Personal Health Planner

App Overview

The BMI Calculator and Personal Health Planner is an application designed to help users calculate their Body Mass Index (BMI), Total Daily Energy Expenditure (TDEE), and macronutrient needs based on their personal health goals. It guides users through inputting their height, weight, age, gender, activity level, and health goals to provide a comprehensive health plan.

Key Features and Concepts

1. **User Input Handling**:
    - The app prompts users for inputs like height, weight, age, gender, and activity level.
    - It includes validation to ensure inputs fall within reasonable ranges, using `try-except` blocks for error handling.
2. **BMI Calculation**:
    - The `calculate_bmi` function computes BMI
    - It classifies BMI into categories such as Underweight, Normal Weight, Overweight, and Obese.
3. **TDEE and BMR Calculation**:
    - The `bmr_tdee` function calculates the Basal Metabolic Rate (BMR) based on the user's gender, height, weight, and age.
    - It adjusts BMR according to the user's activity level to compute TDEE, which estimates daily caloric needs.
4. **Macronutrient Breakdown** :
    - The `macro_calculator` function determines the daily macronutrient requirements (protein, fat, and carbohydrates) based on the user's TDEE and health goals (losing fat, maintaining weight, or building muscle).
    - It adjusts caloric intake according to the user's weight change goals (e.g., gaining or losing weight).

5. **Conditional Logic**:

   - The app uses conditional statements (`if-elif-else`) to process user inputs and apply different formulas based on the input values.

   - It includes dictionaries to map user choices (e.g., activity levels) to corresponding values for calculations.

Code Structure and Flow:

- **Main Function** : Orchestrates the flow of the app, guiding users through input collection, performing calculations, and displaying results.

- **Helper Functions** : Modularizes the code into functions (`calculate_bmi`, `bmr_tdee`, `macro_calculator`) for better readability and maintenance.

- **Input Validation**: Ensures user inputs are within valid ranges, providing error messages and re-prompting as necessary.

Example Output:

Based on user inputs, the app outputs the BMI value, classification, daily caloric needs, and a detailed macronutrient breakdown, helping users understand their nutritional requirements.

This app combines fundamental programming concepts such as user input handling, arithmetic calculations, conditional logic, and modular design to deliver a comprehensive health planning tool. It's an excellent example of how programming can be applied to create useful, real-world applications.

PersonalHealthPlannner.py (Main Module)

```
import bmi_calculator
import calorie_calculator as cl
import macro_calculator as mc

print("\n>>>>>>-> Welcome To BMI Calculator And Personal Health Planner <-<<<<<<<")
print('------------------------------------------------------------------------\n')

while True:
    try:

        h  = int(input("Please Enter Your Height in centimeters : \t"))

        if h not in range(50,181):
            raise ValueError("\nInvalid Height\n")

        height = h
        break

    except ValueError as e:

        print(e)

while True:
    try:

        w  = int(input("\nPlease Enter Your Weight in Kilo Grams  : \t"))
```

```
        if w not in range(30,181):
            raise ValueError("\nInvalid weight\n")
        weight = w
        break
    except ValueError as e:
        print(e)

while True:
    try:
        ag = int(input("\nPlease Enter Your Age                : \t"))

        if ag not in range(1,101):
            raise ValueError("\nInavalid Age\n")
        age = ag
        break

    except ValueError as e:
        print(e)


print(f"\nBased on the details given by you, your BMI is : {bmi_calculator.calculate_bmi(height,weight)[0]}")

print(f"\nAnd You are classified as {bmi_calculator.calculate_bmi(height,weight)[1]}")

print("-------------------------------------------------------------------------\n")

while True:
```

```
    try:

        g = int(input("What is your current Goal ?\n\n1) Lose Fat\t2) Maintain\t3) Build Muscle ? (Enter The number)\t"))

        if g not in [1,2,3]:

            raise ValueError("\nInvalid Option\n")

        goal = {1:'lose fat',2:'maintain',3:'build muscle'}[g]

        break

    except ValueError as e:
        print(e)

weekly_weight = 0

if goal == 'build muscle' or goal == 'lose fat':

    while True:

        try:

            w_goal = int(input("\nIf building or losing, how much weight per week? (Options : 1) 0.25 kg, 2) 0.5 kg, 3) 1 kg) (enter the number)\n"))

            if w_goal not in [1,2,3]:
```

```
            raise ValueError('\nInvalid Option\n')

            weekly_weight = {1:0.25,2:0.5,3:1}[w_goal]
            break
        except ValueError as e:

            print(e)

gender        = input("\nPlease Enter Your Gender (M/F)        : \t")

while True:

    try:

        activity      = int(input("\nSelect Your Activity Level : \n\n1) sedentary, \n2) lightly active, \n3) moderately active, \n4) very active \n\nEnter the number corresponding to your activity level  : \t"))
        print("-------------------------------------------------------------------------\n")

        if activity not in [1,2,3,4]:

            raise ValueError('\nInvalid choice\n')

        activity_level = {1:'sedentary',2:'lightly active',3:'moderately active',4:'very active'}[activity]
        break

    except ValueError as e:
        print(e)
```

```
tdee = cl.bmr_tdee(gender,activity_level,height,weight,age)[0]

result = mc.macro_calculator(tdee,weight,goal,weekly_weight)

print(f"To {goal} with weekly weight goal of {weekly_weight}")
print("-----------------------------------------------------")
print("Daily Calorie and Macronutrient Breakdown:")
print(f"Calories  : {result['Daily Calories Needed']} kcal")
print(f"Protein   : {result['Protein']} g")
print(f"Fat       : {result['Fat']} g")
print(f"Carbs     : {result['Carbohydrates']} g")
print("\n<-----------------------------------------------------")

input("\nPress Enter To Exit Program")
```

bmi_calculator.py

```
def calculate_bmi(height,weight):
    height_in_meters = height / 100
    bmi = round(weight/(height_in_meters**2))
    if bmi < 18.5:
        classification = "Under Weight"
    elif 18.5 <= bmi < 24.9:
        classification = "Normal Weight"
    elif 25 <= bmi < 29.9:
        classification = "Over Weight"
    else:
        classification = "Obese"
    return [bmi,classification]
```

calorie_calculator.py

```
def bmr_tdee(gender,activity_level,height,weight,age):

    if gender.lower() == "m":

        bmr = 88.362 + (13.397 * weight) + (4.799 * height) - (5.667 * age)

    else:

        bmr = 447.593 + (9.247 * weight) + (3.098 * height) - (4.330 * age)

    activity_factors = {
    "sedentary": 1.2,
    "lightly active": 1.375,
    "moderately active": 1.55,
    "very active": 1.725
    }

    tdee = bmr * activity_factors[activity_level]

    return [tdee,bmr]
```

macro_calculator.py

```
import calorie_calculator as cl

def macro_calculator(tdee,weight,goal,weekly_weight):

    calorie_adjustment = (weekly_weight * 7700) / 7
    adjusted_tdee = tdee + calorie_adjustment

    if goal == 'lose fat':

        adjusted_tdee = tdee - calorie_adjustment
        protein = 1.5 * weight
        fat     = 0.25 * adjusted_tdee / 9

    elif goal == 'build muscle':

        protein = 2.0 * weight
        fat     = 0.2 * adjusted_tdee / 9

    else:

        protein = 1.8 * weight
        fat     = 0.25 * adjusted_tdee / 9

    protein_cal = protein * 4
    fat_cal = fat * 9
    carb_cal = adjusted_tdee - (protein_cal + fat_cal)
    carbs = carb_cal / 4
```

```
return {

    "Daily Calories Needed" : round(adjusted_tdee,2),
    "Protein"               : round(protein,2),
    "Fat"                   : round(fat,2),
    "Carbohydrates"         : round(carbs,2)
}
```