Introduction to PyDub

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke
Machine Learning Engineer/YouTube Creator

Installing PyDub

```
$ pip install pydub
```

- If using files other than `.wav`, install `ffmpeg` via ffmpeg.org

PyDub's main class, AudioSegment

```
# Import PyDub main class
from pydub import AudioSegment
```

```
# Import an audio file
wav_file = AudioSegment.from_file(file="wav_file.wav", format="wav")
```

```
# Format parameter only for readability
wav_file = AudioSegment.from_file(file="wav_file.wav")
```

```
type(wav_file)
```

```
pydub.audio_segment.AudioSegment
```

Playing an audio file

```
# Install simpleaudio for wav playback
$pip install simpleaudio
```

```
# Import play function
from pydub.playback import play
```

```
# Import audio file
wav_file = AudioSegment.from_file(file="wav_file.wav")


# Play audio file
play(wav_file)
```

Audio parameters

```
# Import audio files
wav_file = AudioSegment.from_file(file="wav_file.wav")
two_speakers = AudioSegment.from_file(file="two_speakers.wav")

# Check number of channels
wav_file.channels, two_speakers.channels
```

```
1, 2
```

```
wav_file.frame_rate
```

```
480000
```

Audio parameters

```
# Find the number of bytes per sample
wav_file.sample_width
```

```
2
```

```
# Find the max amplitude
wav_file.max
```

```
8488
```

Audio parameters

```
# Duration of audio file in milliseconds
len(wav_file)
```

```
3284
```

Changing audio parameters

```
# Change ATTRIBUTENAME of AudioSegment to x
changeed_audio_segment = audio_segment.set_ATTRIBUTENAME(x)
```

```
# Change sample width to 1
wav_file_width_1 = wav_file.sample_width(1)
wav_file_width_1.sample_width
```

```
1
```

Changing audio parameters

```
# Change sample rate
wav_file_16k = wav_file.frame_rate(16000)
wav_file_16k.frame_rate
```

```
16000
```

```
# Change number of channels
wav_file_1_channel = wav_file.set_channels(1)
wav_file_1_channel.channels
```

```
1
```

Let's practice!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Manipulating audio files with PyDub

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke
Machine Learning Engineer/YouTube Creator

Turning it down to 11

```
# Import audio file
wav_file = AudioSegment.from_file("wav_file.wav")
# Minus 60 dB
quiet_wav_file = wav_file - 60
```

```
# Try to recognize quiet audio
recognizer.recognize_google(quiet_wav_file)
```

```
UnknownValueError:
```

Increasing the volume

```
# Increase the volume by 10 dB
louder_wav_file = wav_file + 10
```

```
# Try to recognize
recognizer.recognize_google(louder_wav_file)
```

```
this is a wav file
```

This all sounds the same

```
# Import AudioSegment and normalize
from pydub import AudioSegment
from pydub.effects import normalize
from pydub.playback import play
```

```
# Import uneven sound audio file
loud_quiet = AudioSegment.from_file("loud_quiet.wav")
# Normalize the sound levels
normalized_loud_quiet = normalize(loud_quiet)
```

```
# Check the sound
play(normalized_loud_quiet)
```

Remixing your audio files

```
# Import audio with static at start
static_at_start = AudioSegment.from_file("static_at_start.wav")
```

```
# Remove the static via slicing
no_static_at_start = static_at_start[5000:]


# Check the new sound
play(no_static_at_start)
```

Remixing your audio files

```
# Import two audio files
wav_file_1 = AudioSegment.from_file("wav_file_1.wav")
wav_file_2 = AudioSegment.from_file("wav_file_2.wav")
```

```
# Combine the two audio files
wav_file_3 = wav_file_1 + wav_file_2

# Check the sound
play(wav_file_3)
```

```
# Combine two wav files and make the combination louder
louder_wav_file_3 = wav_file_1 + wav_file_2 + 10
```

Splitting your audio

```
# Import phone call audio
phone_call = AudioSegment.from_file("phone_call.wav")
# Find number of channels
phone_call.channels
```

```
2
```

```
# Split stereo to mono
phone_call_channels = phone_call.split_to_mono()
phone_call_channels
```

```
[<pydub.audio_segment.AudioSegment, <pydub.audio_segment.AudioSegment>]
```

Splitting your audio

```
# Find number of channels of first list item
phone_call_channels[0].channels
```

```
1
```

```
# Recognize the first channel
recognizer.recognize_google(phone_call_channel_1)
```

```
the pydub library is really useful
```

Let's code!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Converting and saving audio files with PyDub

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke
Machine Learning Engineer/YouTube Creator

Exporting audio files

```
from pydub import AudioSegment

# Import audio file
wav_file = AudioSegment.from_file("wav_file.wav")
# Increase by 10 decibels
louder_wav_file = wav_file + 10
# Export louder audio file
louder_wav_file.export(out_f="louder_wav_file.wav", format="wav")
```

```
<_io.BufferedRandom name='louder_wav_file.wav'>
```

Reformatting and exporting multiple audio files

```
def make_wav(wrong_folder_path, right_folder_path):

    # Loop through wrongly formatted files
    for file in os.scandir(wrong_folder_path):

    # Only work with files with audio extensions we're fixing
    if file.path.endswith(".mp3") or file.path.endswith(".flac"):

        # Create the new .wav filename
        out_file = right_folder_path + os.path.splitext(os.path.basename(file.path))[0] + ".wav"

    # Read in the audio file and export it in wav format
    AudioSegment.from_file(file.path).export(out_file,
                                             format="wav")

    print(f"Creating {out_file}")
```

Reformatting and exporting multiple audio files

```
# Call our new function
make_wav("data/wrong_formats/", "data/right_format/")
```

```
Creating data/right_types/wav_file.wav
Creating data/right_types/flac_file.wav
Creating data/right_types/mp3_file.wav
```

Manipulating and exporting

```
def make_no_static_louder(static_quiet, louder_no_static):
    # Loop through files with static and quiet (already in wav format)
    for file in os.scandir(static_quiet_folder_path):

        # Create new file path
        out_file = louder_no_static + os.path.splitext(os.path.basename(file.path))[0] + ".wav"

        # Read the audio file
        audio_file = AudioSegment.from_file(file.path)

        # Remove first three seconds and add 10 decibels and export
        audio_file = (audio_file[3100:] + 10).export(out_file, format="wav")


        print(f"Creating {out_file}")
```

Manipulating and exporting

```
# Remove static and make louder
make_no_static_louder("data/static_quiet/", "data/louder_no_static/")
```

```
Creating data/louder_no_static/speech-recognition-services.wav
Creating data/louder_no_static/order-issue.wav
Creating data/louder_no_static/help-with-acount.wav
```

Your turn!

SPOKEN LANGUAGE PROCESSING IN PYTHON