Converter+UI+Comments

September 30, 2025

1 Unit Converter

1.1 Team 6 Members

- Ye Mann Aung
- Elise Collins
- Jainika Punjani
- Hnin Oo Wai
- Thami Silva

1.2 Project Overview

Unit converter built using IPython widgets that allows conversion between different units (Temperature, Length, Weight).

2 Code

2.1 Initial UI Setup

[1]:
```
import ipywidgets as widgets
from IPython.display import display

# Dropdown menu for the UI box with options for conversion.
conversion_type = widgets.Dropdown(
    options=['Temperature', 'Length', 'Weight'],
    description='Type:',
    disabled=False,
)

from_unit = widgets.Dropdown(description='From:')
to_unit = widgets.Dropdown(description='To:')

value_input = widgets.FloatText(
    value=0,
    description='Value:',
    disabled=False
)
```

```
convert_button = widgets.Button(description="Convert")
clear_button = widgets.Button(description="Clear")
result_output = widgets.Output()

# The UI size and type.
box_layout = widgets.Layout(display='flex', flex_flow='column',␣
 ↪align_items='stretch', width='50%')
```

2.2 Layout Setup

This function ensures the unit dropdowns always show appropriate options based on the selected conversion type.

[2]:
```
# Also set unit conversion type based on selections.
def update_units(change):
    if change['new'] == 'Temperature':
        from_unit.options = ['Fahrenheit', 'Celsius']
        to_unit.options = ['Fahrenheit', 'Celsius']
    elif change['new'] == 'Length':
        from_unit.options = ['Miles', 'Kilometers']
        to_unit.options = ['Miles', 'Kilometers']
    elif change['new'] =='Weight':
        from_unit.options = ['Pounds', 'Kilograms']
        to_unit.options = ['Pounds', 'Kilograms']

    from_unit.value = None
    to_unit.value = None

# Look for any updates to the units and update accordingly
conversion_type.observe(update_units, names='value')
update_units({'new': conversion_type.value})
```

2.3 Core Conversion

Handles all conversion calculations with proper error checking and validation.

2.3.1 Conversion Formulas

- **Temperature**:
  - Fahrenheit to Celsius: `(°F - 32) * 5/9`
  - Celsius to Fahrenheit: `(°C × 9/5) + 32`
- **Length**:
  - Miles to Kilometers: `miles * 1.60934`
  - Kilometers to Miles: `km / 1.60934`
- **Weight**:
  - Pounds to Kilograms: `lbs * 0.453592`
  - Kilograms to Pounds: `kg / 0.453592`

[3]:
```
# Main conversion function
def convert_units(b):
    result_output.clear_output()
    with result_output: # Temporarily use the following code.
        try: # Attempt to do a conversion.
            value = value_input.value
            con_type = conversion_type.value
            f_unit = from_unit.value
            t_unit = to_unit.value

            if not all([con_type, f_unit, t_unit]): #if not all of the inputs/
 ↪options are filled.
                print(" Please select all options")
                return

            if f_unit == t_unit: #making sure you don't try to convert␣
 ↪fahrenheit to fahrenheit for example.
                print(" Please select different units")
                return

            # Conversion calculations
            if con_type == 'Temperature':
                if f_unit == 'Fahrenheit' and t_unit == 'Celsius':
                    result = (value - 32) * 5/9
                else:
                    result = (value * 9/5) + 32
            elif con_type == 'Length':
                if f_unit == 'Miles' and t_unit == 'Kilometers':
                    result = value * 1.60934
                else:
                    result = value / 1.60934
            elif con_type == 'Weight':
                if f_unit == 'Pounds' and t_unit == 'Kilograms':
                    result = value * 0.453592
                else:
                    result = value / 0.453592

            print(f" Result: {result:.4f} {t_unit}")

        except Exception as e: #if maybe we gave invalid inputs or text throw␣
 ↪an error.
            print(f" Error: {str(e)}")

# Clear out the function, to let fresh inputs be made for the next conversion
def clear_inputs(b):
    value_input.value = 0
    result_output.clear_output()
```

```
# Bind buttons
convert_button.on_click(convert_units)
clear_button.on_click(clear_inputs)
```

2.4 Drawing the Display

Connect all widgets and create the complete converter interface

[4]:
```
# Setup the UI elements for display
print(" Unit Converter ")
display(conversion_type)
display(from_unit)
display(to_unit)
display(value_input)
display(widgets.HBox([convert_button, clear_button]))
display(result_output)
```

```
 Unit Converter
```

```
Dropdown(description='Type:', options=('Temperature', 'Length', 'Weight'),␣
  ↪value='Temperature')
```

```
Dropdown(description='From:', options=('Fahrenheit', 'Celsius'), value=None)
```

```
Dropdown(description='To:', options=('Fahrenheit', 'Celsius'), value=None)
```

```
FloatText(value=0.0, description='Value:')
```

```
HBox(children=(Button(description='Convert', style=ButtonStyle()),␣
  ↪Button(description='Clear', style=ButtonSty…
```

```
Output()
```