Plik frames.py służy do obróbki ramek, tzn. kodowania i odkowania wiadomości przychodzących/wychodzących
Zawiera on funkcje służące do zamiany stringa na bity i odwrotnie (str_to_bits i bits_to_str), a każde funkcje
realizujące kodowanie 4b5b i odkodowanie 5b4b (convert4B5B, convert5B4B).
Główną częścią frames.py są funkcje encipher i decipher, które odpowiednio kodują nam wiadomość message od x do y,
czyli "x y message" na ciąg bitów i ten ciąg bitów później odkodowują do początkowej komunikacji, przekazując nadawcę, odbiorcę i wiadomość,
czyli krotkę (x, y, message).
Jeśli suma kontrolna się nie zgadza, bądź nasz ciąg bajtów nie spełnia któregoś z podstawowych warunków na bycie poprawnie zakodowaną wiadomością, to zwracamy pustą krotkę.

Plik play.py służy do nadawania dźwięku, w szczególności może zostać wywołany z argumentem będącym nazwą pliku, do którego chcemy ten dźwięk przekazać (./play.py filename.wav).
Sam program w głównej pętli czeka na standardowym wejściu na kolejne linie, i zapomocą frames.py koduje daną linię i w funkcji playTheBox(message) przesyła wiadomość jako kombinacje dźwięków o częstotliwościach freq0 (częstotliwość bitu 0) i freq1 (częstotliwość bitu 1).
Jeśli przesyła to do pliku to dodatkowo tworzymy na początku ciszę o losowej długości, a także losowe przesunięcie w ramach testowania.
Plik posiada zmienne globalne framerate, amplitude, bitsPerSec, freq0, freq1, których wartości można w ramach rozsądku zmieniać.

Plik receive.py polega na ciągłym próbkowania dźwięku ze źródła, które znów może być plikiem .wav podanym jako argument (./receive.py filename.wav).
Sam program wczytuje kolejne próbki, których rozmiar ma odpowiadawać jednemu bitowi, a następnie ocenia czy częstotliwość próbki mieści, że w marginesie błędu
(do tego służy funkcji checkFreq(sample)). Jeżeli tak to próbujemy się synchronizować za pomocą funkcji synchronize(), która ocenia dla jakiego przesunięcia jest najbardziej
prawdopodobne, że się wpasowaliśmy.
Następnie, jeśli się wpasowaliśmy i po drodze nie było szumu, to czytamy dźwięk dopóki się nie skończy preambuła czyli nie najedziemy na jedynki obok siebie (funkcja read_preambule())
Na końcu za pomocą funkcji read_message() czytamy całą wiadomość i próbujemy ją odkodować i wypisać krotkę (nadawca, odbiorca, wiadomość).
Plik posiada zmienne globalne framerate, sync_precision, bitsPerSec, freq0, freq1, których wartości można w ramach rozsądku zmieniać.

Część testów była wykonywana w następujący sposób:
./play.py test.wav
./receive.py test.wav
Oczywiście w play.py dawaliśmy losowe przesunięcie i losową ciszę na początku, a potem porównanie outputu z inputem
Pozostała część testów było przeprowadzana na mikrofonie z głośnikiem.