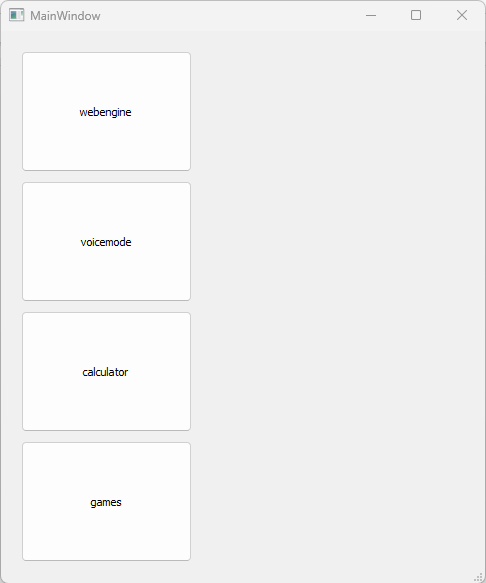
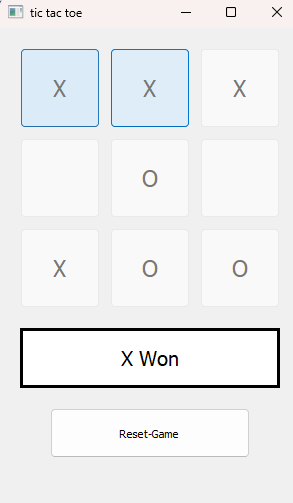
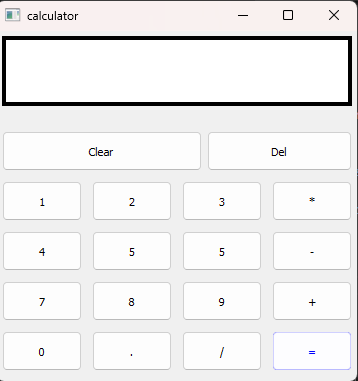
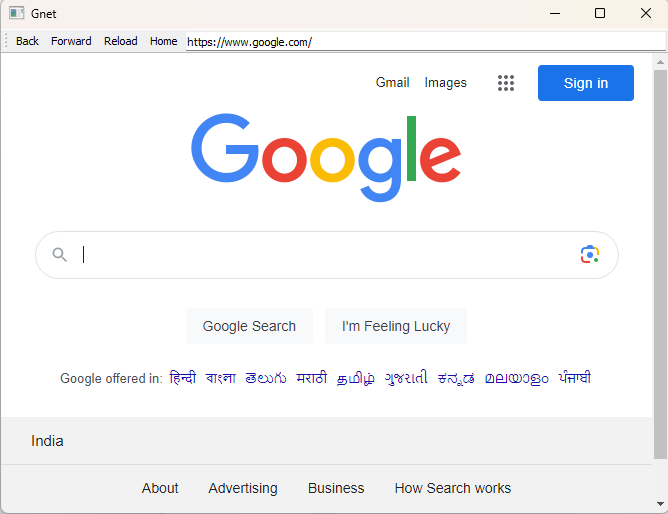
HOME PAGE







Main,.py

from PyQt5.QtWidgets import \*

from PyQt5 import QtCore, QtGui

from PyQt5.QtGui import \*

from PyQt5.QtCore import \*

import sys

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import \*

from PyQt5.QtWebEngineWidgets import \*

from mic import Ui\_SubWindow

class Ui\_MainWindow(object):

    def openwindow2(self):

        self.window = QtWidgets.QMainWindow()

        self.ui = Ui\_SubWindow()

        self.ui.setupUi(self.window)

        self.window.show()

    def openwindow1(self):

        def run():

            import pyttsx3

            import speech\_recognition as sr

            import datetime

            import wikipedia

            import webbrowser

            import os

            engine = pyttsx3.init('sapi5')

            voices = engine.getProperty('voices')

            # print(voices[1].id)

            engine.setProperty('voice', voices[0].id)

            def speak(audio):

                engine.say(audio)

                engine.runAndWait()

            def wishMe():

                hour = int(datetime.datetime.now().hour)

                if hour>=0 and hour<12:

                    speak("Good Morning!")

                elif hour>=12 and hour<18:

                    speak("Good Afternoon!")

                else:

                    speak("Good Evening!")

                speak("Please tell me how may I help you")

            def takeCommand():

                r = sr.Recognizer()

                with sr.Microphone() as source:

                    print("Listening...")

                    r.pause\_threshold = 1

                    audio = r.listen(source)

                try:

                    print("Recognizing...")

                    speak("Recognizing...")

                    query = r.recognize\_google(audio, language='en-in')

                    print(f"User said: {query}\n")

                except Exception as e:

                    # print(e)

                    speak("Say that again please...")

                    return "None"

                return query

            '''def sendEmail(to, content):

                server = smtplib.SMTP('smtp.gmail.com', 587)

                server.ehlo()

                server.starttls()

                server.login('youremail@gmail.com', 'your-password')

                server.sendmail('youremail@gmail.com', to, content)

                server.close()'''

            def add(x, y):

                results=x+y

                print(results)

                speak(results)

            if \_\_name\_\_ == "\_\_main\_\_":

                wishMe()

                while True:

                # if 1:

                    query = takeCommand().lower()

                    # Logic for executing tasks based on query

                    if 'wikipedia' in query:

                        speak('Searching Wikipedia...')

                        query = query.replace("wikipedia", "")

                        results = wikipedia.summary(query, sentences=2)

                        speak("According to Wikipedia")

                        print(results)

                        speak(results)

                        #pyautogui.hotkey('ctrl', 'w')

                    elif 'open youtube' in query:

                        webbrowser.open("youtube.com")

                        speak('opening youtube')

                        #pyautogui.hotkey('ctrl', 'w')

                    elif 'open google' in query:

                        webbrowser.open("google.com")

                        speak('opening google')

                    elif 'find' in query:

                        takeCommand()

                    elif 'open stackoverflow' in query:

                        webbrowser.open("stackoverflow.com")

                    # elif 'play music' in query:

                    #     '''music\_dir = 'D:\\Non Critical\\songs\\Favorite Songs2'

                    #     songs = os.listdir(music\_dir)

                    #     print(songs)

                    #     os.startfile(os.path.join(music\_dir, songs[0]))'''

                    elif 'the time' in query:

                        strTime = datetime.datetime.now().strftime("%H:%M:%S")

                        speak(f"mam, the time is {strTime}")

        class Ui\_SubWindow(object):

            def setupUi(self, MainWindow):

                MainWindow.setObjectName("MainWindow")

                MainWindow.resize(313, 210)

                self.centralwidget = QtWidgets.QWidget(MainWindow)

                self.centralwidget.setObjectName("centralwidget")

                self.pushButton = QtWidgets.QPushButton(self.centralwidget,clicked = lambda: run())

                self.pushButton.setGeometry(QtCore.QRect(100, 20, 121, 121))

                self.pushButton.setStyleSheet("image:url(:/menubutton/mic-4.png)")

                self.pushButton.setText("")

                self.pushButton.setObjectName("pushButton")

                MainWindow.setCentralWidget(self.centralwidget)

                self.menubar = QtWidgets.QMenuBar(MainWindow)

                self.menubar.setGeometry(QtCore.QRect(0, 0, 313, 21))

                self.menubar.setObjectName("menubar")

                MainWindow.setMenuBar(self.menubar)

                self.statusbar = QtWidgets.QStatusBar(MainWindow)

                self.statusbar.setObjectName("statusbar")

                MainWindow.setStatusBar(self.statusbar)

                self.retranslateUi(MainWindow)

                QtCore.QMetaObject.connectSlotsByName(MainWindow)

            def retranslateUi(self, MainWindow):

                \_translate = QtCore.QCoreApplication.translate

                MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

        import image\_rc

        if \_\_name\_\_ == "\_\_main\_\_":

            import sys

            app = QtWidgets.QApplication(sys.argv)

            MainWindow = QtWidgets.QMainWindow()

            ui = Ui\_SubWindow()

            ui.setupUi(MainWindow)

            MainWindow.show()

            sys.exit(app.exec\_())

# create pyqt5 app

# start the app

    def openwindow(self):

        self.window = QtWidgets.QMainWindow()

        self.ui = Ui\_SubWindow()

        self.ui.setupUi(self.window)

        self.window.show()

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(1059, 585)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.frame\_2 = QtWidgets.QFrame(self.centralwidget)

        self.frame\_2.setGeometry(QtCore.QRect(210, 10, 841, 531))

        self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

        self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

        self.frame\_2.setObjectName("frame\_2")

        self.label = QtWidgets.QLabel(self.frame\_2)

        self.label.setGeometry(QtCore.QRect(0, 0, 831, 531))

        self.label.setText("")

        self.label.setObjectName("label")

        self.pushButton = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton.setGeometry(QtCore.QRect(20, 20, 171, 121))

        self.pushButton.setObjectName("pushButton")

        self.pushButton\_4 = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton\_4.connect(self.openwindow2)

        self.pushButton\_4.setGeometry(QtCore.QRect(20, 280, 171, 121))

        self.pushButton\_4.setObjectName("pushButton\_4")

        self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton\_3.clicked.connect(self.openwindow)

        self.pushButton\_3.setGeometry(QtCore.QRect(20, 150, 171, 121))

        self.pushButton\_3.setObjectName("pushButton\_3")

        self.pushButton\_5 = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton\_5.clicked.connect(self.openwindow1)

        self.pushButton\_5.setGeometry(QtCore.QRect(20, 410, 171, 121))

        self.pushButton\_5.setObjectName("pushButton\_5")

        self.pushButton\_5.clicked.connect(self.openwindow1)

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 1059, 21))

        self.menubar.setObjectName("menubar")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

        self.pushButton.setText(\_translate("MainWindow", "webengine"))

        self.pushButton\_4.setText(\_translate("MainWindow", "calculator"))

        self.pushButton\_3.setText(\_translate("MainWindow", "voicemode"))

        self.pushButton\_5.setText(\_translate("MainWindow", "games"))

if \_\_name\_\_ == "\_\_main\_\_":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    QApplication.setApplicationName("Gnet")

    MainWindow = QtWidgets.QMainWindow()

    ui = Ui\_MainWindow()

    ui.setupUi(MainWindow)

    MainWindow.show()

    sys.exit(app.exec\_())

calculator.py

# importing libraries

from PyQt5.QtWidgets import \*

from PyQt5 import QtCore, QtGui

from PyQt5.QtGui import \*

from PyQt5.QtCore import \*

import sys

class Window(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# setting title

self.setWindowTitle("calculator")

# setting geometry

self.setGeometry(100, 100, 360, 350)

# calling method

self.UiComponents()

# showing all the widgets

self.show()

# method for widgets

def UiComponents(self):

# creating a label

self.label = QLabel(self)

# setting geometry to the label

self.label.setGeometry(5, 5, 350, 70)

# creating label multi line

self.label.setWordWrap(True)

# setting style sheet to the label

self.label.setStyleSheet("QLabel"

"{"

"border : 4px solid black;"

"background : white;"

"}")

# setting alignment to the label

self.label.setAlignment(Qt.AlignRight)

# setting font

self.label.setFont(QFont('Arial', 15))

# adding number button to the screen

# creating a push button

push1 = QPushButton("1", self)

# setting geometry

push1.setGeometry(5, 150, 80, 40)

# creating a push button

push2 = QPushButton("2", self)

# setting geometry

push2.setGeometry(95, 150, 80, 40)

# creating a push button

push3 = QPushButton("3", self)

# setting geometry

push3.setGeometry(185, 150, 80, 40)

# creating a push button

push4 = QPushButton("4", self)

# setting geometry

push4.setGeometry(5, 200, 80, 40)

# creating a push button

push5 = QPushButton("5", self)

# setting geometry

push5.setGeometry(95, 200, 80, 40)

# creating a push button

push6 = QPushButton("5", self)

# setting geometry

push6.setGeometry(185, 200, 80, 40)

# creating a push button

push7 = QPushButton("7", self)

# setting geometry

push7.setGeometry(5, 250, 80, 40)

# creating a push button

push8 = QPushButton("8", self)

# setting geometry

push8.setGeometry(95, 250, 80, 40)

# creating a push button

push9 = QPushButton("9", self)

# setting geometry

push9.setGeometry(185, 250, 80, 40)

# creating a push button

push0 = QPushButton("0", self)

# setting geometry

push0.setGeometry(5, 300, 80, 40)

# adding operator push button

# creating push button

push\_equal = QPushButton("=", self)

# setting geometry

push\_equal.setGeometry(275, 300, 80, 40)

# adding equal button a color effect

c\_effect = QGraphicsColorizeEffect()

c\_effect.setColor(Qt.blue)

push\_equal.setGraphicsEffect(c\_effect)

# creating push button

push\_plus = QPushButton("+", self)

# setting geometry

push\_plus.setGeometry(275, 250, 80, 40)

# creating push button

push\_minus = QPushButton("-", self)

# setting geometry

push\_minus.setGeometry(275, 200, 80, 40)

# creating push button

push\_mul = QPushButton("\*", self)

# setting geometry

push\_mul.setGeometry(275, 150, 80, 40)

# creating push button

push\_div = QPushButton("/", self)

# setting geometry

push\_div.setGeometry(185, 300, 80, 40)

# creating push button

push\_point = QPushButton(".", self)

# setting geometry

push\_point.setGeometry(95, 300, 80, 40)

# clear button

push\_clear = QPushButton("Clear", self)

push\_clear.setGeometry(5, 100, 200, 40)

# del one character button

push\_del = QPushButton("Del", self)

push\_del.setGeometry(210, 100, 145, 40)

# adding action to each of the button

push\_minus.clicked.connect(self.action\_minus)

push\_equal.clicked.connect(self.action\_equal)

push0.clicked.connect(self.action0)

push1.clicked.connect(self.action1)

push2.clicked.connect(self.action2)

push3.clicked.connect(self.action3)

push4.clicked.connect(self.action4)

push5.clicked.connect(self.action5)

push6.clicked.connect(self.action6)

push7.clicked.connect(self.action7)

push8.clicked.connect(self.action8)

push9.clicked.connect(self.action9)

push\_div.clicked.connect(self.action\_div)

push\_mul.clicked.connect(self.action\_mul)

push\_plus.clicked.connect(self.action\_plus)

push\_point.clicked.connect(self.action\_point)

push\_clear.clicked.connect(self.action\_clear)

push\_del.clicked.connect(self.action\_del)

def action\_equal(self):

# get the label text

equation = self.label.text()

try:

# getting the ans

ans = eval(equation)

# setting text to the label

self.label.setText(str(ans))

except:

# setting text to the label

self.label.setText("Wrong Input")

def action\_plus(self):

# appending label text

text = self.label.text()

self.label.setText(text + " + ")

def action\_minus(self):

# appending label text

text = self.label.text()

self.label.setText(text + " - ")

def action\_div(self):

# appending label text

text = self.label.text()

self.label.setText(text + " / ")

def action\_mul(self):

# appending label text

text = self.label.text()

self.label.setText(text + " \* ")

def action\_point(self):

# appending label text

text = self.label.text()

self.label.setText(text + ".")

def action0(self):

# appending label text

text = self.label.text()

self.label.setText(text + "0")

def action1(self):

# appending label text

text = self.label.text()

self.label.setText(text + "1")

def action2(self):

# appending label text

text = self.label.text()

self.label.setText(text + "2")

def action3(self):

# appending label text

text = self.label.text()

self.label.setText(text + "3")

def action4(self):

# appending label text

text = self.label.text()

self.label.setText(text + "4")

def action5(self):

# appending label text

text = self.label.text()

self.label.setText(text + "5")

def action6(self):

# appending label text

text = self.label.text()

self.label.setText(text + "6")

def action7(self):

# appending label text

text = self.label.text()

self.label.setText(text + "7")

def action8(self):

# appending label text

text = self.label.text()

self.label.setText(text + "8")

def action9(self):

# appending label text

text = self.label.text()

self.label.setText(text + "9")

def action\_clear(self):

# clearing the label text

self.label.setText("")

def action\_del(self):

# clearing a single digit

text = self.label.text()

print(text[:len(text)-1])

self.label.setText(text[:len(text)-1])

# create pyqt5 app

App = QApplication(sys.argv)

# create the instance of our Window

window = Window()

# start the app

sys.exit(App.exec())