```
import os
import azure.cognitiveservices.speech as speechsdk

def translate_wav(
    wav_path: str,
    target_language: str,
    source_language: str | None = None,
    output_txt: str | None = None
):
    speech_key = os.getenv("AZURE_SPEECH_KEY")
    speech_region = os.getenv("AZURE_SPEECH_REGION")
    if not speech_key or not speech_region:
        raise RuntimeError("Please set AZURE_SPEECH_KEY and
AZURE_SPEECH_REGION.")

    if not os.path.isfile(wav_path):
        raise FileNotFoundError(f"WAV file not found: {wav_path}")

    translation_config = speechsdk.translation.SpeechTranslationConfig(
        subscription=speech_key, region=speech_region
    )

    if source_language:
        translation_config.speech_recognition_language = source_language
        auto_langs = None
    else:
        auto_langs = speechsdk.languageconfig.AutoDetectSourceLanguageConfig(
            languages=["en-US", "hi-IN", "mr-IN", "gu-IN"]
        )

    translation_config.add_target_language(target_language)

    audio_config = speechsdk.audio.AudioConfig(filename=wav_path)

    recognizer = (
        speechsdk.translation.TranslationRecognizer(
            translation_config=translation_config,
            audio_config=audio_config,
            auto_detect_source_language_config=auto_langs,
        )
        if not source_language
        else speechsdk.translation.TranslationRecognizer(
            translation_config=translation_config, audio_config=audio_config
        )
    )

    translated_lines = []
    done = False

    def recognized(evt):
        result = evt.result
        if result.reason == speechsdk.ResultReason.TranslatedSpeech:
            tgt = result.translations.get(target_language, "")
            if tgt:
                print(f"[{target_language.upper()}] {tgt}")
                translated_lines.append(tgt)

    def stop(evt):
        nonlocal done
        done = True

    recognizer.recognized.connect(recognized)
    recognizer.session_stopped.connect(stop)
    recognizer.canceled.connect(stop)

    print("Starting translation...")
    recognizer.start_continuous_recognition()
    while not done:
        pass
    recognizer.stop_continuous_recognition()

    if output_txt:
        with open(output_txt, "w", encoding="utf-8") as f:
            f.write("\n".join(translated_lines))
        print(f"Saved translated text → {output_txt}")

def main():
    import argparse

    p = argparse.ArgumentParser()
    p.add_argument("wav", help="Path to WAV file")
    p.add_argument("--to", required=True, help="Target language code, e.g., hi or en")
    p.add_argument("--from-lang", default=None, help="Source language code, e.g., en-US")
    p.add_argument("--out", default=None, help="Output text file")
    a = p.parse_args()

    translate_wav(a.wav, a.to, a.from_lang, a.out)
```

```
if __name__ == "__main__":
    main()
```