```
"""
An interactive command line program that plays sound files in the music folder.
"""
import pygame
import os

DEBUG = False


class MusicPlayer():
    """One object of this class represents a running music player
    command line program."""

    def __init__(self):
        """Constructor for a MusicPlayer object.
        Sets up the Pygame Mixer so it can play files in the ./music folder
        and then runs the main interactive loop."""
        # Initiating Pygame
        pygame.init()
        # Initiating Pygame Mixer
        pygame.mixer.init()
        # Declaring status member variable
        self.status = ""
        # Declaring playlist member variable
        self.playlist = []

        # Changing directory for fetching songs
        os.chdir("./music")

        # Refeshing the playlist
        self.refreshlist()

        # Running the main interaction loop
        self.main()

    def refreshlist(self):
        """Refreshes the playlist of files, ignoring files whose names start with
        a period because those are typically hidden operating system files or
        file system meta-data files."""
        self.playlist = []

        temp = os.listdir()
        for i in temp:
            if i[0] == ".":
                x=1
            else:
                self.playlist.append(i)

        # TODO: Fetch song filenames for the self.playlist
        # using the os.listdir method

        # TODO: Remove filenames that start with "." from the self.playlist

    def playsong(self, playlistIndex):
        """Plays the file in the playlist whose index is playlistIndex.
        playlistIndex: int"""
        playlist = self.playlist
        self.status = "Playing {}".format(playlist[playlistIndex])
        # TODO: Update self.status to be "Playing filename" by
```

```
    # getting the filename from self.playlist using playlistIndex

    # TODO: Load the selected song by

    # getting the filename from self.playlist using playlistIndex
    pygame.mixer.music.load(playlist[playlistIndex])
    # Playing selected song
    pygame.mixer.music.play()
    # Displaying status
    print(self.status)

def stopsong(self):
    """Stops any currently playing song."""
    # Updating status
    self.status = "-Stopped"
    # Stopping the current song
    pygame.mixer.music.stop()
    # Displaying status
    print(self.status)

def printlist(self):
    """TODO: Print the playlist along with its zero-index."""
    playlist = self.playlist
    n = len(playlist)
    # print(temp)
    for n in range(len(playlist)):
        print(playlist.index(playlist[n]), playlist[n])

def main(self):
    """TODO: Prompt user and play songs based on typed input."""
    print("Welcome to Music Player!")
    n = 1
    while (n == 1):
        self.refreshlist()
        print(" ")
        self.printlist()
        print("Type the song number to play it,")
        print("or r to refresh the list,")
        print("or q to quit:")
        a = input()
        if (a == "q") or (a == "Q"):
            n==0

        elif (a == "r") or (a == "R"):
            n==1
        elif not a.isnumeric():
            n==1
        else:
            b = int(a)
            if (b < 0) or (b > (len(self.playlist))):
                n==1
            else:
                self.playsong(b)

    # TODO: You are required to use a while loop, input, self.printlist,
    # self.refreshlist, and self.playsong.
    # See: https://docs.python.org/3/library/functions.html#input
    # TODO: You are required to prevent any user input errors
    # (see the README.md file's TODO comments). To do that
```

```
        # you are expected to use the built-in len function,
        # and the str methods strip, casefold, and isnumeric.
        # Do NOT catch exceptions.
        self.stopsong()


def main():
    """Create a MusicPlayer object which will immedietly run its own
    command line interaction loop."""
    # It is fine that player is created but never used any further
    player = MusicPlayer()


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