Home Work 7

July 22, 2024

Dylan Liesenfelt

1 Exercise 1: Meal and Nutrition Tracker

Objective: Develop code that allows users to log daily food intake, track nutritional values, and monitor their diet against personal health goals, supporting informed food choices and dietary objectives.

Classes and Components:

1.1 FoodItem

Variables:

name (private), calories (private), proteins (private), carbs (private), fats (private)

Instance Methods:

___init___(self, name, calories, proteins, carbs, fats): Constructor to initialize a new food item with nutritional info. Getter Methods for each private variable and a display method to print the food item.

```python
class FoodItem:
    def __init__(self,name,calories,proteins,carbs,fats):
        self.__name = name
        self.__calories = calories
        self.__proteins = proteins
        self.__carbs = carbs
        self.__fats = fats

    def getName(self):
        return self.__name

    def getCalories(self):
        return self.__calories

    def getProteins(self):
        return self.__proteins

    def getCarbs(self):
```

```
        return self.__carbs

    def getFats(self):
        return self.__fats

    def display(self):
        print(f'Food Item: {self.__name}, Calories: {self.__calories}, Proteins:
 ↪ {self.__proteins}g, Carbs: {self.__carbs}g, Fat: {self.__fats}g')
```

1.2 DailyLog

Variables:

date (private), food_items (a list of FoodItem instances, private)

Instance Methods:

__init__(self, date): Constructor to initialize a new daily log.

add_food_item: Adds a FoodItem instance to the log.

get_total_calories: Calculates total calories consumed on that day.

get_total_nutrients: Calculates total proteins, carbs, and fats consumed.

display: Print the daily log.

```
[ ]: class DailyLog:
    def __init__(self, date):
        self.__date = date
        self.__food_items = []

    def getDate(self):
        return self.__date

    def add_food_item(self, food_item):
        self.__food_items.append(food_item)

    def get_total_calories(self):
        sum = 0
        for item in self.__food_items:
            sum += item.getCalories()
        return sum

    def get_total_nutrients(self):
        sumProtein, sumCarb, sumFat, = 0,0,0
        for item in self.__food_items:
            sumProtein += item.getProteins()
            sumCarb += item.getCarbs()
            sumFat += item.getFats()
        return sumProtein, sumCarb, sumFat
```

```python
    def display(self):
        print(f'Daily Log: {self.__date}')
        for item in self.__food_items:
            item.display()
```

1.3 NutritionProfile:

Variables:

user_id (private), daily_logs (a dictionary with dates as keys and DailyLog instances as values)

Instance Methods:

__init__(self, user_id): Constructor to initialize a new nutrition profile.

add_daily_log(self, daily_log): Adds a DailyLog instance to the profile.

get_log_by_date(self, date): Retrieves a DailyLog by date.

display: Print the nutrition profile.

```python
class NutritionProfile:
    def __init__(self, user_id):
        self.__user_id = user_id
        self.__daily_logs = {}

    def add_daily_log(self, daily_log):
        self.__daily_logs[daily_log.getDate()] = daily_log

    def get_log_by_date(self, date):
        return self.__daily_logs.get(date)

    def display(self):
        print(f'Nutrition Profile: {self.__user_id}')
        for log in self.__daily_logs:
            self.__daily_logs[log].display()
```

1.4 Testing:

```python
# Creating some food items
orange  = FoodItem('Orange', 60, 0.9, 11, 0.1)
egg = FoodItem('Egg', 78, 6, 0.6, 5)
avocado = FoodItem('Avocado', 240, 3, 13, 22)


# Create a daily log and add food items
daily_log = DailyLog('2023-04-02')
daily_log.add_food_item(orange)
daily_log.add_food_item(egg)
```

```
daily_log.add_food_item(avocado)

# Create a nutrition profile and add the daily log
profile = NutritionProfile('User1') # type: ignore
profile.add_daily_log(daily_log)

# Testing outputs using display
orange.display()
egg.display()
avocado.display()
print('\n')
daily_log.display()
print('\n')
profile.display()
```

```
Food Item: Orange, Calories: 60, Proteins: 0.9g, Carbs: 11g, Fat: 0.1g
Food Item: Egg, Calories: 78, Proteins: 6g, Carbs: 0.6g, Fat: 5g
Food Item: Avocado, Calories: 240, Proteins: 3g, Carbs: 13g, Fat: 22g


Daily Log: 2023-04-02
Food Item: Orange, Calories: 60, Proteins: 0.9g, Carbs: 11g, Fat: 0.1g
Food Item: Egg, Calories: 78, Proteins: 6g, Carbs: 0.6g, Fat: 5g
Food Item: Avocado, Calories: 240, Proteins: 3g, Carbs: 13g, Fat: 22g


Nutrition Profile: User1
Daily Log: 2023-04-02
Food Item: Orange, Calories: 60, Proteins: 0.9g, Carbs: 11g, Fat: 0.1g
Food Item: Egg, Calories: 78, Proteins: 6g, Carbs: 0.6g, Fat: 5g
Food Item: Avocado, Calories: 240, Proteins: 3g, Carbs: 13g, Fat: 22g
```

2 Exercise 2: Create a Module

Objective:

Save the classes FoodItem, DailyLog, and NutritionProfile into a Python file named nutrition_tracker.py.

This file will act as your module and import it to another jupyter notebook to produce the same output as in Exercise 1.

```
[ ]: import nutrition_tracker

     # Create some food items
     apple = nutrition_tracker.FoodItem("Apple", 95, 0.5, 25, 0.3)
     banana = nutrition_tracker.FoodItem("Banana", 105, 1.3, 27, 0.3)
```

```python
# Create a daily log and add food items
daily_log = nutrition_tracker.DailyLog("2023-04-02")
daily_log.add_food_item(apple)
daily_log.add_food_item(banana)

# Create a nutrition profile and add the daily log
profile = nutrition_tracker.NutritionProfile("User1")
profile.add_daily_log(daily_log)

# Testing outputs using display
apple.display()
banana.display()
print('\n')
daily_log.display()
print('\n')
profile.display()
```

```
Food Item: Apple, Calories: 95, Proteins: 0.5g, Carbs: 25g, Fat: 0.3g
Food Item: Banana, Calories: 105, Proteins: 1.3g, Carbs: 27g, Fat: 0.3g


Daily Log: 2023-04-02
Food Item: Apple, Calories: 95, Proteins: 0.5g, Carbs: 25g, Fat: 0.3g
Food Item: Banana, Calories: 105, Proteins: 1.3g, Carbs: 27g, Fat: 0.3g



Nutrition Profile: User1
Daily Log: 2023-04-02
Food Item: Apple, Calories: 95, Proteins: 0.5g, Carbs: 25g, Fat: 0.3g
Food Item: Banana, Calories: 105, Proteins: 1.3g, Carbs: 27g, Fat: 0.3g
```

3   Exercise 3: BONUS - Plot the daily log.

Explore the package matplotlib and create a bar plot from the daily log. Implement the plot as another method, example: daily_log.plot_nutrients()

```python
import matplotlib.pyplot as plt

class PlotDailyLog(DailyLog):
    def __init__(self, date):
        super().__init__(date)

    def plot_nutrients(self):
        nutrients = ['Proteins', 'Carbs', 'Fats']
        totalOfNutrients = self.get_total_nutrients()
        colors = ['blue', 'orange', 'green']
```

```python
        plt.bar(nutrients, totalOfNutrients, color=colors)
        plt.xlabel('Nutrients')
        plt.ylabel('Grams')
        plt.title(f'Total Nutrients for {self.getDate()}')
        plt.show()

newLog = PlotDailyLog('2024-07-22')
newLog.add_food_item(apple)
newLog.add_food_item(banana)
newLog.add_food_item(avocado)
newLog.add_food_item(egg)
newLog.add_food_item(orange)

newLog.plot_nutrients()
```


Total Nutrients for 2024-07-22