ALL IN ONE CONVERTOR

A Project Work Report

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE
BIG DATA ANALYTICS

Submitted by:

DIGVIJAY KUMAR

AVINASH RAJ

19BCS3878

19BCS3899

Under the Supervision of:

JYOTI MEHRA

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

December, 2020

Name and signature of student(s)

Name and signature of Supervisor

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

PROJECT COMPLETION CERTIFICATE

Project Title

This is to certify that the DIGVIJAY KUMAR AND AVINASH RAJ have successfully completed the project work titled " ALL IN ONE CONVERTOR " *Submitted in the partial fulfilment for the award of the degree of* **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE BIG DATA ANALYTICS**

This project is the record of authentic work carried out during the academic year.

JYOTI MEHRA

Project Guide

Date: 5 DEC 2020

DECLARATION

I the undersigned solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of jyoti mehra. I assert the statements made and conclusions drawn are an outcome of my work. I further certify that the work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/ diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

NAME - DIGVIJAY KUMAR
AVINASH RAJ

UID - 19BCS3878
19BCS3899

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to (Name of your Organization Guide) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.
I would like to express my gratitude towards my parents and my department for their kind co-operation and encouragement which help me in completion of this project.

THANKS AGAIN TO ALL WHO HELPED

Chapter 1: Introduction to project

Chapter 2: Project Requirements (Software/Hardware requirements)

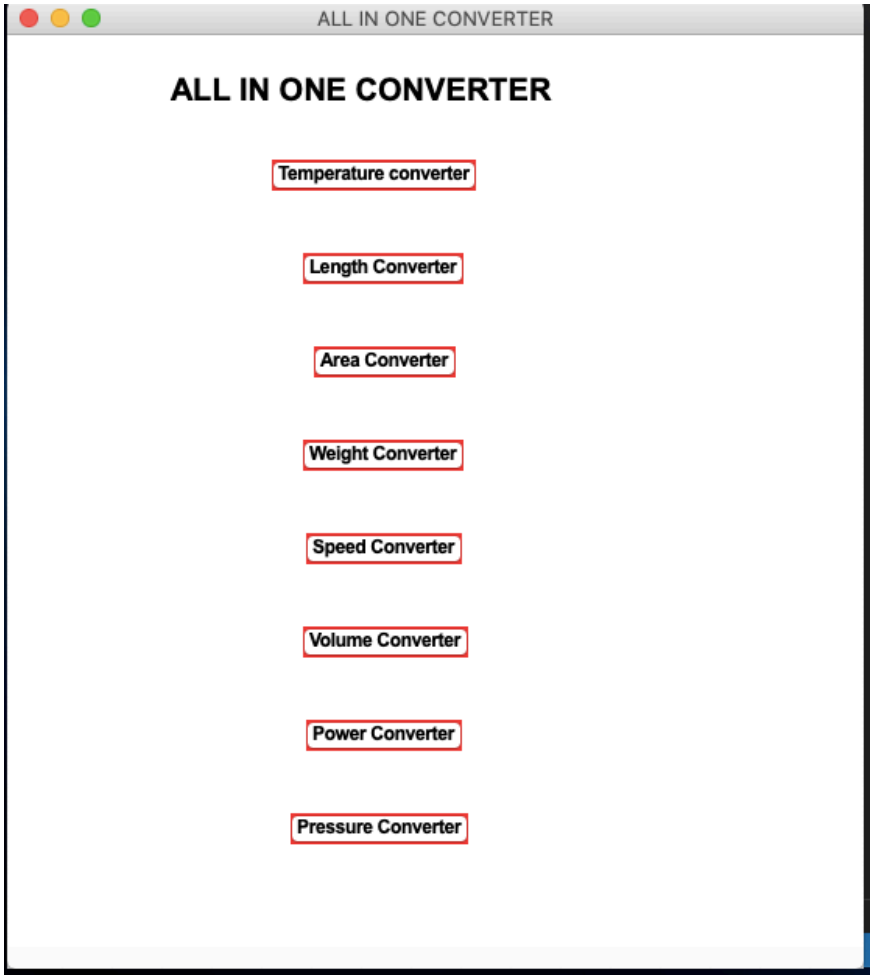
Chapter 3: Implementation Details (Algorithm, code)

Chapter 4: Output Analysis (screenshots)

Introduction to project

This is a all in one convertor in which you can convert different different parameters like speed, power, pressure, volume, weight, area, length, temperature.

We developed this project using python language, however the project is a GUI application that convert different quantities.



Project Requirements (Software/Hardware requirements)

Software

1. Operating system - window 7 or above
   Mac OS X or above
   Linux

2. IDE (working station) - VS code
   Sublime
   Pycharm
   Spyder

Hardware

For Mac - work in all MacBook

For window - Intel pentium 4 or above
   700 MB of harddisk drive
   256 MB of RAM

Implementation Details (Algorithm, code)

ALGORITHM

Step0 : start a function named speed convertor.

Step1 : set IDS for each measurement types.

Step2 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step3 : inside function if frm! “meps”

SET amt = amt*factors[frm]

Return amt/factors[to]

[END of the function]

Step4 : [END of the main function]

Step5 : start a function named power convertor.

Step6 : set IDS for each measurement types.

Step7 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step8 : inside function if frm! = “W”

SET amt = amt*factor[frm]

Return amt/factors[to]

Else return amt/factors[to]

[END of the function].

Step9 : [END of the main function].

Step10 : start a function named pressure convertor.

Step11 : set IDS for each measurement types.

Step12 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step13 : iside function if frm! = “W”

SET amt = amt*factor[frm]

Return amt/factors[to]

Else return amt/factors[to]

[END of the function].

Step14 : [END of the main function].

Step15 : start a function named pressure convertor.

Step16 : set IDS for each measurement types.

Step17 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step18 : inside function if frm! = “1”

SET amt = amt*factor[frm]

Return amt/factor[to]

Else return amt/factors[to]

[END of the function].

Step19 : [END of the main function].

STEP20 : start a function named weight convertor.

Step21 : set IDS for each measurement types.

Step22 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step23 : inside function if frm! = ‘g’

SET amt = amt*factors[frm]

Return amt/factors[to]

Else return amt/factors[to]

[END of the function].

Step24 : [END of the main function].

STEP25 : start a function named area convertor.

Step26 : set IDS for each measurement types.

Step27 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step28 : inside function if fromvar.get() present in meter factor.keys() and to var.get() present in meter factor.keys().

Step29 : result = (float(str(x))*meter factor [from unit] / (meter function [to unit])

resultxt.insert (0, str(result))

[END of the function].

Step30 : [END of the main function].

STEP31 : start a function named length convertor.

Step32 : set IDS for each measurement types.

Step33 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step34 : inside function if frm! = ‘m’

SET amt = amt*factor[frm]

Return amt/factors[to]

Else return amt/factors[to]

[END of the function].

Step35 : [END of the main function].

STEP36 : start a function named temperature convertor.

Step37 : set IDS for each measurement types.

Step38 : write a function named “CONVERT” with parameters “amount”, “from”, “to”.

Step39 : inside function if celTempvar.get()! = 0.0

celToFah = (celTemp*9/5 + 32)

fahTempvar.set(celToFah)

Elif fahTempvar.get()! = 0.0

fahToCel = ((fahTemp - 32)*(5/9))

celTempvar.set(fahToCel)

[END of the function].

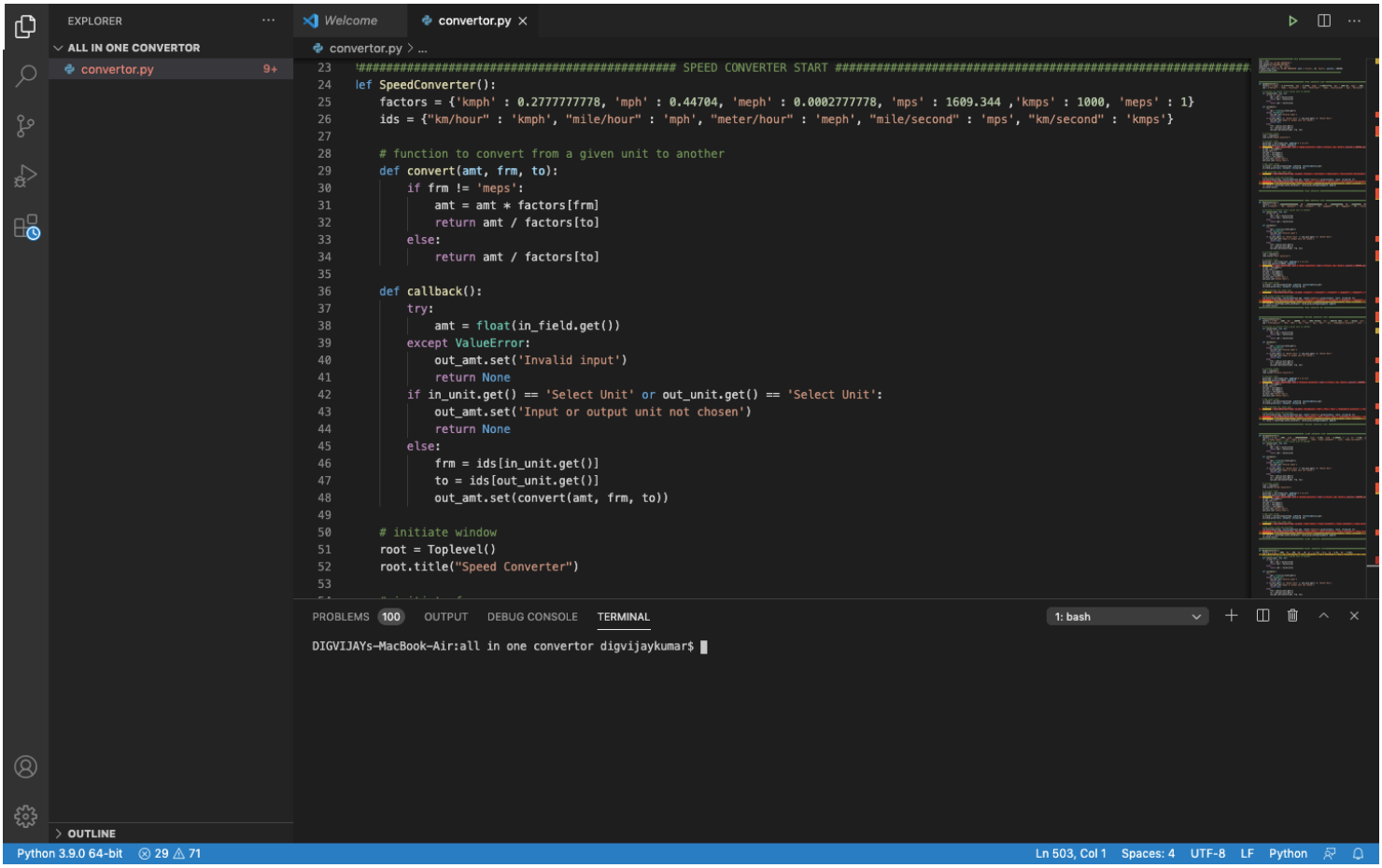
Step40 : [END of the main function].

CODE

```
import sys
import tkinter as tk
from tkinter import *
import urllib.request
import webbrowser
from functools import partial
from tkinter import Tk, StringVar , ttk



#################################### MAIN
########################################
root = Tk()
root.title('ALL IN ONE CONVERTER')
root.geometry("550x600+200+300")
labelfont = ('ariel', 56, 'bold')
l=Label(root,text='ALL IN ONE CONVERTER',font = ("Arial", 20,
"bold"), justify = CENTER)
l.place(x=100,y=20)
############################################################
####################



############################################## SPEED CONVERTER
START
############################################################
#######
def SpeedConverter():
    factors = {'kmph' : 0.2777777778, 'mph' : 0.44704, 'meph' :
0.0002777778, 'mps' : 1609.344 ,'kmps' : 1000, 'meps' : 1}
    ids = {"km/hour" : 'kmph', "mile/hour" : 'mph', "meter/
hour" : 'meph', "mile/second" : 'mps', "km/second" : 'kmps'}

    # function to convert from a given unit to another
    def convert(amt, frm, to):
```

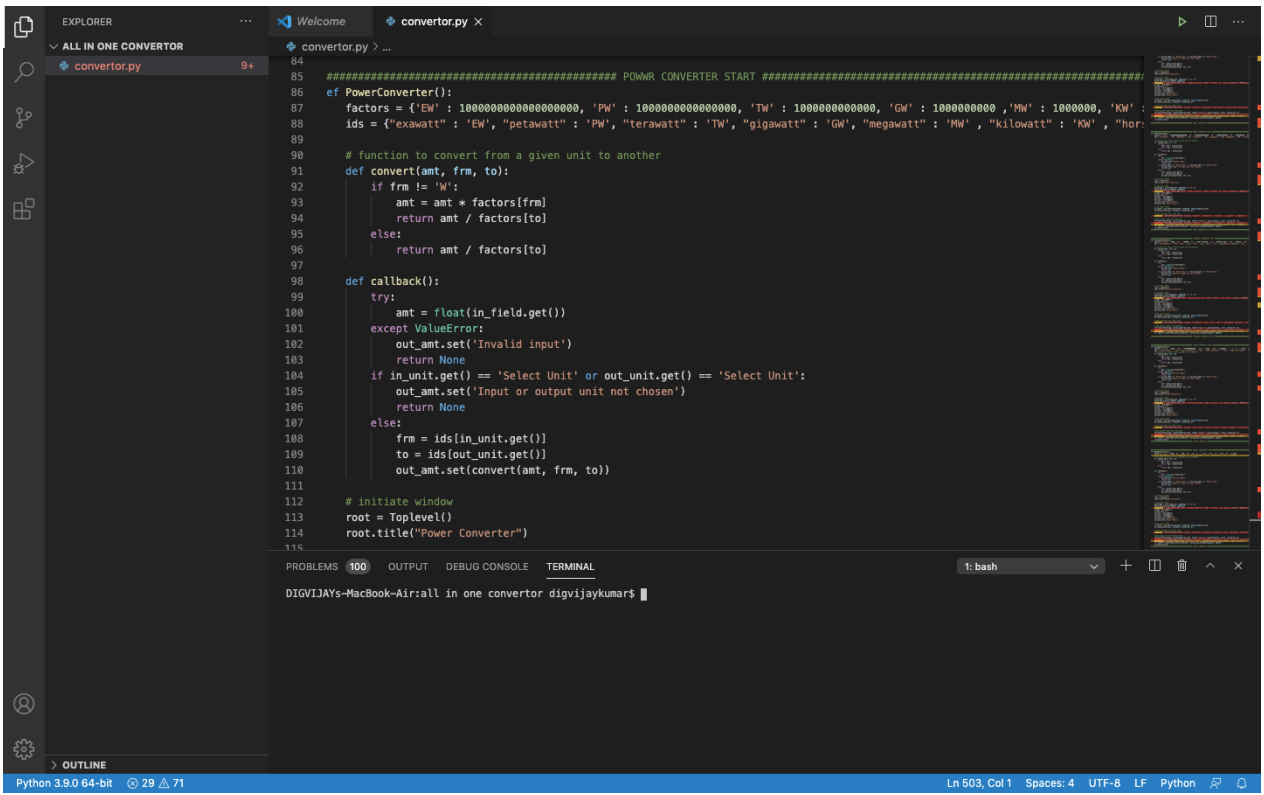
```
        if frm != 'meps':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Speed Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
    titleLabel = Label (mainframe, text = "Speed Converter", font
= ("Arial", 12, "bold"), justify = CENTER).grid(column=1,row=1)
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()
    in_unit = StringVar()
    out_unit = StringVar()
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit, "km/hour", "mile/
hour", "meter/hour", "mile/second","km/second") .grid(column=3,
row=1, sticky=W)
```

```
    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit, "km/hour", "mile/
hour", "meter/hour", "mile/second","km/second").grid(column=3,
row=3, sticky=W)
    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)
    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)
    in_field.focus()

################################################ SPEED CONVERTER
END#################################################################
##########
```



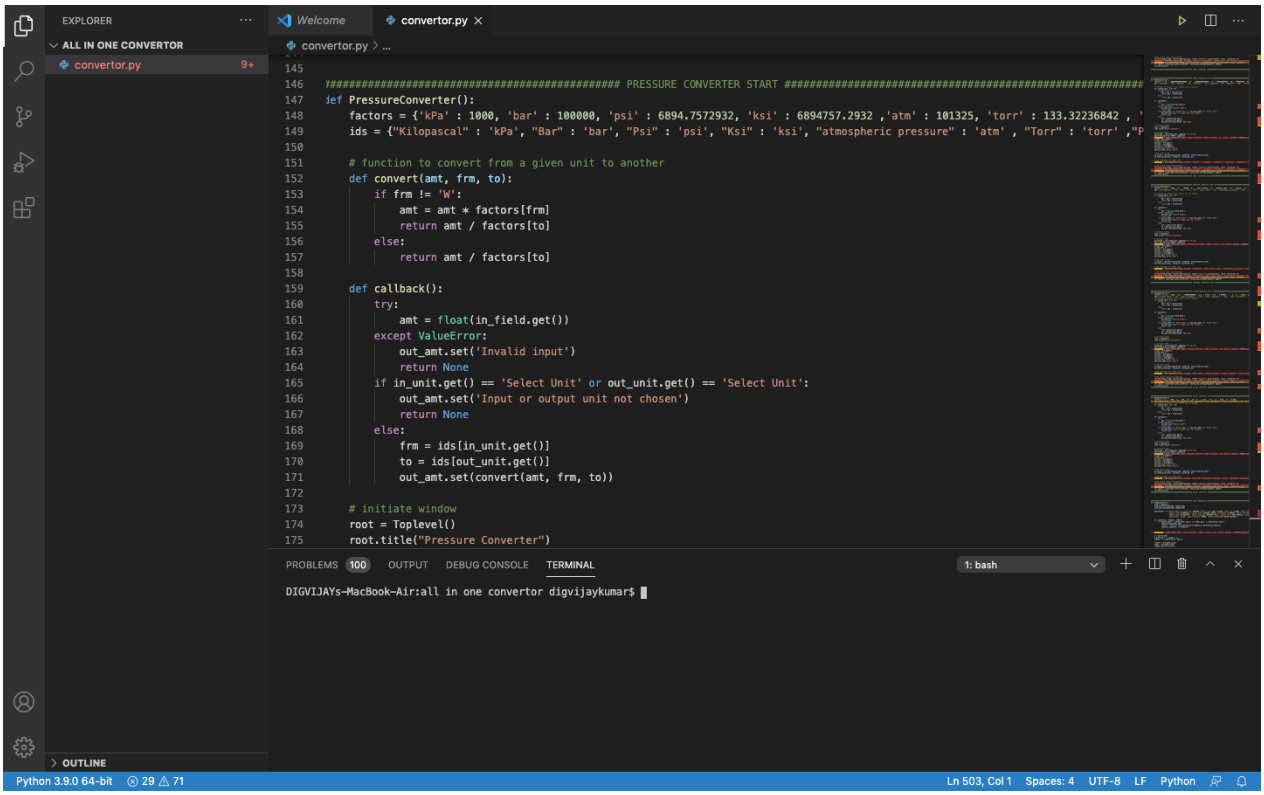
```
############################################## POWWR CONVERTER
START
#################################################################
#######
def PowerConverter():
    factors = {'EW' : 1000000000000000000, 'PW' :
1000000000000000, 'TW' : 1000000000000, 'GW' : 1000000000 ,'MW' :
1000000, 'KW' : 1000 , 'HP' : 746 , 'W' : 1}
    ids = {"exawatt" : 'EW', "petawatt" : 'PW', "terawatt" :
'TW', "gigawatt" : 'GW', "megawatt" : 'MW' , "kilowatt" : 'KW' ,
"horsepower" : 'HP' , "watt" : 'W'}

    # function to convert from a given unit to another
    def convert(amt, frm, to):
        if frm != 'W':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Power Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
    titleLabel = Label (mainframe, text = "Power Converter", font
= ("Arial", 12, "bold"), justify = CENTER).grid(column=1,row=1)
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()
```

```
    in_unit = StringVar()
    out_unit = StringVar()
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit, "exawatt" ,
"petawatt" , "terawatt" , "gigawatt" , "megawatt" , "kilowatt",
"horsepower" , "watt") .grid(column=3, row=1, sticky=W)

    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit, "exawatt" ,
"petawatt" , "terawatt" , "gigawatt" , "megawatt" , "kilowatt",
"horsepower" , "watt").grid(column=3, row=3, sticky=W)
    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)
    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)
    in_field.focus()
############################################## POWER CONVERTER
END
#########################################################################
#######
```



```
############################################## PRESSURE
CONVERTER START
#################################################################
#######
def PressureConverter():
    factors = {'kPa' : 1000, 'bar' : 100000, 'psi' :
6894.7572932, 'ksi' : 6894757.2932 ,'atm' : 101325, 'torr' :
133.32236842 , 'Pa' :1}
    ids = {"Kilopascal" : 'kPa', "Bar" : 'bar', "Psi" : 'psi',
"Ksi" : 'ksi', "atmospheric pressure" : 'atm' , "Torr" :
'torr' ,"Pascal" : 'Pa'}

    # function to convert from a given unit to another
    def convert(amt, frm, to):
        if frm != 'W':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Pressure Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
```

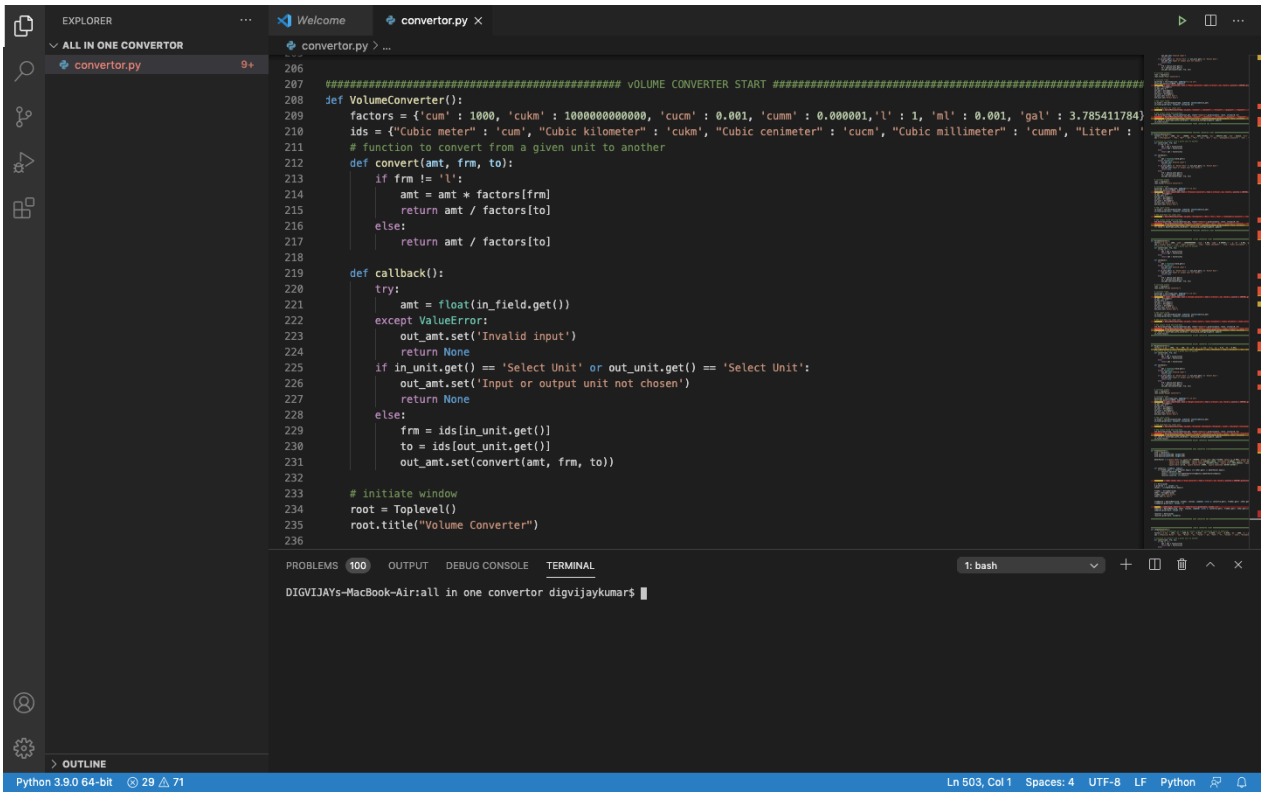
```
    titleLabel = Label (mainframe, text = "Pressure Converter",
font = ("Arial", 12, "bold"), justify =
CENTER).grid(column=1,row=1)
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()
    in_unit = StringVar()
    out_unit = StringVar()
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit, "Kilopascal",
"Bar", "Psi", "Ksi" , "atmospheric pressure" ,
"Torr" ,"Pascal") .grid(column=3, row=1, sticky=W)

    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit, "Kilopascal",
"Bar", "Psi", "Ksi" , "atmospheric pressure" ,
"Torr" ,"Pascal").grid(column=3, row=3, sticky=W)
    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)
    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)

############################################## PRESSURE
CONVERTER START
#########################################################################
#######
```




```
########################################### vOLUME CONVERTER START
#########################################################################
def VolumeConverter():
    factors = {'cum' : 1000, 'cukm' : 1000000000000, 'cucm' : 0.001, 'cumm' : 0.000001,'l' : 1, 'ml' : 0.001, 'gal' : 3.785411784}
    ids = {"Cubic meter" : 'cum', "Cubic kilometer" : 'cukm', "Cubic cenimeter" : 'cucm', "Cubic millimeter" : 'cumm', "Liter" : 'l', "Milliliter" : 'ml', "gallon" : 'gal'}
    # function to convert from a given unit to another
    def convert(amt, frm, to):
        if frm != 'l':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
```

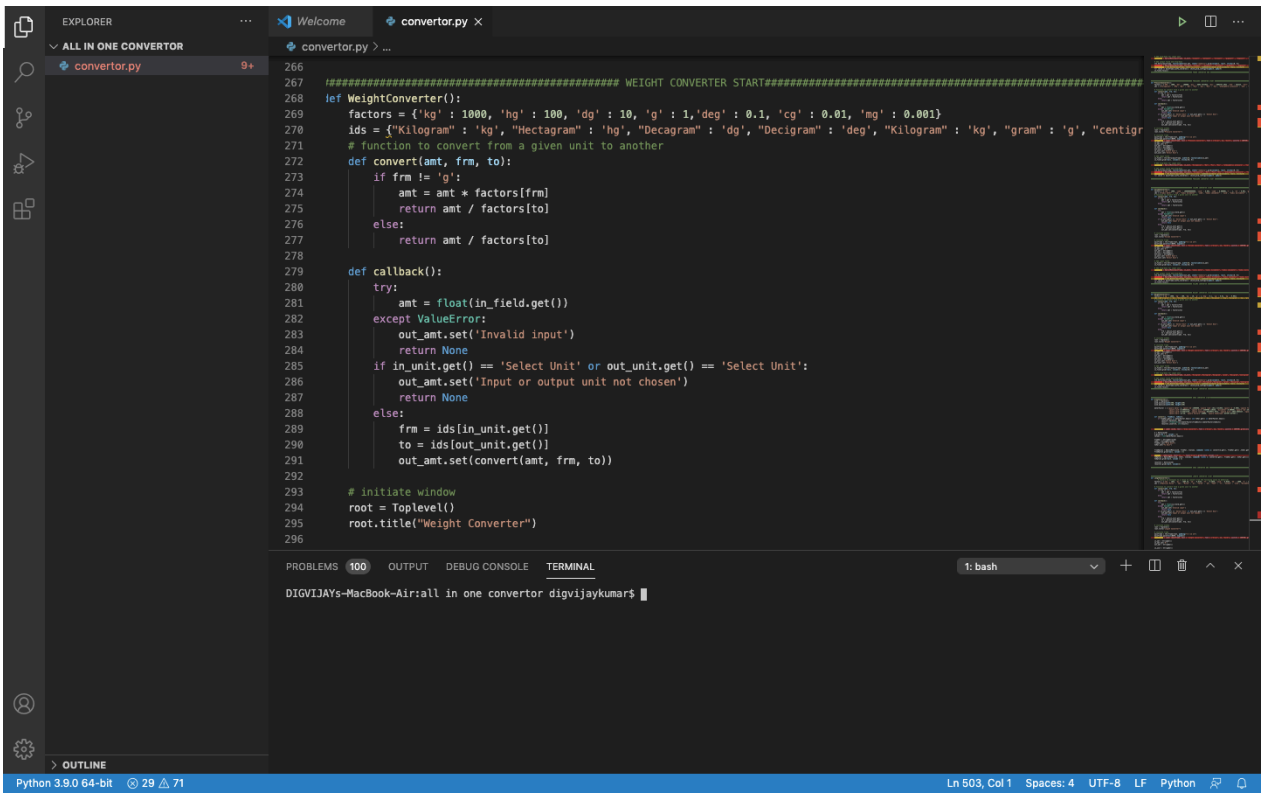
```
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Volume Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
    titleLabel = Label (mainframe, text = "Volume Converter",
font = ("Arial", 12, "bold"), justify =
CENTER).grid(column=1,row=1)
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()
    in_unit = StringVar()
    out_unit = StringVar()
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit, "Cubic meter",
"Cubic kilometer", "Cubic cenimeter", "Cubic millimeter",
"Liter", "Milliliter", "gallon") .grid(column=3, row=1, sticky=W)

    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit, "Cubic meter",
"Cubic kilometer", "Cubic cenimeter", "Cubic millimeter",
"Liter", "Milliliter", "gallon").grid(column=3, row=3, sticky=W)
```

```
    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)
    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)
    in_field.focus()
########################################### VOLUME CONVERTER
END#############################################################
##########
```



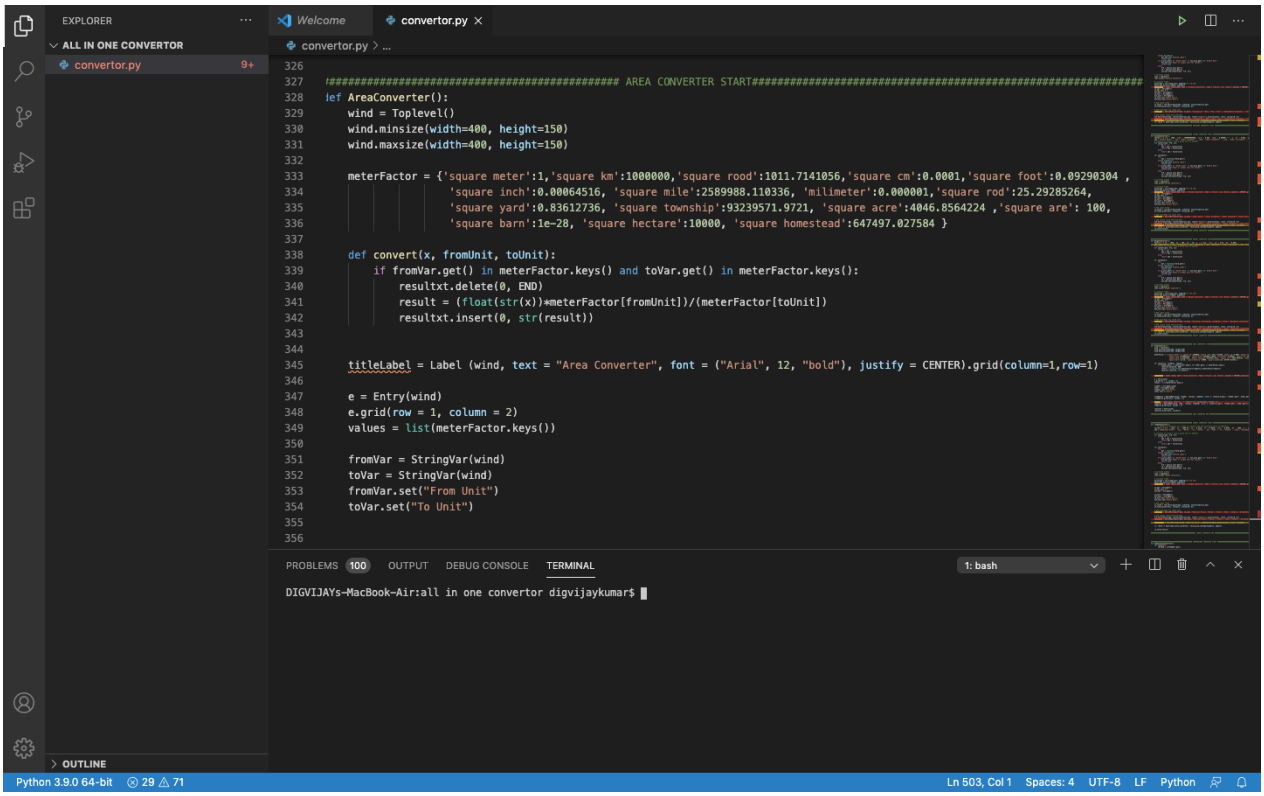
```
############################################## WEIGHT CONVERTER
START#############################################################
############
def WeightConverter():
    factors = {'kg' : 1000, 'hg' : 100, 'dg' : 10, 'g' :
1,'deg' : 0.1, 'cg' : 0.01, 'mg' : 0.001}
    ids = {"Kilogram" : 'kg', "Hectagram" : 'hg', "Decagram" :
'dg', "Decigram" : 'deg', "Kilogram" : 'kg', "gram" : 'g',
"centigram" : 'cg', "milligram" : 'mg'}
    # function to convert from a given unit to another
    def convert(amt, frm, to):
        if frm != 'g':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Weight Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
    titleLabel = Label (mainframe, text = "Weight Converter",
font = ("Arial", 12, "bold"), justify =
CENTER).grid(column=1,row=1)
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()
    in_unit = StringVar()
    out_unit = StringVar()
```

```
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit,
"Kilogram","Hectagram","Decagram", "gram",
"Decigram","Centigram", "Milligram") .grid(column=3, row=1,
sticky=W)

    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit,
"Kilogram","Hectagram","Decagram", "gram",
"Decigram","Centigram", "Milligram").grid(column=3, row=3,
sticky=W)
    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)
    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)
    in_field.focus()
########################################### WEIGHT CONVERTER
END#############################################################
##########
```



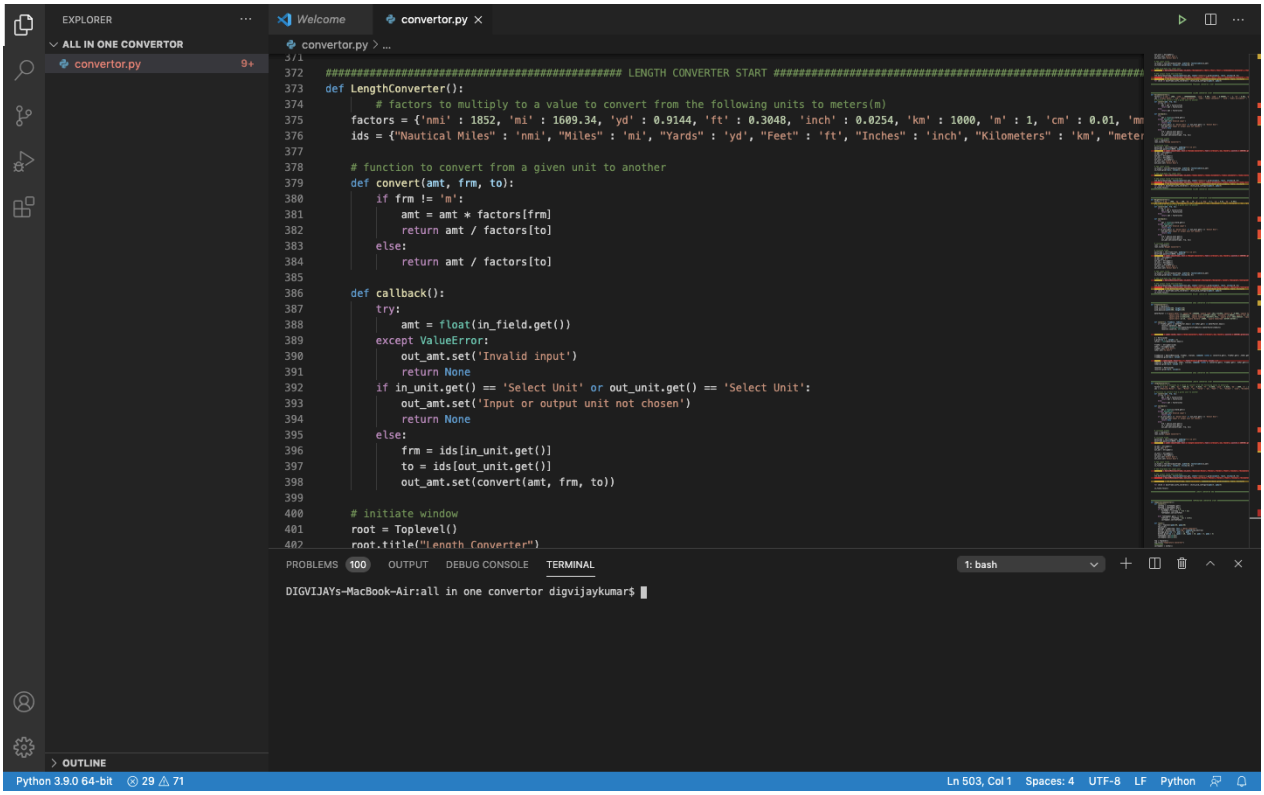
```
############################################## AREA CONVERTER
START#############################################################
############
def AreaConverter():
    wind = Toplevel()
    wind.minsize(width=400, height=150)
    wind.maxsize(width=400, height=150)

    meterFactor = {'square meter':1,'square km':1000000,'square
rood':1011.7141056,'square cm':0.0001,'square foot':0.09290304 ,
                   'square inch':0.00064516, 'square mile':
2589988.110336, 'milimeter':0.000001,'square rod':25.29285264,
                   'square yard':0.83612736, 'square township':
93239571.9721, 'square acre':4046.8564224 ,'square are': 100,
                   'square barn':1e-28, 'square hectare':10000,
'square homestead':647497.027584 }

    def convert(x, fromUnit, toUnit):
        if fromVar.get() in meterFactor.keys() and toVar.get() in
meterFactor.keys():
            resultxt.delete(0, END)
            result = (float(str(x))*meterFactor[fromUnit])/
(meterFactor[toUnit])
            resultxt.insert(0, str(result))


    titleLabel = Label (wind, text = "Area Converter", font =
("Arial", 12, "bold"), justify = CENTER).grid(column=1,row=1)

    e = Entry(wind)
    e.grid(row = 1, column = 2)
    values = list(meterFactor.keys())

    fromVar = StringVar(wind)
    toVar = StringVar(wind)
    fromVar.set("From Unit")
    toVar.set("To Unit")


    fromOption = OptionMenu(wind, fromVar, *values, command=
lambda y: convert(e.get(), fromVar.get() ,toVar.get()))
    fromOption.grid(row=1, column = 3)

    toLabel = Label(wind, text="To : ", font="Arial").grid(row=2,
column = 2)
    toOption = OptionMenu(wind, toVar, *values, command= lambda
x: convert(e.get(), fromVar.get() ,toVar.get()))
```

```
    toOption.grid(row=3, column = 3)

    resultxt = Entry(wind)
    resultxt.grid(row=3, column=2)

########################################### AREA CONVERTER END
#########################################################################
#######
```



```
############################################## LENGTH CONVERTER
START
#########################################################################
#######
def LengthConverter():
        # factors to multiply to a value to convert from the
following units to meters(m)
    factors = {'nmi' : 1852, 'mi' : 1609.34, 'yd' : 0.9144,
'ft' : 0.3048, 'inch' : 0.0254, 'km' : 1000, 'm' : 1, 'cm' :
0.01, 'mm' : 0.001}
    ids = {"Nautical Miles" : 'nmi', "Miles" : 'mi', "Yards" :
'yd', "Feet" : 'ft', "Inches" : 'inch', "Kilometers" : 'km',
"meters" : 'm', "centimeters" : 'cm', "millileters" : 'mm'}

    # function to convert from a given unit to another
    def convert(amt, frm, to):
        if frm != 'm':
            amt = amt * factors[frm]
            return amt / factors[to]
        else:
            return amt / factors[to]

    def callback():
        try:
            amt = float(in_field.get())
        except ValueError:
            out_amt.set('Invalid input')
            return None
        if in_unit.get() == 'Select Unit' or out_unit.get() ==
'Select Unit':
            out_amt.set('Input or output unit not chosen')
            return None
        else:
            frm = ids[in_unit.get()]
            to = ids[out_unit.get()]
            out_amt.set(convert(amt, frm, to))

    # initiate window
    root = Toplevel()
    root.title("Length Converter")

    # initiate frame
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.pack(fill=BOTH, expand=1)
    titleLabel = Label (mainframe, text = "Length Converter",
font = ("Arial", 12, "bold"), justify =
CENTER).grid(column=1,row=1)
```

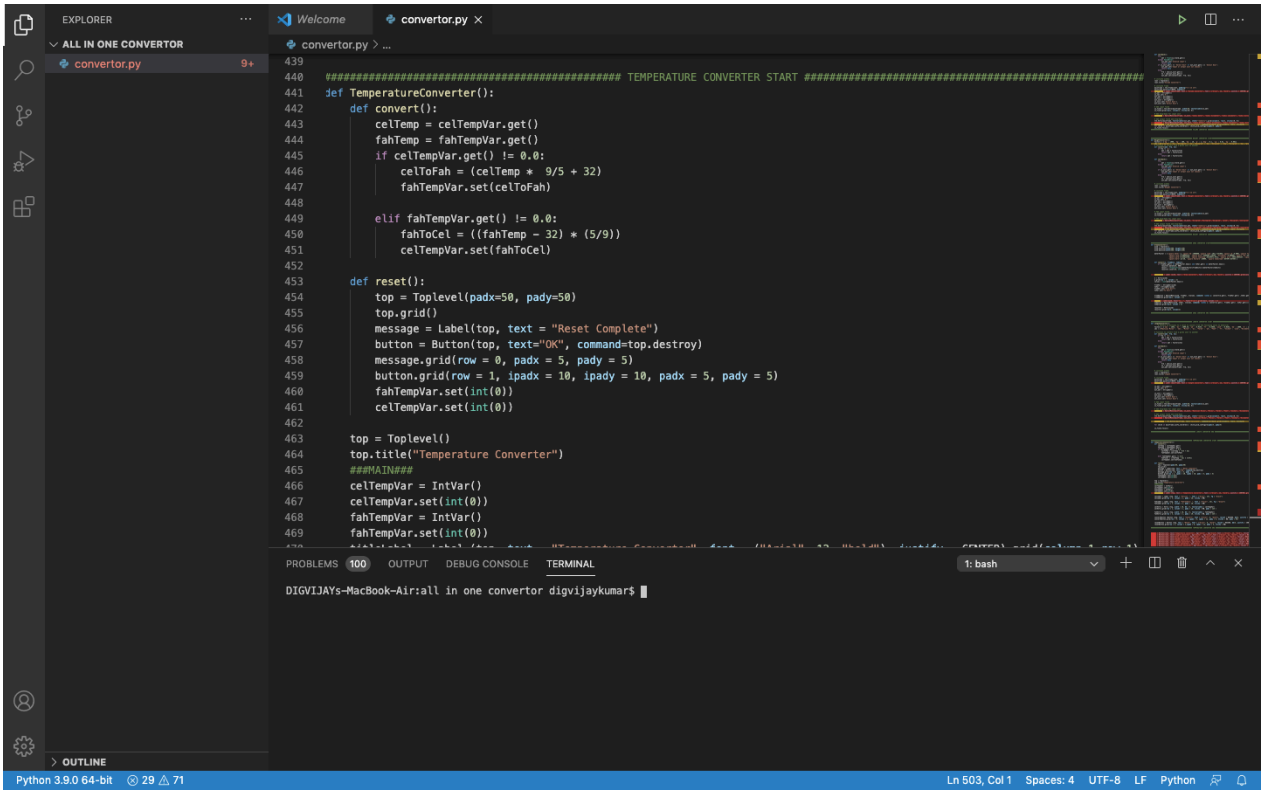
```
    in_amt = StringVar()
    in_amt.set('0')
    out_amt = StringVar()

    in_unit = StringVar()
    out_unit = StringVar()
    in_unit.set('Select Unit')
    out_unit.set('Select Unit')

    # Add input field
    in_field = ttk.Entry(mainframe, width=20,
textvariable=in_amt)
    in_field.grid(row=1, column=2, sticky=(W, E))

    # Add drop-down for input unit
    in_select = OptionMenu(mainframe, in_unit, "Nautical Miles",
"Miles", "Yards", "Feet", "Inches", "Kilometers", "meters",
"centimeters", "millileters").grid(column=3, row=1, sticky=W)

    # Add output field and drop-down
    ttk.Entry(mainframe, textvariable=out_amt,
state="readonly").grid(column=2, row=3, sticky=(W, E))
    in_select = OptionMenu(mainframe, out_unit, "Nautical Miles",
"Miles", "Yards", "Feet", "Inches", "Kilometers", "meters",
"centimeters", "millileters").grid(column=3, row=3, sticky=W)

    calc_button = ttk.Button(mainframe, text="Calculate",
command=callback).grid(column=2, row=2, sticky=E)

    for child in mainframe.winfo_children():
child.grid_configure(padx=5, pady=5)

    in_field.focus()

################################################## LENGTH
CONVERTER END
#################################################################
##########
```




```
########################################## TEMPERATURE CONVERTER START ##########################################
def TemperatureConverter():
    def convert():
        celTemp = celTempVar.get()
        fahTemp = fahTempVar.get()
        if celTempVar.get() != 0.0:
            celToFah = (celTemp *  9/5 + 32)
            fahTempVar.set(celToFah)

        elif fahTempVar.get() != 0.0:
            fahToCel = ((fahTemp - 32) * (5/9))
            celTempVar.set(fahToCel)

    def reset():
        top = Toplevel(padx=50, pady=50)
        top.grid()
        message = Label(top, text = "Reset Complete")
```

```
        button = Button(top, text="OK", command=top.destroy)
        message.grid(row = 0, padx = 5, pady = 5)
        button.grid(row = 1, ipadx = 10, ipady = 10, padx = 5,
pady = 5)
        fahTempVar.set(int(0))
        celTempVar.set(int(0))

    top = Toplevel()
    top.title("Temperature Converter")
    ###MAIN###
    celTempVar = IntVar()
    celTempVar.set(int(0))
    fahTempVar = IntVar()
    fahTempVar.set(int(0))
    titleLabel = Label (top, text = "Temperature Converter", font
= ("Arial", 12, "bold"), justify = CENTER).grid(column=1,row=1)

    celLabel = Label (top, text = "Celcius: ", font = ("Arial",
14), fg = "black")
    celLabel.grid(row = 2, column = 1, pady = 10, sticky = NW)

    fahLabel = Label (top, text = "Fahrenheit: ", font =
("Arial", 14), fg = "black")
    fahLabel.grid(row = 3, column = 1, pady = 10, sticky = NW)

    celEntry = Entry (top, width = 10, bd = 5, textvariable =
celTempVar)
    celEntry.grid(row = 2, column = 1, pady = 10, sticky = NW,
padx = 125 )

    fahEntry = Entry (top, width = 10, bd = 5, textvariable =
fahTempVar)
    fahEntry.grid(row = 3, column = 1, pady = 10, sticky = NW,
padx = 125 )

    convertButton =Button (top, text = "Convert", font =
("Arial", 8, "bold"), relief = RAISED, bd=5, justify = CENTER,
highlightbackground = "red", overrelief = GROOVE,
activebackground = "green", activeforeground="blue", command =
convert)
    convertButton.grid(row = 4, column = 1, ipady = 8, ipadx =
12, pady = 5, sticky = NW, padx = 55)

    resetButton = Button (top, text = "Reset", font = ("Arial",
8, "bold"), relief = RAISED, bd=5, justify = CENTER,
highlightbackground = "red", overrelief = GROOVE,
```

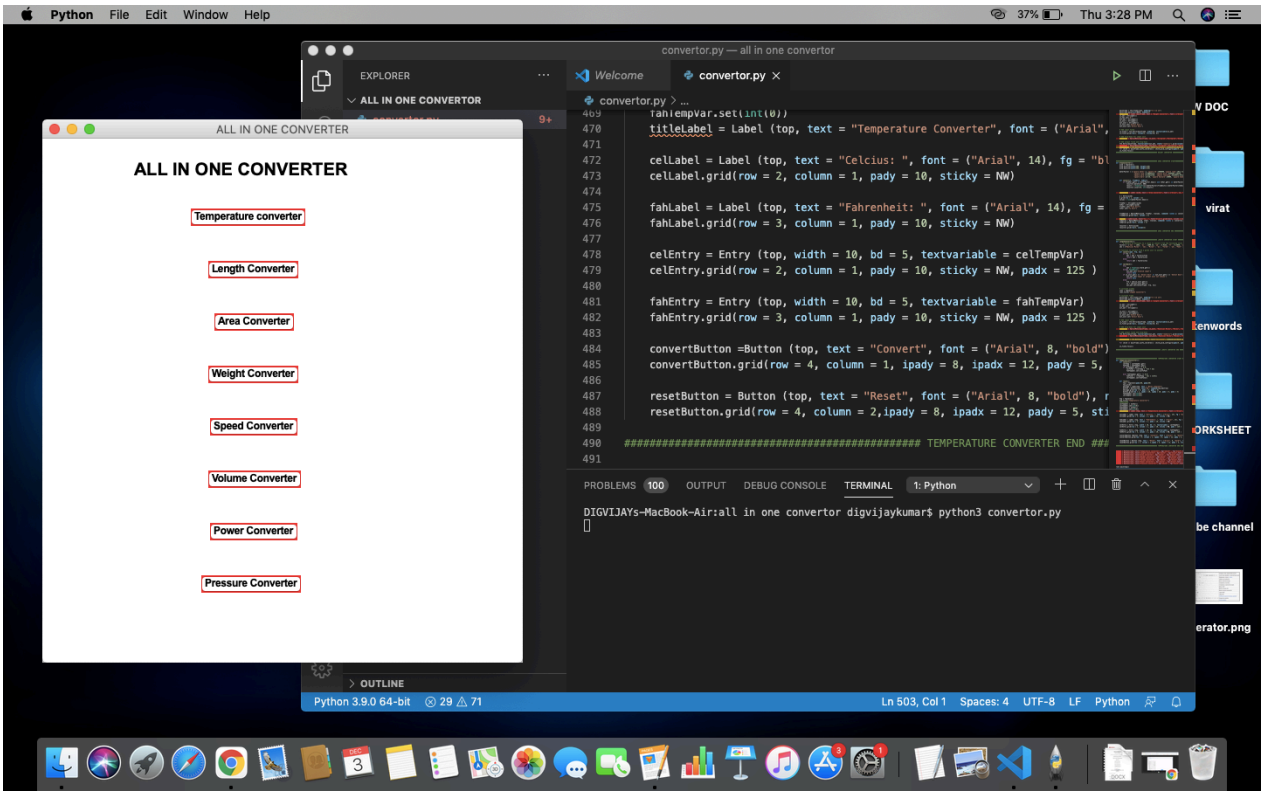
```
activebackground = "green", activeforeground="blue", command =
reset)
    resetButton.grid(row = 4, column = 2,ipady = 8, ipadx = 12,
pady = 5, sticky = NW)

########################################### TEMPERATURE
CONVERTER END
#########################################################################
#######
```



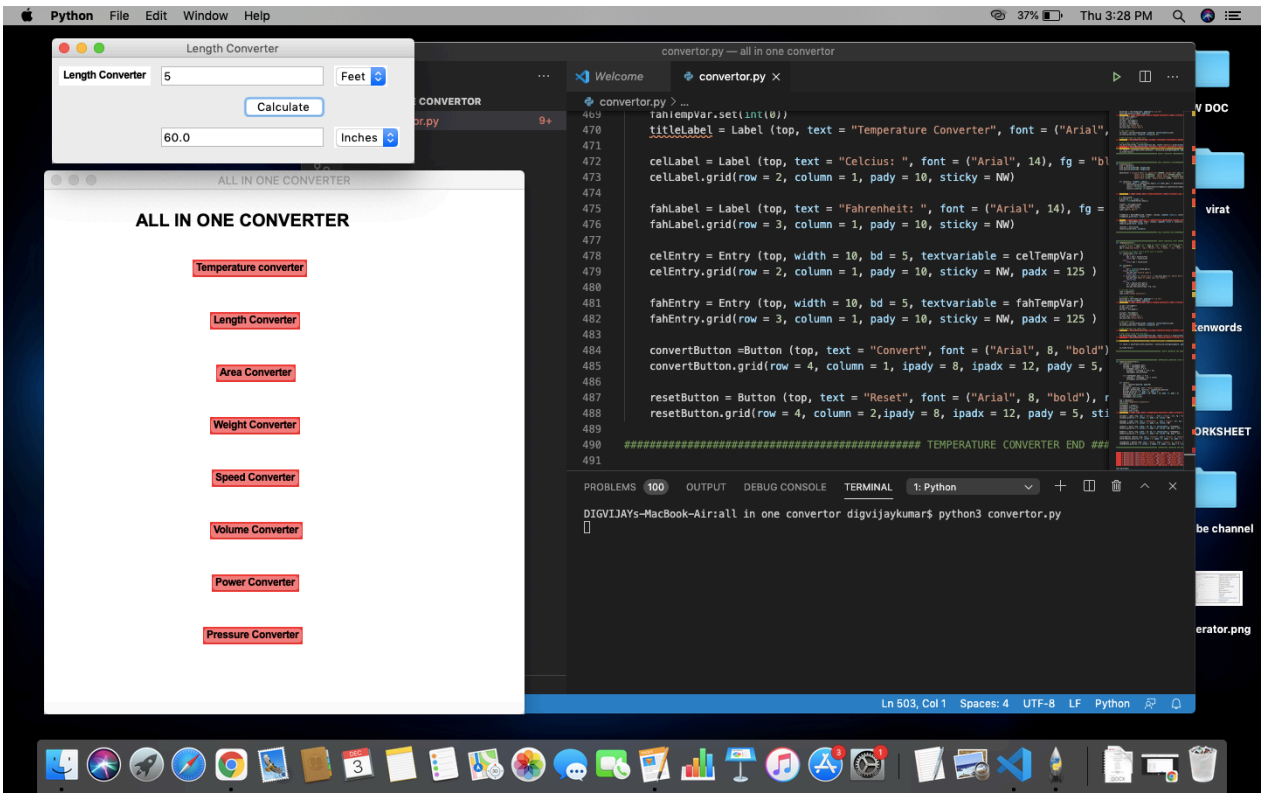

```
widget = Button(root, text="Temperature converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=TemperatureConverter).place(x=170,y=80)
widget = Button(root, text="Length Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=LengthConverter).place(x=190,y=140)
```

```
widget = Button(root, text="Area Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=AreaConverter).place(x=197,y=200)
widget = Button(root, text="Weight Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=WeightConverter).place(x=190,y=260)
widget = Button(root, text="Speed Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=SpeedConverter).place(x=192,y=320)
widget = Button(root, text="Volume Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=VolumeConverter).place(x=190,y=380)
widget = Button(root, text="Power Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=PowerConverter).place(x=192,y=440)
widget = Button(root, text="Pressure Converter", bg="white" ,
fg="black",font = ("Arial", 12, "bold"), relief = RAISED, bd=5,
justify = CENTER, highlightbackground = "red", overrelief =
GROOVE, activebackground = "green", activeforeground="blue",
command=PressureConverter).place(x=182,y=500)

root.mainloop()
```

Output Analysis (screenshots)



Here we convert 5 feet into 60 inches through length convertor. Similarly we can convert other quantities.



THANK YOU