SWPSD – 2024/2025

Kalkulator głosowy w Pythonie z GUI (Tkinter)

Poniżej znajduje się implementacja kalkulatora głosowego w Pythonie z graficznym interfejsem użytkownika, wykorzystująca biblioteki speech_recognition i pyttsx3 zamiast Microsoft Speech Platform:

```
import speech_recognition as sr
import pyttsx3
import tkinter as tk
from tkinter import ttk
from threading import Thread
import queue
import re

class VoiceCalculatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Kalkulator głosowy - Python")
        self.root.geometry("500x400")

        # Inicjalizacja syntezatora mowy
        self.engine = pyttsx3.init()
        self.engine.setProperty('rate', 150)

        # Inicjalizacja rozpoznawania mowy
        self.recognizer = sr.Recognizer()
        self.microphone = sr.Microphone()
        self.speech_on = False
        self.command_queue = queue.Queue()

        # Utwórz interfejs użytkownika
        self.create_ui()

        # Uruchom wątek do przetwarzania komend
        self.process_thread = Thread(target=self.process_commands,
daemon=True)
        self.process_thread.start()

    def create_ui(self):
        """Tworzy interfejs użytkownika"""
        main_frame = ttk.Frame(self.root, padding="20")
        main_frame.pack(fill=tk.BOTH, expand=True)

        # Nagłówek
        ttk.Label(main_frame, text="Kalkulator głosowy", font=('Arial',
16)).pack(pady=10)

        # Przyciski kontrolne
        button_frame = ttk.Frame(main_frame)
        button_frame.pack(pady=10)
```

```
        self.start_button = ttk.Button(button_frame, text="Start",
command=self.start_listening)
        self.start_button.pack(side=tk.LEFT, padx=5)

        self.stop_button = ttk.Button(button_frame, text="Stop",
command=self.stop_listening, state=tk.DISABLED)
        self.stop_button.pack(side=tk.LEFT, padx=5)

        # Pole statusu
        self.status_label = ttk.Label(main_frame, text="Naciśnij Start,
aby rozpocząć nasłuchiwanie", wraplength=400)
        self.status_label.pack(pady=10)

        # Wynik
        self.result_label = ttk.Label(main_frame, text="",
font=('Arial', 14), foreground='blue')
        self.result_label.pack(pady=10)

        # Historia
        ttk.Label(main_frame, text="Historia:").pack()
        self.history_listbox = tk.Listbox(main_frame, height=5)
        self.history_listbox.pack(fill=tk.BOTH, expand=True, pady=5)

        # Przycisk czyszczenia
        ttk.Button(main_frame, text="Wyczyść historię",
command=self.clear_history).pack(pady=5)

    def start_listening(self):
        """Rozpoczyna nasłuchiwanie komend głosowych"""
        if not self.speech_on:
            self.speech_on = True
            self.start_button.config(state=tk.DISABLED)
            self.stop_button.config(state=tk.NORMAL)
            self.update_status("Nasłuchuję... Powiedz np. 'Oblicz 5
plus 3'")
            self.speak("Nasłuchuję")

            # Uruchom wątek do nasłuchiwania
            listen_thread = Thread(target=self.listen_loop,
daemon=True)
            listen_thread.start()

    def stop_listening(self):
        """Zatrzymuje nasłuchiwanie komend głosowych"""
        if self.speech_on:
            self.speech_on = False
            self.start_button.config(state=tk.NORMAL)
            self.stop_button.config(state=tk.DISABLED)
```

```
            self.update_status("Zatrzymano nasłuchiwanie")

    def listen_loop(self):
        """Pętla nasłuchująca komend głosowych"""
        with self.microphone as source:
            self.recognizer.adjust_for_ambient_noise(source)

            while self.speech_on:
                try:
                    audio = self.recognizer.listen(source, timeout=1,
phrase_time_limit=3)
                    command = self.recognizer.recognize_google(audio,
language="pl-PL")
                    self.command_queue.put(command.lower())
                except sr.WaitTimeoutError:
                    continue
                except sr.UnknownValueError:
                    self.command_queue.put("nie rozumiem")
                except Exception as e:
                    self.command_queue.put(f"błąd: {str(e)}")

    def process_commands(self):
        """Przetwarza komendy z kolejki"""
        while True:
            try:
                command = self.command_queue.get(timeout=0.1)

                # Aktualizacja interfejsu musi być w głównym wątku
                self.root.after(0, self.process_command, command)
            except queue.Empty:
                continue

    def process_command(self, command):
        """Przetwarza pojedynczą komendę"""
        self.update_status(f"Rozpoznano: {command}")

        if "stop" in command:
            self.stop_listening()
        elif "pomoc" in command:
            self.speak("Powiedz: Oblicz liczba plus minus razy lub
podzielić przez liczba")
        elif "nie rozumiem" in command or "błąd" in command:
            self.speak("Nie rozumiem, powtórz")
        elif "oblicz" in command:
            self.process_math_command(command)
        else:
            self.speak("Nie rozpoznano komendy")

    def process_math_command(self, command):
```

```
        """Przetwarza komendę matematyczną"""
        # Użyj wyrażenia regularnego do wyodrębnienia liczb i operacji
        match = re.search(r'oblicz (\d+) (plus|minus|razy|podzielić
przez) (\d+)', command)

        if match:
            num1 = int(match.group(1))
            operation = match.group(2)
            num2 = int(match.group(3))

            try:
                if operation == "plus":
                    result = num1 + num2
                elif operation == "minus":
                    result = num1 - num2
                elif operation == "razy":
                    result = num1 * num2
                elif operation == "podzielić przez":
                    result = num1 / num2

                output = f"{num1} {operation} {num2} = {result}"
                self.update_result(output)
                self.speak(f"Wynik to {result}")
                self.add_to_history(output)
            except ZeroDivisionError:
                self.update_result("Błąd: dzielenie przez zero")
                self.speak("Nie można dzielić przez zero")
            except Exception as e:
                self.update_result(f"Błąd: {str(e)}")
                self.speak("Wystąpił błąd podczas obliczeń")
        else:
            self.update_status("Nieprawidłowy format komendy
matematycznej")
            self.speak("Powtórz komendę w formacie: Oblicz liczba plus
minus razy lub podzielić przez liczba")

    def speak(self, text):
        """Syntezuje mowę"""
        self.engine.say(text)
        self.engine.runAndWait()

    def update_status(self, text):
        """Aktualizuje pole statusu"""
        self.status_label.config(text=text)

    def update_result(self, text):
        """Aktualizuje pole wyniku"""
        self.result_label.config(text=text)
```

```
    def add_to_history(self, item):
        """Dodaje wpis do historii"""
        self.history_listbox.insert(tk.END, item)
        self.history_listbox.see(tk.END)

    def clear_history(self):
        """Czyści historię"""
        self.history_listbox.delete(0, tk.END)

    def on_closing(self):
        """Zamyka aplikację"""
        self.speech_on = False
        self.root.destroy()

if __name__ == "__main__":
    root = tk.Tk()
    app = VoiceCalculatorApp(root)
    root.protocol("WM_DELETE_WINDOW", app.on_closing)
    root.mainloop()
```

Aby uruchomić ten program, potrzebujesz:

1. Pythona 3.6 lub nowszego

2. Następujących bibliotek (można zainstalować przez pip):

2. pip install speechrecognition pyttsx3 tkinter
3. Działa najlepiej na Windows, ale powinien też działać na Linux/Mac z odpowiednimi modyfikacjami

Uwagi dotyczące implementacji

1. **Różnice w stosunku do oryginalnego zadania**:
    1. Zamiast Microsoft Speech Platform użyto otwartych bibliotek Pythonowych
    2. GUI zbudowano przy użyciu Tkinter zamiast WPF
    3. Logika aplikacji jest podobna, ale implementacja różni się ze względu na różnice między językami
2. **Funkcjonalności**:
    1. Rozpoznawanie komend głosowych w języku polskim
    2. Synteza mowy z odpowiedziami
    3. Obsługa podstawowych operacji matematycznych
    4. Historia obliczeń
    5. Przyciski start/stop do kontroli nasłuchiwania