assert value >= 0, f"{label_text} cannot be negative."

            results = [value * factor for factor in conversion_factors]

            for widget, val in zip(result_widgets, results):
                self.update_result_widgets(widget, val)

            result_str = f"{label_text}: {value:.2f}\n"

            # Define units for each type of conversion
            unit_labels = ["", "Centimeters(cm)", "Meters(m)", "Inches(in)"] if "Height" in label_text else ["", "Gram(g)", "Pounds(lb)", "Our

            # Create a list of formatted strings for each unit and result
            formatted_results = [f"{unit:<15}: {res:.2f}" for res, unit in zip(results, unit_labels[1:])]

            # Modify the format of the result_str concatenation
            result_str += "\n".join(formatted_results)
            result_str += "\n" + "=" * 30  # Line separator
            self.recent_results.append(result_str)

        except ValueError as error:
            # Error Exception: Handle ValueError during conversion
            self.show_error_message(error, error_label)
        except AssertionError as error:
            # Assert: Handle AssertionError for input validation
            self.show_error_message(error, error_label)

    def update_result_widgets(self, result_widget, value):
```

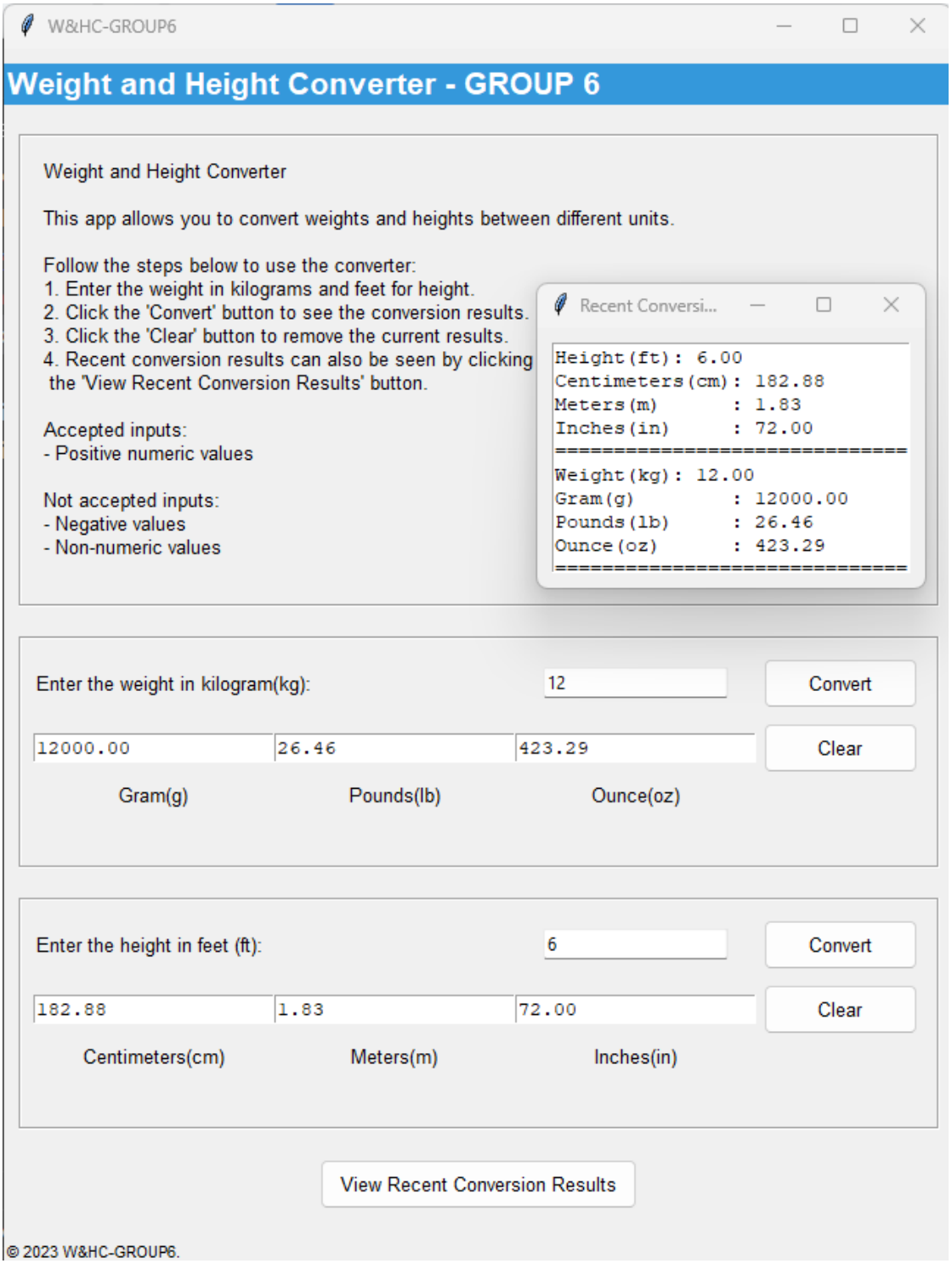



## Original Code



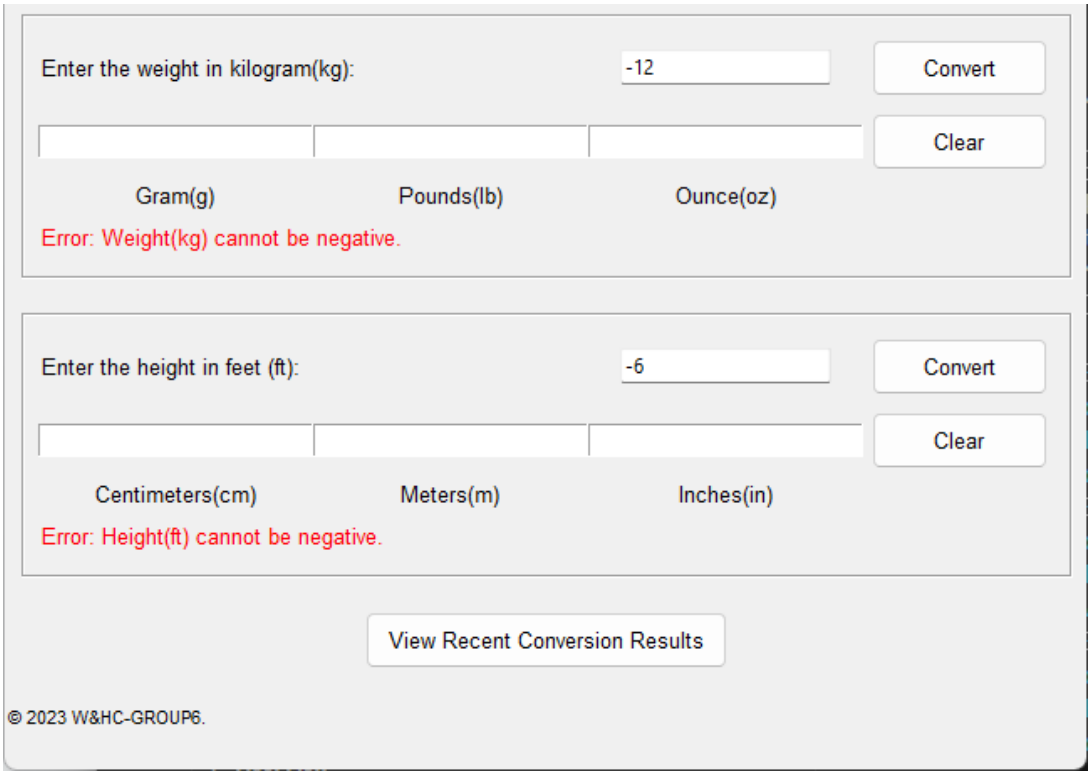
# Modified Code

## ADDITIONAL FEATURES: INSTRUCTIONS, HEIGHT CONVERSION AND RECENT RESULTS

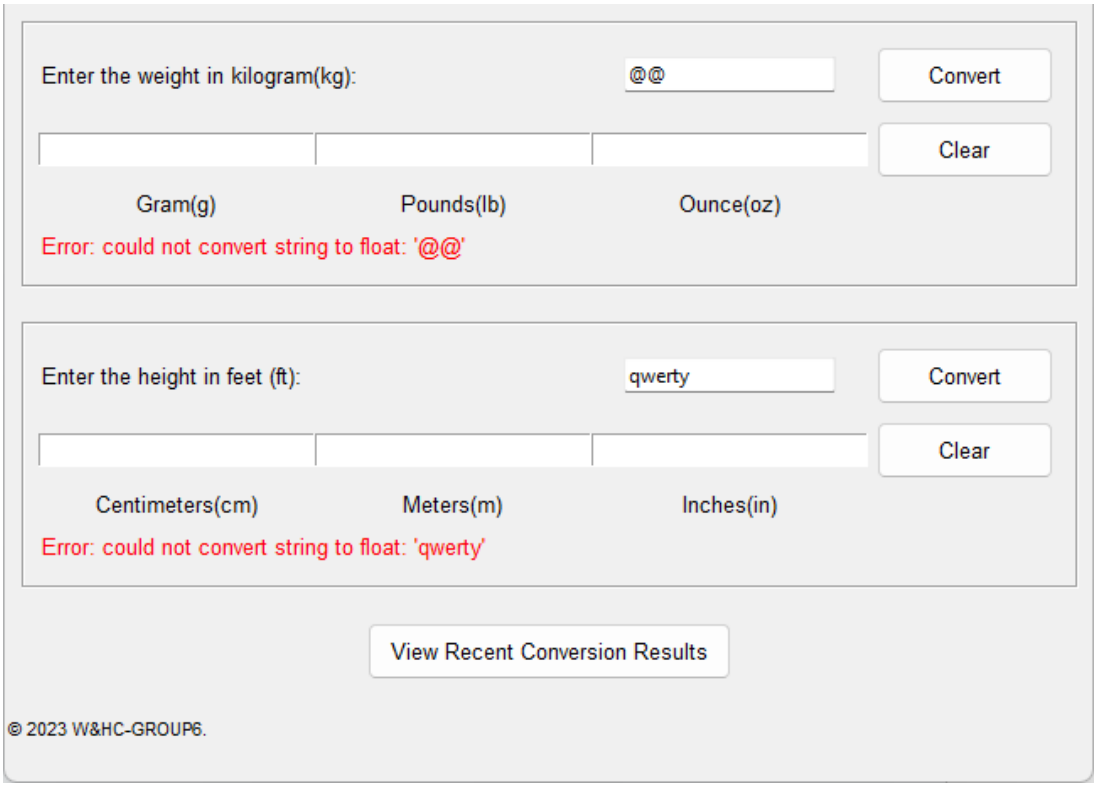


# INPUT VALIDATION and ERROR HANDLING

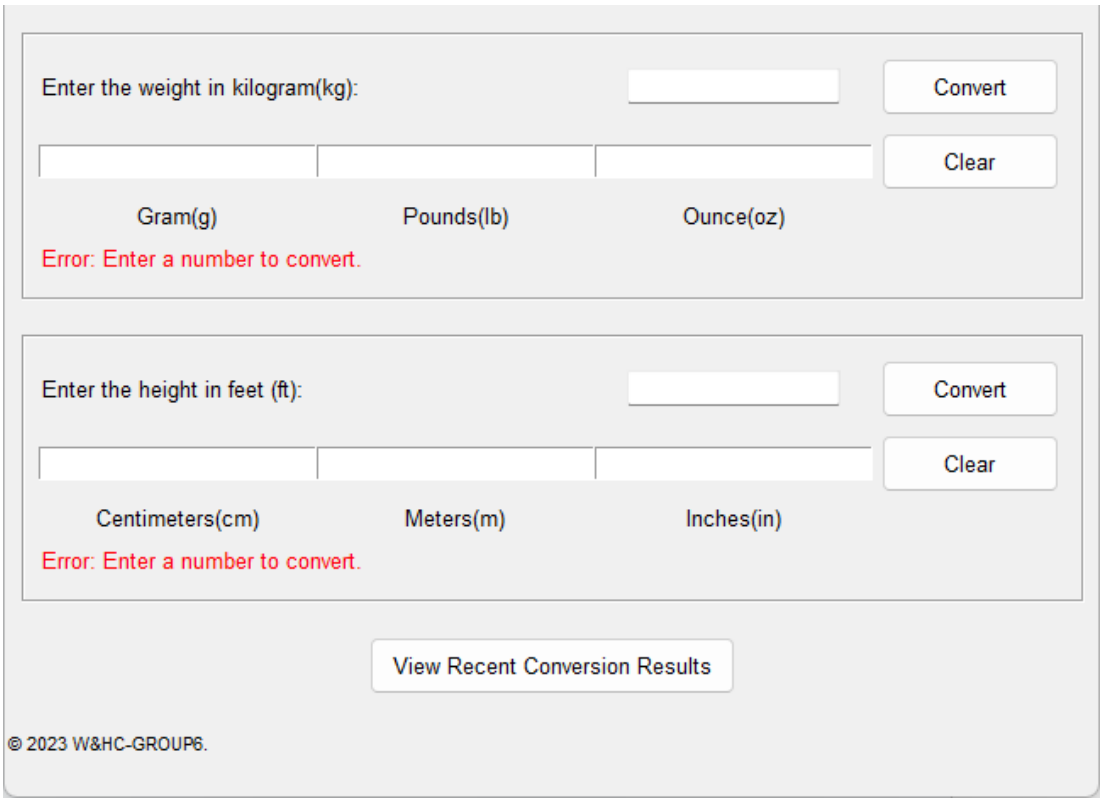
-TRYING TO INPUT A NEGATIVE VALUE




-TRYING TO INPUT NON NUMERIC VALUE



-TRYING TO INPUT EMPTY VALUE




## IV. Results

As a result of this modification, the weight converter program has undergone significant improvements. The code has been restructured using object-oriented programming, resulting in enhanced organization and readability. The graphical user interface (GUI) now features a design, including instructions, footer, clear buttons with and a convenient button for viewing recent conversion results. The application supports both weight and height conversion, with a more dynamic and modular approach. Clear error messages are provided to users for better input validation. Overall, these modifications have transformed the program into a user-friendly tool for converting weights and heights, with improved design and functionality.

## V. Reference

**Original source code:**
https://www.geeksforgeeks.org/python-weight-conversion-gui-using-tkinter