```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


import csv

sample_scores = {
    "ENGLISH": 0.9839,
    "GEOGRAPHY": 0.9331,
    "CIVICS": 0.8113,
    "HISTORY": 0.8758,
    "BIOLOGY": 0.5690,
    "PHYSICS": 0.8623,
    "CHEMISTRY": 0.5938,
    "MATHEMATICS": 0.5181
}

thinkingOrientation = {
    "analyticalIndex": 62.88,
    "technicalIndex": 23.43,
    "creativeIndex": 4.69,
    "peopleOrientation": {
        "CAUTIOUS": 0,
        "INSPIRING": 3,
        "DOMINANT": 2,
        "SUPPORTIVE": 3
    },
    "thinkingOrientationForGrads": "ANALYTICAL",
    "peopleOrientationIndex": 29.18
}

cognitive_indices = {
    "ENTJ": {"ENGLISH": 0.9, "GEOGRAPHY": 0.8, "CIVICS": 0.7, "HISTORY": 0.8, "BIOLOGY": 0.7, "PHYSICS": 0.9, "CHEMISTRY": 0.8, "MATHEMATICS": 0.9},
    "INTJ": {"ENGLISH": 0.8, "GEOGRAPHY": 0.9, "CIVICS": 0.8, "HISTORY": 0.9, "BIOLOGY": 0.7, "PHYSICS": 0.8, "CHEMISTRY": 0.9, "MATHEMATICS": 0.9},
    "ENTP": {"ENGLISH": 0.9, "GEOGRAPHY": 0.9, "CIVICS": 0.6, "HISTORY": 0.7, "BIOLOGY": 0.8, "PHYSICS": 0.7, "CHEMISTRY": 0.7, "MATHEMATICS": 0.8},
    "INTP": {"ENGLISH": 0.8, "GEOGRAPHY": 0.8, "CIVICS": 0.9, "HISTORY": 0.9, "BIOLOGY": 0.7, "PHYSICS": 0.8, "CHEMISTRY": 0.9, "MATHEMATICS": 0.7},
    "ENFJ": {"ENGLISH": 0.7, "GEOGRAPHY": 0.6, "CIVICS": 0.8, "HISTORY": 0.6, "BIOLOGY": 0.7, "PHYSICS": 0.6, "CHEMISTRY": 0.6, "MATHEMATICS": 0.6},
    "INFJ": {"ENGLISH": 0.7, "GEOGRAPHY": 0.7, "CIVICS": 0.7, "HISTORY": 0.7, "BIOLOGY": 0.7, "PHYSICS": 0.7, "CHEMISTRY": 0.7, "MATHEMATICS": 0.7},
    "ENFP": {"ENGLISH": 0.6, "GEOGRAPHY": 0.6, "CIVICS": 0.6, "HISTORY": 0.6, "BIOLOGY": 0.6, "PHYSICS": 0.6, "CHEMISTRY": 0.6, "MATHEMATICS": 0.6},
    "INFP": {"ENGLISH": 0.9, "GEOGRAPHY": 0.9, "CIVICS": 0.9, "HISTORY": 0.9, "BIOLOGY": 0.9, "PHYSICS": 0.9, "CHEMISTRY": 0.9, "MATHEMATICS": 0.9},
}

thinking_orientation_indices = {
    "ENTJ": {"CAUTIOUS": 0.7, "INSPIRING": 0.8, "DOMINANT": 0.9, "SUPPORTIVE": 0.6},
    "INTJ": {"CAUTIOUS": 0.8, "INSPIRING": 0.7, "DOMINANT": 0.8, "SUPPORTIVE": 0.9},
    "ENTP": {"CAUTIOUS": 0.6, "INSPIRING": 0.7, "DOMINANT": 0.8, "SUPPORTIVE": 0.9},
    "INTP": {"CAUTIOUS": 0.9, "INSPIRING": 0.8, "DOMINANT": 0.7, "SUPPORTIVE": 0.6},
    "ENFJ": {"CAUTIOUS": 0.8, "INSPIRING": 0.7, "DOMINANT": 0.8, "SUPPORTIVE": 0.7},
    "INFJ": {"CAUTIOUS": 0.9, "INSPIRING": 0.8, "DOMINANT": 0.8, "SUPPORTIVE": 0.9},
    "ENFP": {"CAUTIOUS": 0.8, "INSPIRING": 0.7, "DOMINANT": 0.8, "SUPPORTIVE": 0.6},
    "INFP": {"CAUTIOUS": 0.6, "INSPIRING": 0.7, "DOMINANT": 0.6, "SUPPORTIVE": 0.6},
}

# Define the analytical, technical, creative, and people orientation indices
analytical_index = thinkingOrientation["analyticalIndex"]
technical_index = thinkingOrientation["technicalIndex"]
creative_index = thinkingOrientation["creativeIndex"]
people_orientation_index = thinkingOrientation["peopleOrientationIndex"]

# Define a list of career tracks along with their suitability indices
list_of_career_tracks = []

# Read career tracks from the CSV file
with open("career_tracks.csv", mode="r") as career_tracks_file:
    reader = csv.DictReader(career_tracks_file)
    for row in reader:
        track_name = row.get("Track", "")
        suitability_index_str = row.get("suitability_index", "")  # Corrected key

        # Check if both track name and suitability index are not empty
        if track_name and suitability_index_str:
            suitability_index = int(suitability_index_str)
            list_of_career_tracks.append({"track": track_name, "suitability_index": suitability_index})

# Define the personality type for the single student
personality_type = "INTJ"  # Replace with the actual personality type

# Create a dictionary to store suitability scores for the single student
suitability_scores = {}
```

```
# Iterate through each career track and calculate scores for the single student
for career_track in list_of_career_tracks:
    suitability_index = career_track["suitability_index"]
    P = suitability_index * 10
    C = sum(sample_scores[subject] * cognitive_indices[personality_type][subject] for subject in sample_scores)

    # Corrected key to match the career track name
    T = analytical_index + technical_index + creative_index + people_orientation_index + sum(thinkingOrientation["peopleOrientation"][subject]*thinking_orientation_indices[personality_

    # Calculate the final score for the student in this career track
    final_score = P + C + T

    # Store the suitability score for this career track
    suitability_scores[career_track["track"]] = final_score

print(final_score)

# Find the career tracks where the suitability score matches with the student
matching_career_tracks = [track for track, score in suitability_scores.items() if score == final_score]
sorted_career_tracks = matching_career_tracks[:1]
print("Top Career Track : ", sorted_career_tracks)
```

```
201.73302
Top Career Track :  ['Software Programmer / Architect']
```