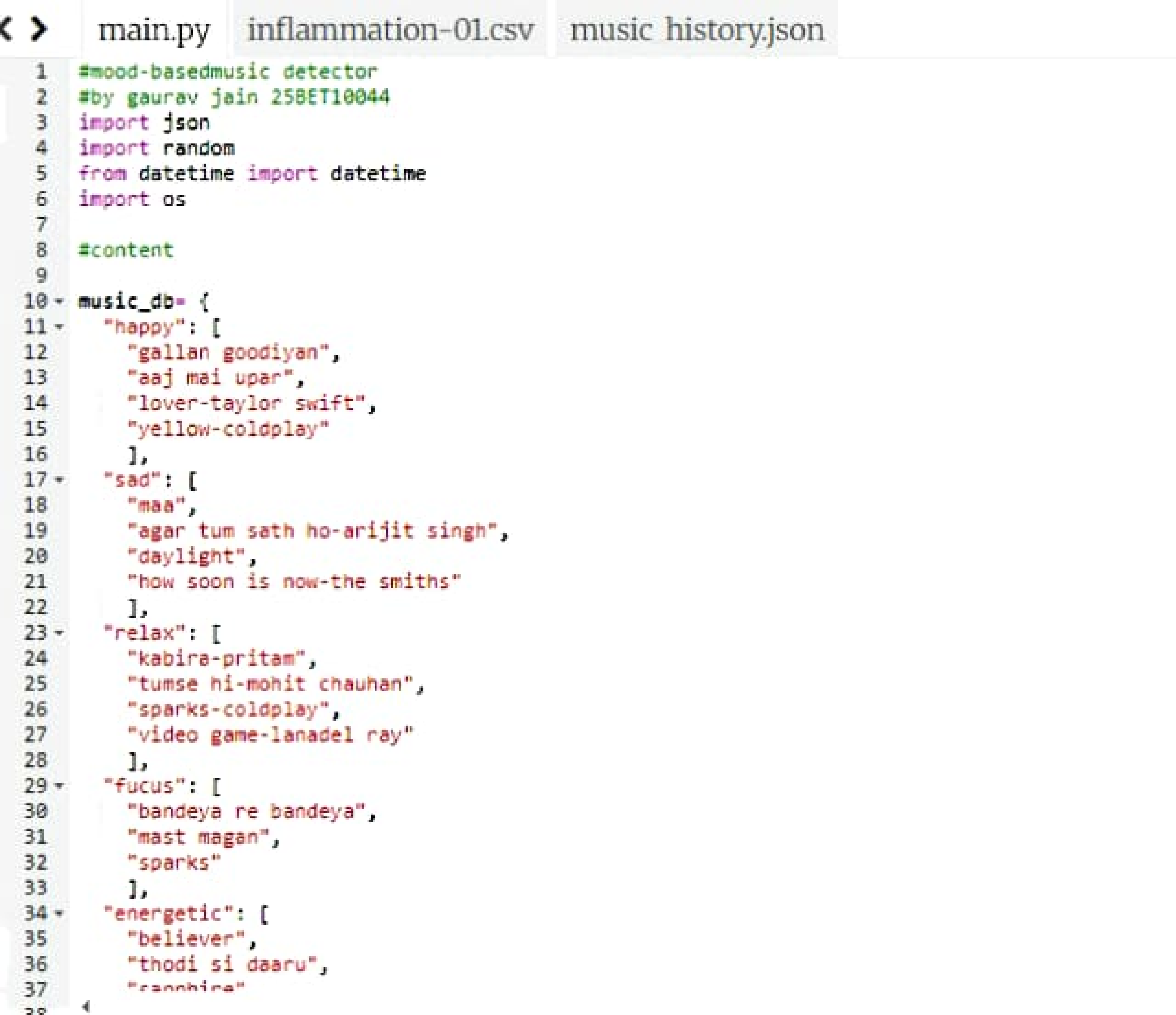
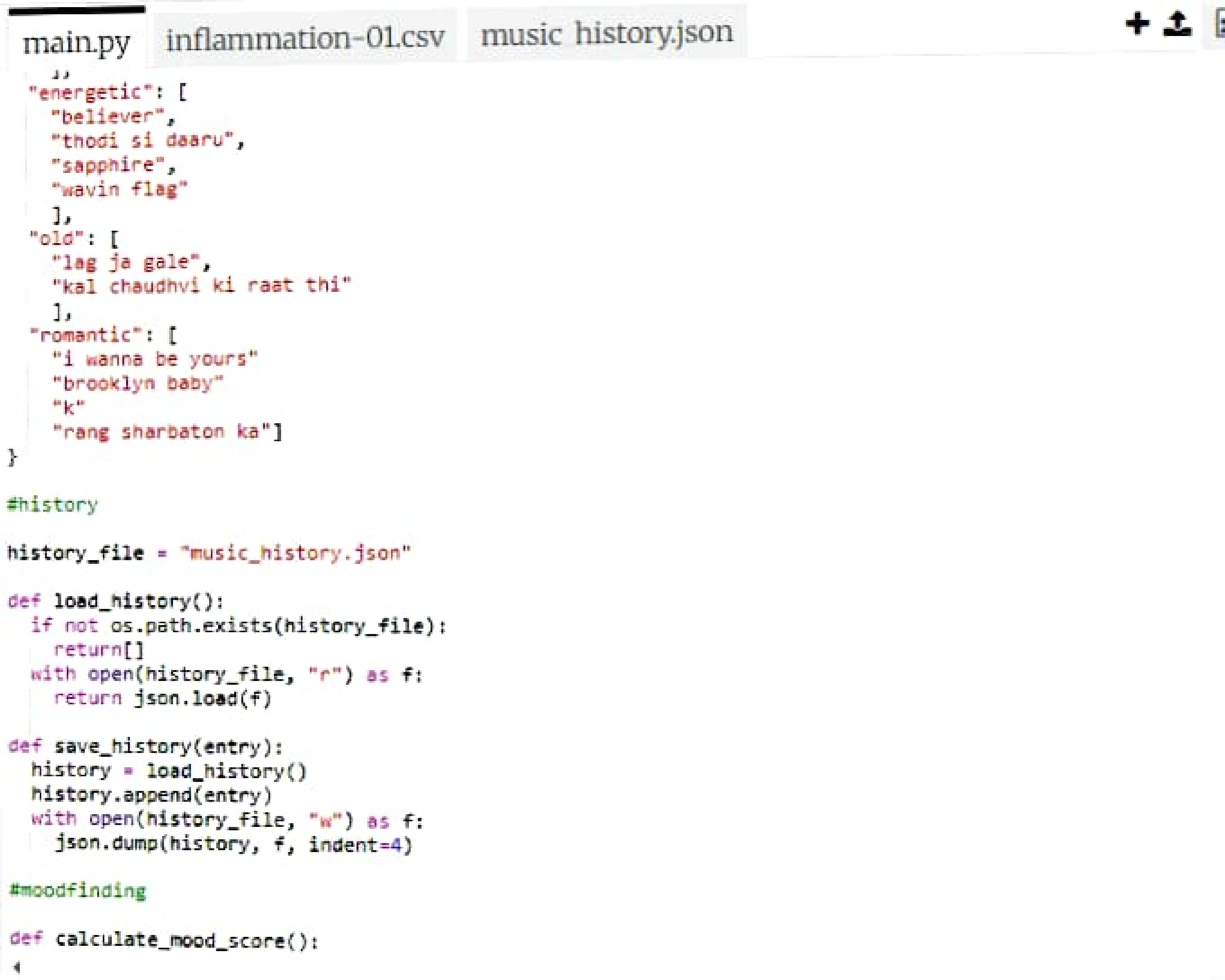
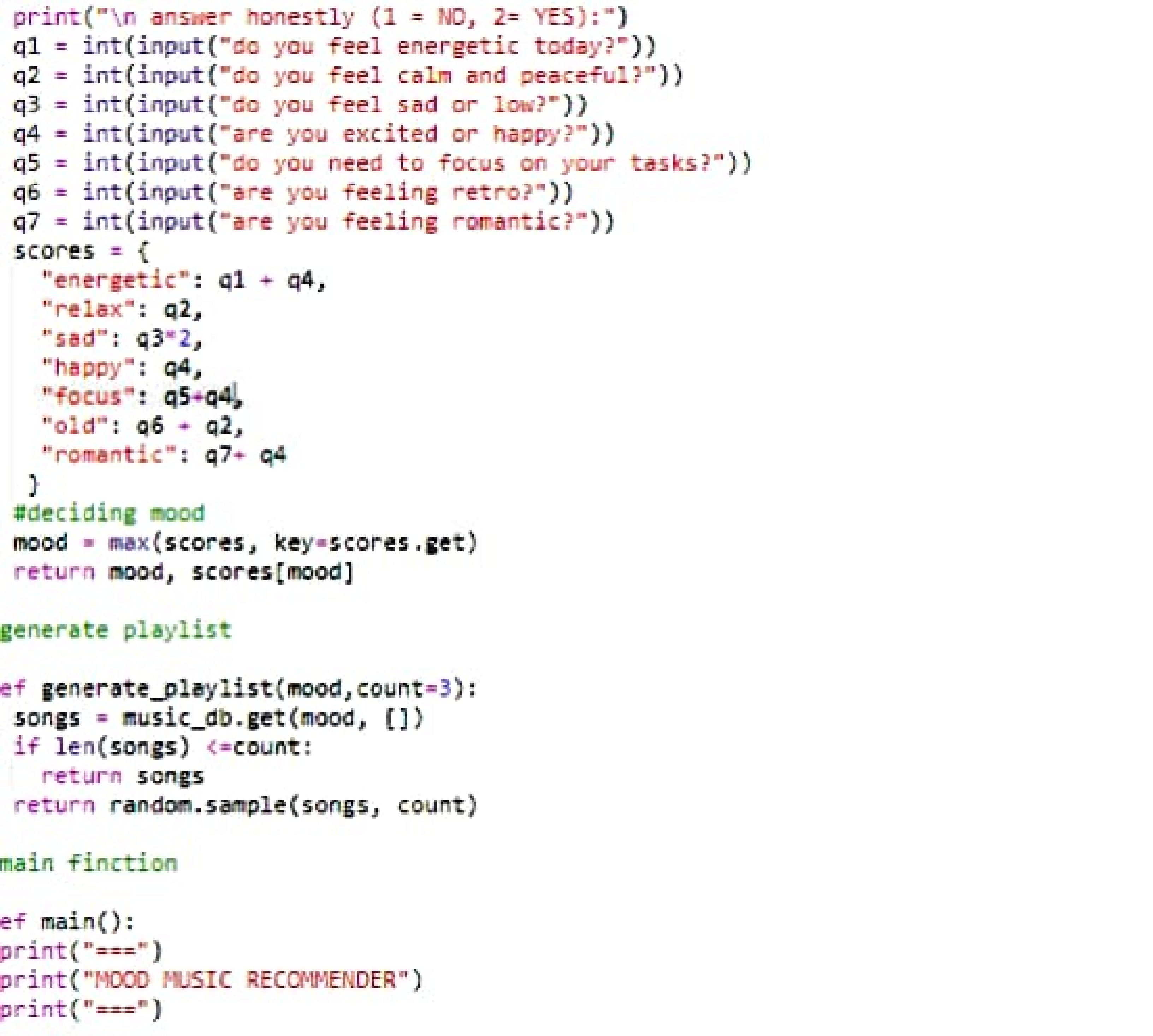


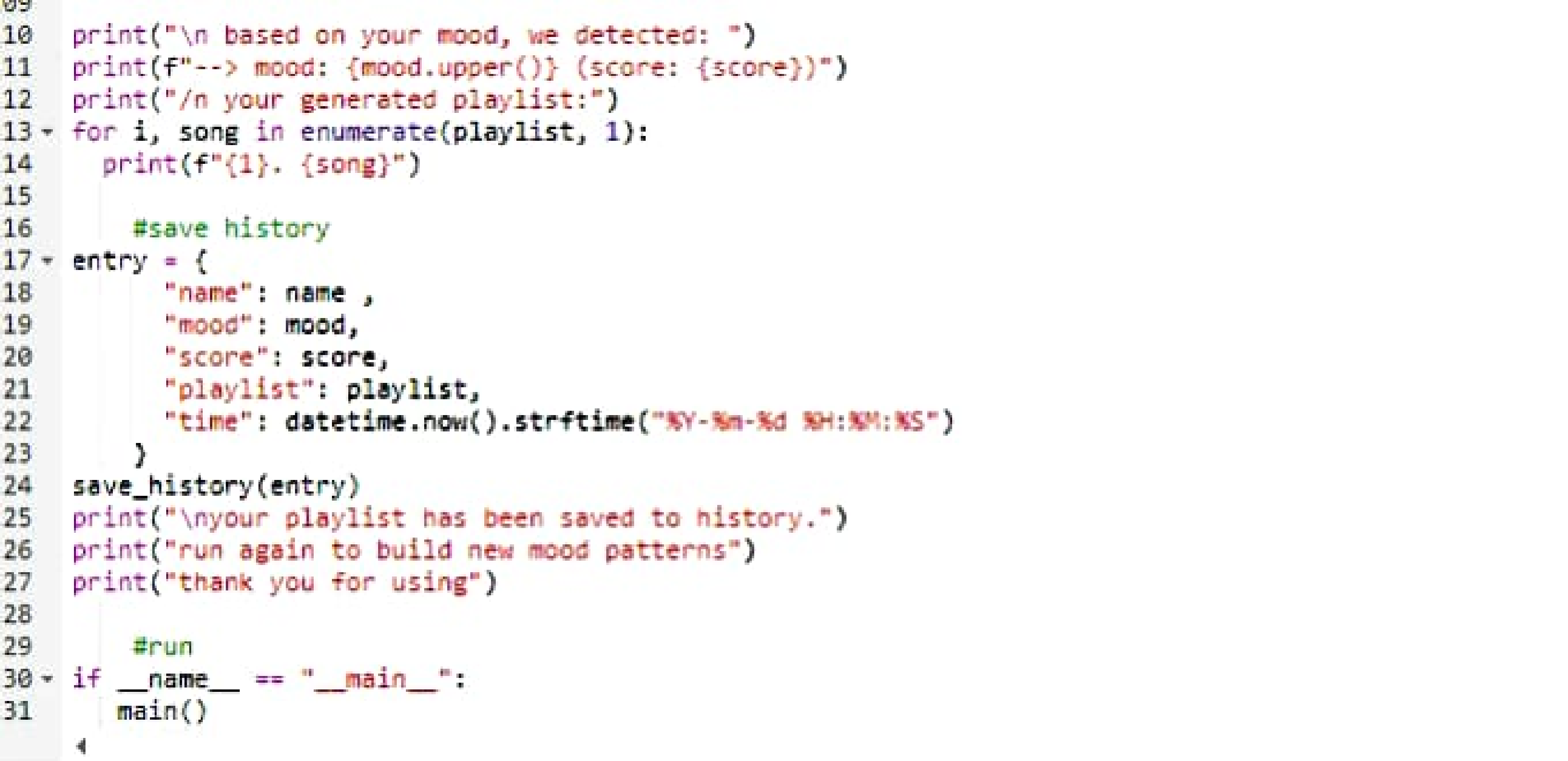
```
#mood-basedmusic detector
#by gaurav jain 25BET10044
import json
import random
from datetime import datetime
import os

#content

music_db= {
  "happy": [
    "gallan goodiyan",
    "aaj mai upar",
    "lover-taylor swift",
    "yellow-coldplay"
    ],
  "sad": [
    "maa",
    "agar tum sath ho-arijit singh",
    "daylight",
    "how soon is now-the smiths"
    ],
  "relax": [
    "kabira-pritam",
    "tumse hi-mohit chauhan",
    "sparks-coldplay",
    "video game-lanadel ray"
    ],
  "fucus": [
    "bandeya re bandeya",
    "mast magan",
    "sparks"
    ],
  "energetic": [
    "believer",
    "thodi si daaru",
    "sapphire"
```



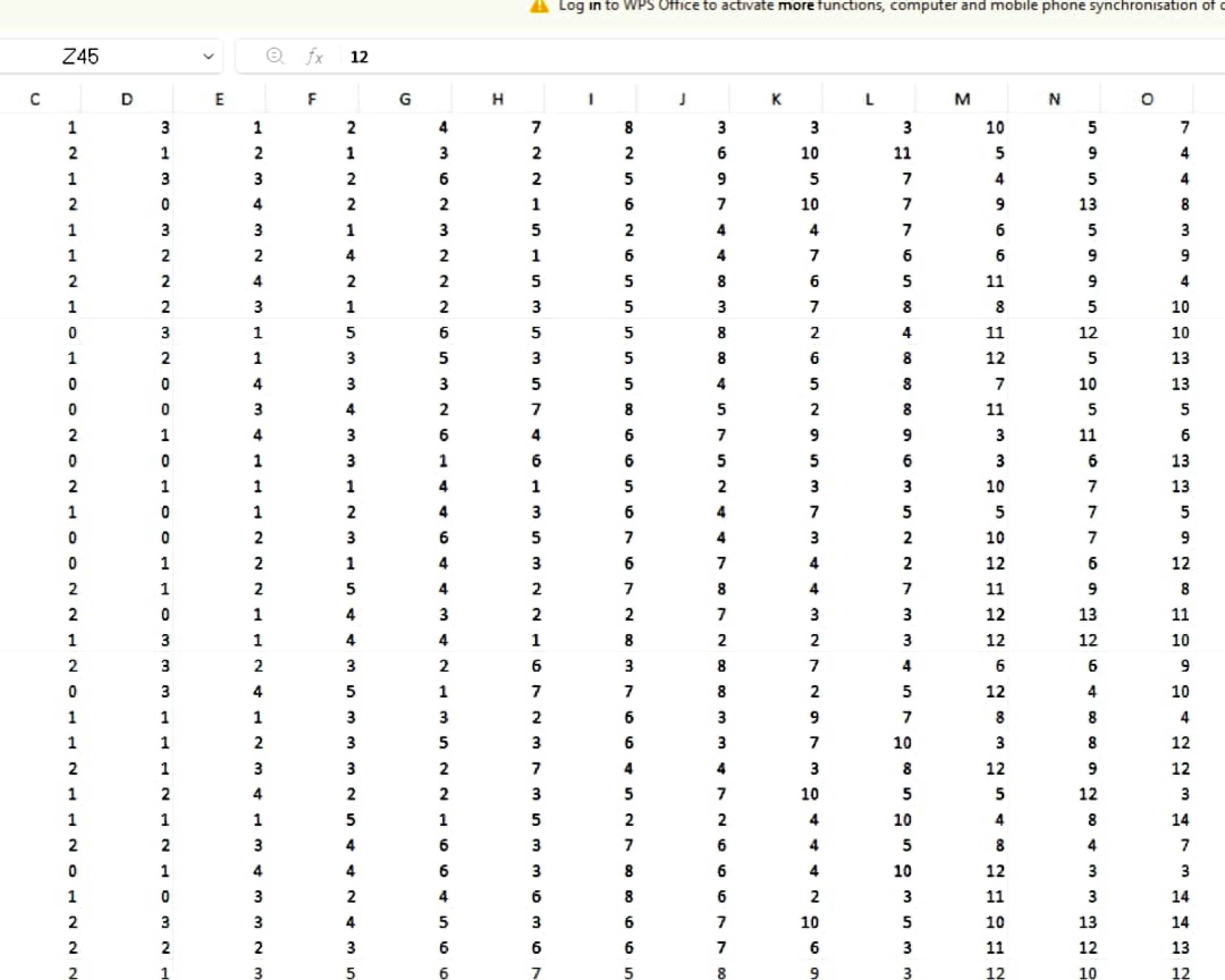
```
    ],
  "energetic": [
    "believer",
    "thodi si daaru",
    "sapphire",
    "wavin flag"
    ],
  "old": [
    "lag ja gale",
    "kal chaudhvi ki raat thi"
    ],
  "romantic": [
    "i wanna be yours"
    "brooklyn baby"
    "k"
    "rang sharbaton ka"]
}

#history

history_file = "music_history.json"

def load_history():
  if not os.path.exists(history_file):
    return[]
  with open(history_file, "r") as f:
    return json.load(f)

def save_history(entry):
  history = load_history()
  history.append(entry)
  with open(history_file, "w") as f:
    json.dump(history, f, indent=4)

#moodfinding

def calculate_mood_score():
```



```
 print("\n answer honestly (1 = NO, 2= YES):")
 q1 = int(input("do you feel energetic today?"))
 q2 = int(input("do you feel calm and peaceful?"))
 q3 = int(input("do you feel sad or low?"))
 q4 = int(input("are you excited or happy?"))
 q5 = int(input("do you need to focus on your tasks?"))
 q6 = int(input("are you feeling retro?"))
 q7 = int(input("are you feeling romantic?"))
 scores = {
   "energetic": q1 + q4,
   "relax": q2,
   "sad": q3*2,
   "happy": q4,
   "focus": q5+q4,
   "old": q6 + q2,
   "romantic": q7+ q4
  }
 #deciding mood
 mood = max(scores, key=scores.get)
 return mood, scores[mood]

generate playlist

ef generate_playlist(mood,count=3):
 songs = music_db.get(mood, [])
 if len(songs) <=count:
   return songs
 return random.sample(songs, count)

main finction

ef main():
print("===")
print("MOOD MUSIC RECOMMENDER")
print("===")
```



```
print("\n based on your mood, we detected: ")
print(f"--> mood: {mood.upper()} (score: {score})")
print("/n your generated playlist:")
for i, song in enumerate(playlist, 1):
  print(f"{1}. {song}")

    #save history
entry = {
      "name": name ,
      "mood": mood,
      "score": score,
      "playlist": playlist,
      "time": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    }
save_history(entry)
print("\nyour playlist has been saved to history.")
print("run again to build new mood patterns")
print("thank you for using")

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



⚠ Log in to WPS Office to activate more functions, computer and mobile phone synchronisation of c
Z45
fx
12

| C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 2 | 4 | 7 | 8 | 3 | 3 | 3 | 10 | 5 | 7 |
| 2 | 1 | 2 | 1 | 3 | 2 | 2 | 6 | 10 | 11 | 5 | 9 | 4 |
| 1 | 3 | 3 | 2 | 6 | 2 | 5 | 9 | 5 | 7 | 4 | 5 | 4 |
| 2 | 0 | 4 | 2 | 2 | 1 | 6 | 7 | 10 | 7 | 9 | 13 | 8 |
| 1 | 3 | 3 | 1 | 3 | 5 | 2 | 4 | 4 | 7 | 6 | 5 | 3 |
| 1 | 2 | 2 | 4 | 2 | 1 | 6 | 4 | 7 | 6 | 6 | 9 | 9 |
| 2 | 2 | 4 | 2 | 2 | 5 | 5 | 8 | 6 | 5 | 11 | 9 | 4 |
| 1 | 2 | 3 | 1 | 2 | 3 | 5 | 3 | 7 | 8 | 8 | 5 | 10 |
| 0 | 3 | 1 | 5 | 6 | 5 | 5 | 8 | 2 | 4 | 11 | 12 | 10 |
| 1 | 2 | 1 | 3 | 5 | 3 | 5 | 8 | 6 | 8 | 12 | 5 | 13 |
| 0 | 0 | 4 | 3 | 3 | 5 | 5 | 4 | 5 | 8 | 7 | 10 | 13 |
| 0 | 0 | 3 | 4 | 2 | 7 | 8 | 5 | 2 | 8 | 11 | 5 | 5 |
| 2 | 1 | 4 | 3 | 6 | 4 | 6 | 7 | 9 | 9 | 3 | 11 | 6 |
| 0 | 0 | 1 | 3 | 1 | 6 | 6 | 5 | 5 | 6 | 3 | 6 | 13 |
| 2 | 1 | 1 | 1 | 4 | 1 | 5 | 2 | 3 | 3 | 10 | 7 | 13 |
| 1 | 0 | 1 | 2 | 4 | 3 | 6 | 4 | 7 | 5 | 5 | 7 | 5 |
| 0 | 0 | 2 | 3 | 6 | 5 | 7 | 4 | 3 | 2 | 10 | 7 | 9 |
| 0 | 1 | 2 | 1 | 4 | 3 | 6 | 7 | 4 | 2 | 12 | 6 | 12 |
| 2 | 1 | 2 | 5 | 4 | 2 | 7 | 8 | 4 | 7 | 11 | 9 | 8 |
| 2 | 0 | 1 | 4 | 3 | 2 | 2 | 7 | 3 | 3 | 12 | 13 | 11 |
| 1 | 3 | 1 | 4 | 4 | 1 | 8 | 2 | 2 | 3 | 12 | 12 | 10 |
| 2 | 3 | 2 | 3 | 2 | 6 | 3 | 8 | 7 | 4 | 6 | 6 | 9 |
| 0 | 3 | 4 | 5 | 1 | 7 | 7 | 8 | 2 | 5 | 12 | 4 | 10 |
| 1 | 1 | 1 | 3 | 3 | 2 | 6 | 3 | 9 | 7 | 8 | 8 | 4 |
| 1 | 1 | 2 | 3 | 5 | 3 | 6 | 3 | 7 | 10 | 3 | 8 | 12 |
| 2 | 1 | 3 | 3 | 2 | 7 | 4 | 4 | 3 | 8 | 12 | 9 | 12 |
| 1 | 2 | 4 | 2 | 2 | 3 | 5 | 7 | 10 | 5 | 5 | 12 | 3 |
| 1 | 1 | 1 | 5 | 1 | 5 | 2 | 2 | 4 | 10 | 4 | 8 | 14 |
| 2 | 2 | 3 | 4 | 6 | 3 | 7 | 6 | 4 | 5 | 8 | 4 | 7 |
| 0 | 1 | 4 | 4 | 6 | 3 | 8 | 6 | 4 | 10 | 12 | 3 | 3 |
| 1 | 0 | 3 | 2 | 4 | 6 | 8 | 6 | 2 | 3 | 11 | 3 | 14 |
| 2 | 3 | 3 | 4 | 5 | 3 | 6 | 7 | 10 | 5 | 10 | 13 | 14 |
| 2 | 2 | 2 | 3 | 6 | 6 | 6 | 7 | 6 | 3 | 11 | 12 | 13 |
| 2 | 1 | 3 | 5 | 6 | 7 | 5 | 8 | 9 | 3 | 12 | 10 | 12 |