Project Documentation: Health Calculator

📌 Overview

Health Calculator is a basic Python program that calculates:

- ✅ **BMI** (Body Mass Index)
- 🔥 **BMR** (Basal Metabolic Rate)
- 💪 **TDEE** (Total Daily Energy Expenditure)
- 🩸 **Diabetic Risk Level**
- ❤️ **Heart Disease Risk Level**
- 🥗 **Suggested Diet Plan**

It supports both **metric** and **imperial** units and stores the final health analysis in a **JSON file**.

⚙️ Features

- Dual unit system (metric or imperial)
- Risk analysis for diabetes and heart disease
- Auto-calculated BMI, BMR, TDEE
- Suggested diet plans based on risk
- Saves user data in `health_data.json`

🛠️ Requirements

- Python 3.x
- No external libraries needed (uses built-in `json`)

📌Code

```
import json
def Calculate_BMI (weight, height):
    if unit=="imperial":
      weight = weight * 0.453592  # lbs to kg
      height = height * 0.0254    # inches to meters
      bmi = weight / (height ** 2)
    else:
      bmi= ( weight / (height ** 2))
    return round(bmi, 2)

def Calculate_BMR(weight, height, age):
    if unit=="imperial":
      bmr= 66 + (6.23 * weight) + (12.7 * height) - (6.8 * age)
    else:
      bmr= 655 + (9.6 * weight) + (1.8 * height) - (4.7 * age)
    return round(bmr, 2)

def Calculate_TDEE(bmr, activity_level):
    activity_multiplier = {
        "sedentary": 1.2,
        "lightly active": 1.375,
        "moderately active": 1.55,
        "very active": 1.725,
        "super active": 1.9
    }
    if activity_level in activity_multiplier:
        tdee = bmr * activity_multiplier[activity_level]
        return round(tdee, 2)
    else:
        raise ValueError("Invalid activity level provided.")

def Calculate_diabetic(bmi,glucose, family_history):
    if bmi < 18.5:
        return "Underweight"
    elif 18.5 <= bmi < 24.9:
        if glucose < 100 and not family_history:
            return "Normal"
        else:
            return "Prediabetes"
    elif 25 <= bmi < 29.9:
        if glucose < 100 and not family_history:
            return "Overweight"
        else:
            return "Prediabetes"
    else:
        if glucose < 100 and not family_history:
```

```
            return "Obese"
        else:
            return "Diabetes"

def Calculate_heart_disease(bmi, age, cholesterol, blood_pressure):
    if bmi < 18.5:
        return "Underweight"
    elif 18.5 <= bmi < 24.9:
        if age < 45 and cholesterol < 200 and blood_pressure < 120:
            return "Low risk"
        else:
            return "Moderate risk"
    elif 25 <= bmi < 29.9:
        if age < 45 and cholesterol < 200 and blood_pressure < 120:
            return "Moderate risk"
        else:
            return "High risk"
    else:
        if age < 45 and cholesterol < 200 and blood_pressure < 120:
            return "High risk"
        else:
            return "Very high risk"
def Suggested_diet(risk_level):
    diets = {
        "Underweight": "High-calorie diet with protein-rich foods. Include nuts, full-fat dairy,
bananas, avocados, peanut butter, eggs, and lean meats.",
        "Normal": "Balanced diet with a variety of nutrients. Include whole grains, seasonal
fruits, vegetables, lean meats or plant proteins, and moderate healthy fats.",
        "Overweight": "Calorie deficit diet with controlled portions. Emphasize steamed
vegetables, oats, lean protein, drink water before meals, and avoid sugary snacks.",
        "Obese": "Low-calorie diet with high fiber and low sugar. Eat salads, legumes, whole
grains, lean protein, and avoid deep-fried or packaged food.",
        "Prediabetes": "Low-carb diet with high fiber and healthy fats. Choose brown rice,
beans, leafy greens, flaxseed, walnuts, and avoid white bread, rice, and soft drinks.",
        "Diabetes": "Low-carb, high-fiber diet with controlled portions. Focus on non-starchy
vegetables, chia seeds, fish, whole grains (like quinoa), and limit sweets, juices.",
        "Low risk": "Heart-healthy diet with whole grains and lean proteins. Eat oats, grilled
chicken, lentils, almonds, and cook with olive or mustard oil.",
        "Moderate risk": "Heart-healthy diet with reduced saturated fats. Include fish, green
vegetables, switch to low-fat milk, and avoid red meat and butter.",
        "High risk": "Heart-healthy diet with low sodium and high fiber. Eat barley, apples,
spinach, and unsalted nuts. Avoid processed and salty foods.",
        "Very high risk": "Strict heart-healthy diet with medical supervision. Eat small, frequent
meals with low GI foods, avoid red meat, and consult a doctor or dietitian regularly."
    }
    if risk_level in diets:
        return diets[risk_level]
    else:
```

```
        return diets.get(risk_level, "Consult a healthcare provider for personalized advice.")

def save_to_json(data, filename="health_data.json"):
    with open(filename, 'w') as file:
        json.dump(data, file, indent=4)

def get_health_data():
    data = {
        "unit": input("Enter unit (imperial/metric): ").strip().lower(),
        "weight": float(input("Enter your weight: ")),
        "height": float(input("Enter your height: ")),
        "age": int(input("Enter your age: ")),
        "activity_level": input("Enter activity level (sedentary/lightly active/moderately
active/very active/super active): ").strip().lower(),
        "glucose": float(input("Enter your glucose level: ")),
        "family_history": input("Do you have a family history of diabetes? (yes/no):
").strip().lower() == 'yes',
        "cholesterol": float(input("Enter your cholesterol level: ")),
        "blood_pressure": float(input("Enter your blood pressure: "))
    }
    return data

def main():
    print("Welcome to the Health Calculator!")
    print("Please enter your health data below:")
    print("Note: Ensure all inputs are in the same unit (imperial or metric).")
    print("Enter Your name: ",end="")
    name = str(input())
    global unit
    health_data = get_health_data()
    unit = health_data["unit"]
    bmi = Calculate_BMI(health_data["weight"], health_data["height"])
    bmr = Calculate_BMR(health_data["weight"], health_data["height"], health_data["age"])
    tdee = Calculate_TDEE(bmr, health_data["activity_level"])

    diabetic_risk = Calculate_diabetic(bmi, health_data["glucose"],
health_data["family_history"])
    heart_disease_risk = Calculate_heart_disease(bmi, health_data["age"],
health_data["cholesterol"], health_data["blood_pressure"])

    diet_suggestion = Suggested_diet(diabetic_risk if diabetic_risk in ["Diabetes",
"Prediabetes"] else heart_disease_risk)
    print("\n📊 Health Analysis Results:")
    print(f"👤 Name: {name}")
    print(f"\n📊 Your BMI: {bmi}")
    print(f"🔥 Your BMR: {bmr}")
    print(f"💪 Your TDEE: {tdee}")
    print(f"🩸 Diabetic Risk Level: {diabetic_risk}")
```

```
    print(f"❤️ Heart Disease Risk Level: {heart_disease_risk}")
    print(f"🥗 Suggested Diet: {diet_suggestion}")

    results = {
        "Name": name,
        "Unit": unit,
        "BMI": bmi,
        "BMR": bmr,
        "TDEE": tdee,
        "Diabetic Risk": diabetic_risk,
        "Heart Disease Risk": heart_disease_risk,
        "Diet Suggestion": diet_suggestion
    }

    save_to_json(results)
    print("Health data saved to health_data.json")
    print(json.dumps(results, indent=4))

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

📋 How It Works

1. User Inputs:

- Unit system (`metric` or `imperial`)
- Weight and height
- Age and activity level
- Glucose level
- Family history of diabetes
- Cholesterol and blood pressure

2. Calculations Performed:

- **BMI** = weight / height²
- **BMR** = based on Mifflin-St Jeor formula (separate for each unit)
- **TDEE** = BMR × activity multiplier

3. Risk Assessment:

- Based on BMI, glucose, family history → Diabetic Risk
- Based on BMI, age, cholesterol, blood pressure → Heart Disease Risk

4. Suggested Diet:

- Custom diet plan based on risk level

5. Output:

- Display results on screen
- Save full report to `health_data.json`

🧪 Sample Output

Welcome to the Health Calculator!
Please enter your health data below:
Note: Ensure all inputs are in the same unit (imperial or metric).
Enter Your name: Roaida

📊 Health Analysis Results:
👤 Name: Nabiha
📊 Your BMI: 24.8
🔥 Your BMR: 1420.5
💪 Your TDEE: 1963.0
🩸 Diabetic Risk Level: Prediabetes
❤️ Heart Disease Risk Level: Moderate risk
🥗 Suggested Diet: Low-carb diet with high fiber and healthy fats...

Health data saved to health_data.json

📁 File Structure

health_calculator.py
health_data.json
README.md ← (You can use this documentation here)

📚 Function Descriptions

Function	Purpose
`Calculate_BMI()`	Computes BMI from height & weight
`Calculate_BMR()`	Calculates BMR based on age, height, weight
`Calculate_TDEE()`	Estimates calories needed per day
`Calculate_diabetic()`	Assesses diabetes risk
`Calculate_heart_disease()`	Assesses heart disease risk
`Suggested_diet()`	Suggests a diet based on risk level
`save_to_json()`	Saves user results to a `.json` file

`get_health_data()`	Collects all user inputs
`main()`	Runs the full program flow

🚀 Future Improvements

- GUI version using `tkinter` or `streamlit`
- Data visualization using `matplotlib`
- Login and save multiple users' health records
- Email the report to the user