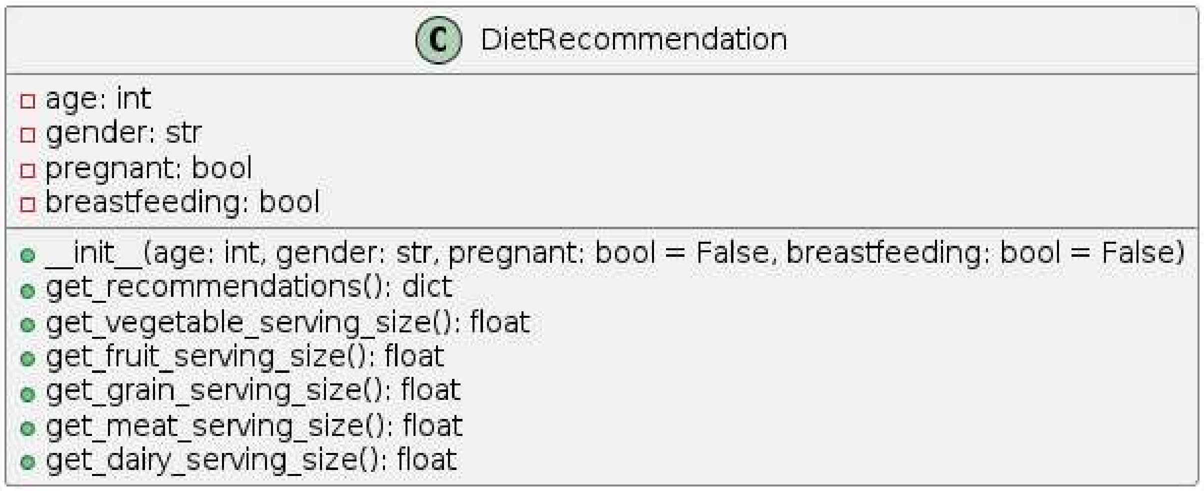
UML Class Diagram:


C
DietRecommendation
age: int
gender: str
pregnant: bool
breastfeeding: bool
__init__(age: int, gender: str, pregnant: bool = False, breastfeeding: bool = False)
get_recommendations(): dict
get_vegetable_serving_size(): float
get_fruit_serving_size(): float
get_grain_serving_size(): float
get_meat_serving_size(): float
get_dairy_serving_size(): float

Assignment2.py Code:

```
class DietRecommendation:
    def __init__(self, age, gender, pregnant=False, breastfeeding=False):
        self.age = age
        self.gender = gender
        self.pregnant = pregnant
        self.breastfeeding = breastfeeding

    def get_recommendations(self):
        recommendations = {
            "vegetables": self.get_vegetable_serving_size(),
            "fruits": self.get_fruit_serving_size(),
            "grains": self.get_grain_serving_size(),
            "meats": self.get_meat_serving_size(),
            "dairy": self.get_dairy_serving_size()
        }

        return recommendations

    def get_vegetable_serving_size(self):
        serving_size = 0
        if self.age in range(2, 4):
            serving_size = 2.5
        elif self.age in range(4, 9):
            serving_size = 4.5
        elif self.age in range(9, 12):
            serving_size = 5
        elif self.age in range(12, 14):
            if self.gender == 'male':
                serving_size = 5.5
            elif self.gender == 'female':
                serving_size = 5
        elif self.age in range(14, 19):
            if self.gender == 'male':
                serving_size = 5.5
            elif self.gender == 'female':
                serving_size = 5
        elif self.age in range(19, 51):
            if self.gender == "male":
```

```python
                serving_size = 6
            if self.gender == "female":
                serving_size = 5
            if self.pregnant and self.gender=='female':
                serving_size = 5
            if self.breastfeeding and self.gender=='female':
                serving_size = 7.5
        elif self.age in range(51, 71):
            if self.gender == "male":
                serving_size = 5.5
            elif self.gender == "female":
                serving_size = 5
        elif self.age >= 71:
            serving_size = 5
        return serving_size

    def get_fruit_serving_size(self):
        serving_size = 0
        if self.age in range(2, 4):
            serving_size = 1
        elif self.age in range(4, 9):
            serving_size = 1.5
        elif self.age in range(9, 12):
            serving_size = 2
        elif self.age in range(12, 14):
            serving_size = 2
        elif self.age in range(14, 19):
            serving_size = 2
        elif self.age in range(19, 51):
            serving_size = 2
        elif self.age in range(51, 71):
            serving_size = 2
        elif self.age >= 71:
            serving_size = 2
        return serving_size

    def get_grain_serving_size(self):
        serving_size = 0
        if self.age in range(2, 4):
            serving_size = 4
        elif self.age in range(4, 9):
            serving_size = 4
        elif self.age in range(9, 12):
            if self.gender == 'male':
                serving_size = 5
            elif self.gender == 'female':
                serving_size = 4
        elif self.age in range(12, 14):
            if self.gender == 'male':
                serving_size = 6
            elif self.gender == 'female':
                serving_size = 5
        elif self.age in range(14, 19):
            serving_size = 7
        elif self.age in range(19, 51):
            if self.gender == "male":
                serving_size = 6
            if self.gender == "female":
                serving_size = 6
            if self.pregnant:
                serving_size = 8.5
            if self.breastfeeding:
                serving_size = 9
        elif self.age in range(51, 71):
            if self.gender == "male":
                serving_size = 6
            if self.gender == "female":
                serving_size = 4
        elif self.age >= 71:
            if self.gender == 'male':
                serving_size = 4.5
            elif self.gender == 'female':
                serving_size = 3
        return serving_size

    def get_meat_serving_size(self):
        serving_size = 0
        if self.age in range(2, 4):
            serving_size = 1
        elif self.age in range(4, 9):
            serving_size = 1.5
        elif self.age in range(9, 12):
            serving_size = 2.5
        elif self.age in range(12, 14):
            serving_size = 2.5
        elif self.age in range(14, 19):
            serving_size = 2.5
        elif self.age in range(19, 51):
            if self.gender == "male":
                serving_size = 3
            if self.gender == "female":
                serving_size = 2.5
            if self.pregnant:
                serving_size = 3.5
            if self.breastfeeding:
                serving_size = 2.5
        elif self.age in range(51, 71):
            if self.gender == "male":
                serving_size = 2.5
            if self.gender == "female":
                serving_size = 2
        elif self.age >= 71:
            if self.gender == 'male':
                serving_size = 2.5
            elif self.gender == 'female':
                serving_size = 2
        return serving_size

    def get_dairy_serving_size(self):
        serving_size = 0
        if self.age in range(2, 4):
            serving_size = 1.5
        elif self.age in range(4, 9):
            if self.gender == 'male':
                serving_size = 2
            else:
                serving_size = 1.5
        elif self.age in range(9, 12):
            if self.gender == 'male':
                serving_size = 2.5
            elif self.gender == 'female':
                serving_size = 3
        elif self.age in range(12, 14):
            serving_size = 3.5
        elif self.age in range(14, 19):
            serving_size = 3.5
        elif self.age in range(19, 51):
            serving_size = 2.5
        elif self.age in range(51, 71):
            if self.gender == "male":
                serving_size = 2.5
            if self.gender == "female":
                serving_size = 4
        elif self.age >= 71:
            if self.gender == 'male':
                serving_size = 3.5
            elif self.gender == 'female':
                serving_size = 4
        return serving_size


def print_food_descriptions():
    print("\nAdditionally, here are detailed descriptions of what
constitutes a single serving for each food category:")
    print(
        "\nVEGETABLES\nSingle serve of vegetable is 75g (100-350kJ). Here
are some examples of single serve of Vegetable:")
    print("• 0.5 cup of cooked green or orange vegetables (like broccoli,
spinach, carrots, or pumpkin)")
    print("• 0.5 cup of cooked dried or canned beans, peas, or lentils")
    print("• 1 cup of green leafy or raw salad vegetables")
    print("• 0.5 cup of sweet corn")
    print("• 0.5 of a medium potato or other starchy vegetables (such as
sweet potato, taro, or cassava)")
    print("• 1 medium tomato")

    print("\nFRUITS\nSingle serve of fruit is 150g (350kJ). Here are some
examples of single serve of fruit:")
    print("• 1 medium apple, banana, orange, or pear")
    print("• 2 small apricots, kiwi fruits, or plums")
    print("• 1 cup of diced or canned fruit (with no added sugar)")

    print("\nGRAINS\nSingle serve of grain is (500kJ). Here are some
examples of single serve of grain:")
    print("• 1 slice (40g) of bread")
    print("• 0.5 medium (40g) roll or flat bread")
    print("• 0.5 cup (75-120g) of cooked rice, pasta, noodles, barley,
buckwheat, semolina, polenta, bulgur or quinoa")
    print("• 0.5 cup (120g) of cooked porridge")
    print("• 0.66 cup (30g) of wheat cereal flakes")
    print("• 0.75 cup (30g) of muesli")
    print("• 3 (35g) crispbreads")
    print("• 1 (60g) crumpet")
    print("• 1 small (35g) English muffin or scone")

    print("\nMEAT\nSingle serve of meat is (500-600kJ). Here are some
examples of single serve of meat:")
    print("• 65g cooked lean red meats such as beef, lamb, veal, pork,
goat, or kangaroo (about 90-100g raw)")
    print("• 80g cooked lean poultry such as chicken or turkey (100g raw)")
    print("• 100g cooked fish fillet (about 115g raw) or one small can of
fish")
    print("• 2 large (120g) eggs")
    print("• 1 cup (150g) cooked or canned legumes/beans such as lentils, 
chickpeas, or split peas")
    print("• 170g tofu")
    print("• 30g nuts, seeds, peanut or almond butter, tahini, or other nut 
or seed paste")

    print("\nDAIRY\nSingle serve of dairy is (500-600kJ). Here are some 
examples of single serve of dairy:")
    print("• 1 cup (250ml) of fresh, UHT long life, reconstituted powdered 
milk or buttermilk")
    print("• 0.5 cup (120ml) of evaporated milk")
    print("• 2 slices (40g) or a 4 x 3 x 2 cm cube (40g) of hard cheese, 
such as cheddar")
    print("• 0.5 cup (120g) of ricotta cheese")
    print("• 0.25 cup (200g) of yoghurt")
    print("• 1 cup (250ml) of soy, rice or other cereal drink with at least 
100mg of added calcium per 100ml")


def main():
    try:
        age = int(input("Enter your age: "))
        if age <= 0:
            raise ValueError("Age must be a positive integer.")

        if age>120:
            raise ValueError("This can't be a human's age!")

        gender = input("Enter your gender (male/female): ").lower()
        if gender not in ['male', 'female']:
            raise ValueError("Gender must be 'male' or 'female'.")

        pregnant_input = input("Are you pregnant? (yes/no): ").lower()
        if pregnant_input not in ['yes', 'no']:
            raise ValueError("Please enter 'yes' or 'no'.")
        pregnant = pregnant_input == "yes"

        breastfeeding_input = input("Are you breastfeeding? (yes/no): 
").lower()
        if breastfeeding_input not in ['yes', 'no']:
            raise ValueError("Please enter 'yes' or 'no'.")
        breastfeeding = breastfeeding_input == "yes"

        if gender == "male" and (pregnant or breastfeeding):
            raise ValueError("Men cannot be pregnant and breastfeeding at 
the same time.")

        if age>=51 and age<=70  and (pregnant or breastfeeding):
            raise ValueError("This is not normal to be pregnant in this 
age")

        if age<19 and (pregnant or breastfeeding):
            raise ValueError("Too young to be pregnant!")

        recommendation = DietRecommendation(age, gender, pregnant, 
breastfeeding)
        recommendations = recommendation.get_recommendations()

        print("\nBased on your inputs, the minimum recommended servings 
are:")
        for food_group, serving_size in recommendations.items():
            print(f"{food_group.capitalize()}: {serving_size} servings per
day")

        print_food_descriptions()

        export = input("\nWould you like to export these recommendations
and serving sizes? (yes/no): ").lower()
        if export == "yes":
            filename = input("Enter a filename for the export (leave blank
for 'DietaryRecommendations.txt'): ").strip()
            if filename and not filename.endswith(".txt"):
                raise ValueError("Filename must end with '.txt'.")
            filename = filename if filename else
"DietaryRecommendations.txt"
            with open(filename, "w") as file:
                file.write("Dietary Recommendations\n\n")
                for food_group, serving_size in recommendations.items():
                    file.write(f"{food_group.capitalize()}: {serving_size}
servings per day\n")
                file.write(
                    "\nAdditionally, here are detailed descriptions of what
constitutes a single serving for each food category:\n")
                file.write(
                    "\nVEGETABLES\nSingle serve of vegetable is 75g (100-
350kJ). Here are some examples of single serve of Vegetable:\n")
                file.write("• 0.5 cup of cooked green or orange vegetables
(like broccoli, spinach, carrots, or pumpkin)\n")
                file.write("• 0.5 cup of cooked dried or canned beans,
peas, or lentils\n")
                file.write("• 1 cup of green leafy or raw salad
vegetables\n")
                file.write("• 0.5 cup of sweet corn\n")
                file.write(
                    "• 0.5 of a medium potato or other starchy vegetables
(such as sweet potato, taro, or cassava)\n")
                file.write("• 1 medium tomato\n")

                file.write(
                    "\nFRUITS\nSingle serve of fruit is 150g (350kJ). Here
are some examples of single serve of fruit:\n")
                file.write("• 1 medium apple, banana, orange, or pear\n")
                file.write("• 2 small apricots, kiwi fruits, or plums\n")
                file.write("• 1 cup of diced or canned fruit (with no added
sugar)\n")

                file.write("\nGRAINS\nSingle serve of grain is (500kJ).
Here are some examples of single serve of grain:\n")
                file.write("• 1 slice (40g) of bread\n")
                file.write("• 0.5 medium (40g) roll or flat bread\n")
                file.write(
                    "• 0.5 cup (75-120g) of cooked rice, pasta, noodles,
barley, buckwheat, semolina, polenta, bulgur or quinoa\n")
                file.write("• 0.5 cup (120g) of cooked porridge\n")
                file.write("• 0.66 cup (30g) of wheat cereal flakes\n")
                file.write("• 0.75 cup (30g) of muesli\n")
                file.write("• 3 (35g) crispbreads\n")
                file.write("• 1 (60g) crumpet\n")
                file.write("• 1 small (35g) English muffin or scone\n")

                file.write("\nMEAT\nSingle serve of meat is (500-600kJ).
Here are some examples of single serve of meat:\n")
```

```
                file.write(
                    "• 65g cooked lean red meats such as beef, lamb, veal,
pork, goat, or kangaroo (about 90-100g raw)\n")
                file.write("• 80g cooked lean poultry such as chicken or
turkey (100g raw)\n")
                file.write("• 100g cooked fish fillet (about 115g raw) or
one small can of fish\n")
                file.write("• 2 large (120g) eggs\n")
                file.write("• 1 cup (150g) cooked or canned legumes/beans
such as lentils, chickpeas, or split peas\n")
                file.write("• 170g tofu\n")
                file.write("• 30g nuts, seeds, peanut or almond butter,
tahini, or other nut or seed paste\n")

                file.write(
                    "\nDAIRY\nSingle serve of dairy is (500-600kJ). Here
are some examples of single serve of dairy:\n")
                file.write("• 1 cup (250ml) of fresh, UHT long life,
reconstituted powdered milk or buttermilk\n")
                file.write("• 0.5 cup (120ml) of evaporated milk\n")
                file.write("• 2 slices (40g) or a 4 x 3 x 2 cm cube (40g)
of hard cheese, such as cheddar\n")
                file.write("• 0.5 cup (120g) of ricotta cheese\n")
                file.write("• 0.25 cup (200g) of yoghurt\n")
                file.write(
                    "• 1 cup (250ml) of soy, rice or other cereal drink
with at least 100mg of added calcium per 100ml\n")

            print(f"Recommendations and serving sizes have been exported to
{filename}")

    except ValueError as e:
        print(f"Error: {e}. Please enter valid input.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")


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

Description:

Functionality:

The program is designed to provide personalized dietary recommendations based on user inputs such as age, gender, and whether they are pregnant or breastfeeding. It calculates the recommended servings of various food groups including vegetables, fruits, grains, meats, and dairy products according to established guidelines. Additionally, it allows the user to export these recommendations to a text file if desired.

Design Decisions:

1. Class Structure: The program utilizes a single class DietRecommendation to encapsulate all the functionality related to generating dietary recommendations. This class follows the

principle of encapsulation, bundling together data (attributes) and behaviors (methods) that operate on the data.

2. Modular Functions: Each food groups serving size calculation is separated into individual methods within the DietRecommendation class, promoting code readability and maintainability.

3. User Interaction: The program interacts with the user through the command line interface (CLI), prompting for input using input() function calls and displaying output accordingly. This provides a straightforward way for users to input their information and receive recommendations.

4. Export Functionality: The program offers the option to export the recommendations to a text file, enhancing user convenience by allowing them to save the information for future reference.

Exception Handling Approach:

1. ValueError Handling: The program employs try-except blocks to catch ValueError exceptions raised when the user inputs invalid data, such as non-numeric age, non-binary gender, or incorrect responses to pregnancy/breastfeeding questions.

2. Custom Error Messages: Custom error messages are provided to guide the user in correcting their input. For example, if the user enters a negative age or an age over 120, the program informs them that the input must be a positive integer and within a reasonable range.

3. Logical Validation: The program includes logical validations to ensure that certain combinations of inputs are not allowed, such as a male user being pregnant or breastfeeding, or users in specific age ranges being pregnant or breastfeeding, which might not be biologically plausible.

4. Catch-All Exception Handling: An outer except block is included to handle any unexpected errors that may occur during program execution. This helps to prevent crashes and provides a more graceful error handling experience for the user.

Class Attributes:

- age: Represents the age of the individual for whom the dietary recommendations are being calculated.

- gender: Represents the gender of the individual, either male or female.

- pregnant: Represents whether the individual is pregnant. Defaults to False.

- breastfeeding: Represents whether the individual is breastfeeding. Defaults to False.

Methods:

1. __init__(self, age, gender, pregnant=False, breastfeeding=False):

- Constructor method that initializes the DietRecommendation object with the provided age, gender, pregnancy status, and breastfeeding status.

2. get_recommendations(self):

   - Method that calculates and returns the recommended servings of different food groups based on age, gender, pregnancy, and breastfeeding status.

   - Returns a dictionary containing recommended servings for vegetables, fruits, grains, meats, and dairy.

3. get_vegetable_serving_size(self):

   - Method that calculates the recommended serving size for vegetables based on age, gender, pregnancy, and breastfeeding status.

   - Returns the recommended serving size in grams.

4. get_fruit_serving_size(self):

   - Method that calculates the recommended serving size for fruits based on age, gender, pregnancy, and breastfeeding status.

   - Returns the recommended serving size in grams.

5. get_grain_serving_size(self):

   - Method that calculates the recommended serving size for grains based on age, gender, pregnancy, and breastfeeding status.

   - Returns the recommended serving size in grams.

6. get_meat_serving_size(self):

   - Method that calculates the recommended serving size for meats based on age, gender, pregnancy, and breastfeeding status.

   - Returns the recommended serving size in grams.

7. get_dairy_serving_size(self):

   - Method that calculates the recommended serving size for dairy products based on age, gender, pregnancy, and breastfeeding status.

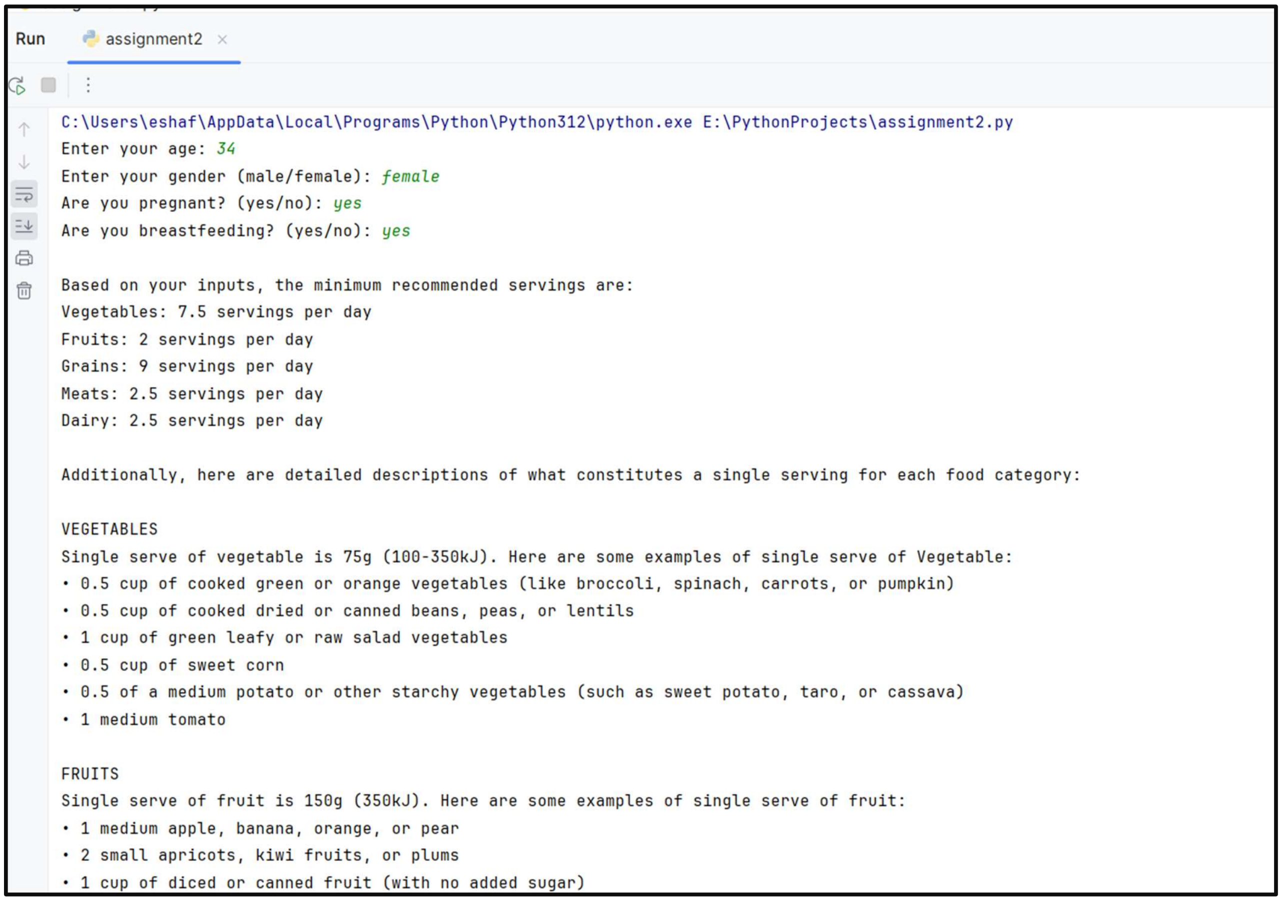
   - Returns the recommended serving size in grams.

Utility Function:

- print_food_descriptions():

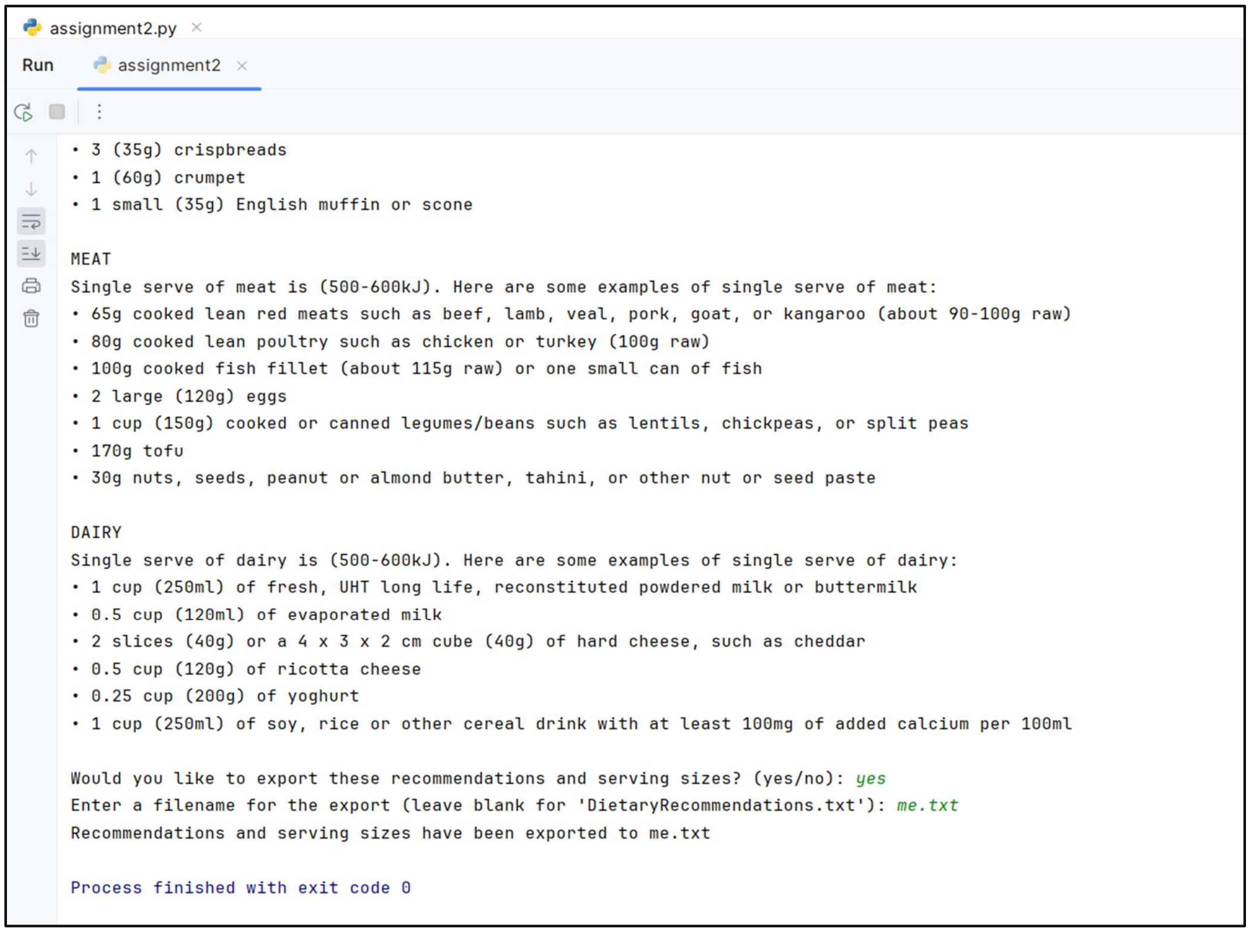
  - Utility function that prints detailed descriptions of what constitutes a single serving for each food category (vegetables, fruits, grains, meats, dairy). This function is not part of the DietRecommendation class but provides supplementary information to the user.

Outputs:



Run assignment2

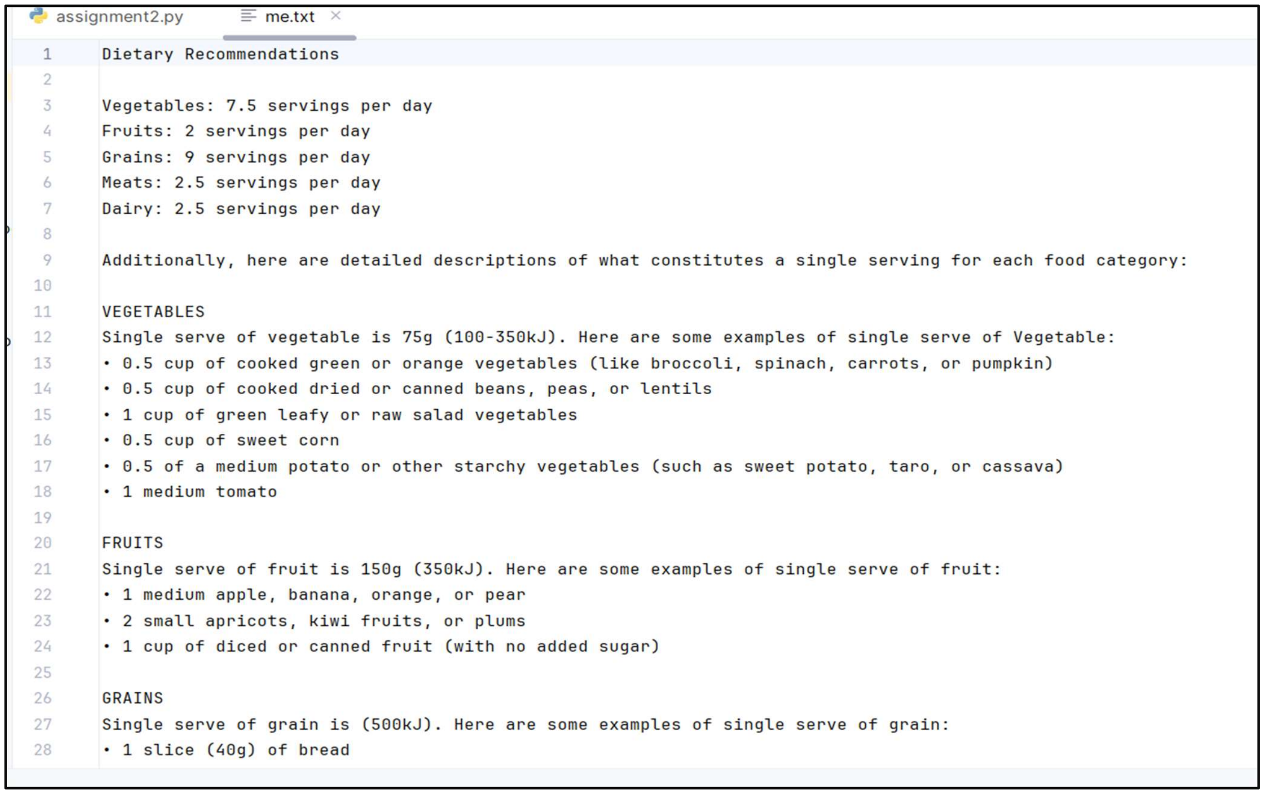
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 34
Enter your gender (male/female): female
Are you pregnant? (yes/no): yes
Are you breastfeeding? (yes/no): yes

Based on your inputs, the minimum recommended servings are:
Vegetables: 7.5 servings per day
Fruits: 2 servings per day
Grains: 9 servings per day
Meats: 2.5 servings per day
Dairy: 2.5 servings per day

Additionally, here are detailed descriptions of what constitutes a single serving for each food category:

VEGETABLES
Single serve of vegetable is 75g (100-350kJ). Here are some examples of single serve of Vegetable:
• 0.5 cup of cooked green or orange vegetables (like broccoli, spinach, carrots, or pumpkin)
• 0.5 cup of cooked dried or canned beans, peas, or lentils
• 1 cup of green leafy or raw salad vegetables
• 0.5 cup of sweet corn
• 0.5 of a medium potato or other starchy vegetables (such as sweet potato, taro, or cassava)
• 1 medium tomato

FRUITS
Single serve of fruit is 150g (350kJ). Here are some examples of single serve of fruit:
• 1 medium apple, banana, orange, or pear
• 2 small apricots, kiwi fruits, or plums
• 1 cup of diced or canned fruit (with no added sugar)
```



assignment2.py ×

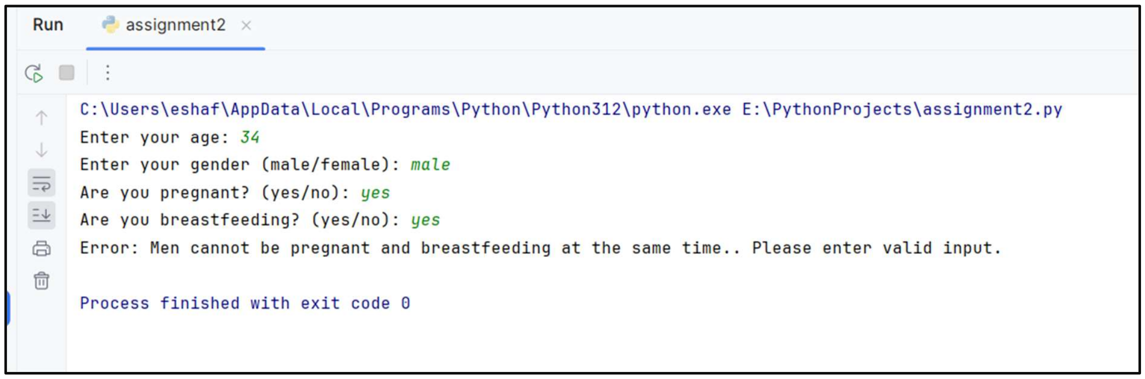
Run   assignment2 ×

```
• 3 (35g) crispbreads
• 1 (60g) crumpet
• 1 small (35g) English muffin or scone

MEAT
Single serve of meat is (500-600kJ). Here are some examples of single serve of meat:
• 65g cooked lean red meats such as beef, lamb, veal, pork, goat, or kangaroo (about 90-100g raw)
• 80g cooked lean poultry such as chicken or turkey (100g raw)
• 100g cooked fish fillet (about 115g raw) or one small can of fish
• 2 large (120g) eggs
• 1 cup (150g) cooked or canned legumes/beans such as lentils, chickpeas, or split peas
• 170g tofu
• 30g nuts, seeds, peanut or almond butter, tahini, or other nut or seed paste

DAIRY
Single serve of dairy is (500-600kJ). Here are some examples of single serve of dairy:
• 1 cup (250ml) of fresh, UHT long life, reconstituted powdered milk or buttermilk
• 0.5 cup (120ml) of evaporated milk
• 2 slices (40g) or a 4 x 3 x 2 cm cube (40g) of hard cheese, such as cheddar
• 0.5 cup (120g) of ricotta cheese
• 0.25 cup (200g) of yoghurt
• 1 cup (250ml) of soy, rice or other cereal drink with at least 100mg of added calcium per 100ml

Would you like to export these recommendations and serving sizes? (yes/no): yes
Enter a filename for the export (leave blank for 'DietaryRecommendations.txt'): me.txt
Recommendations and serving sizes have been exported to me.txt

Process finished with exit code 0
```

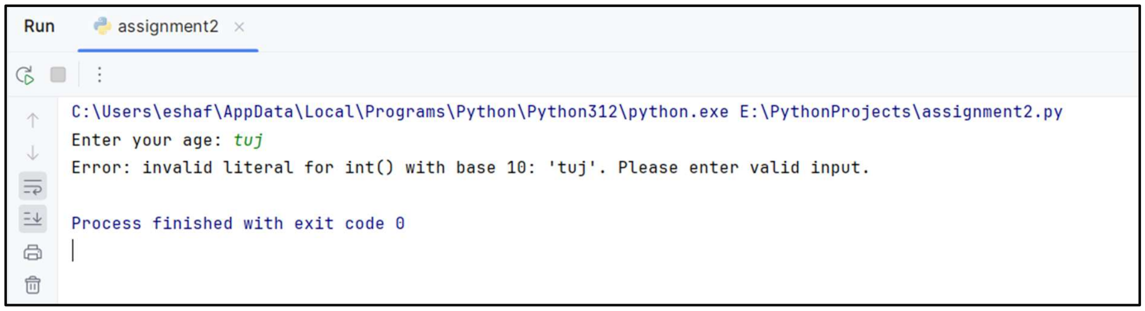


assignment2.py   me.txt ×

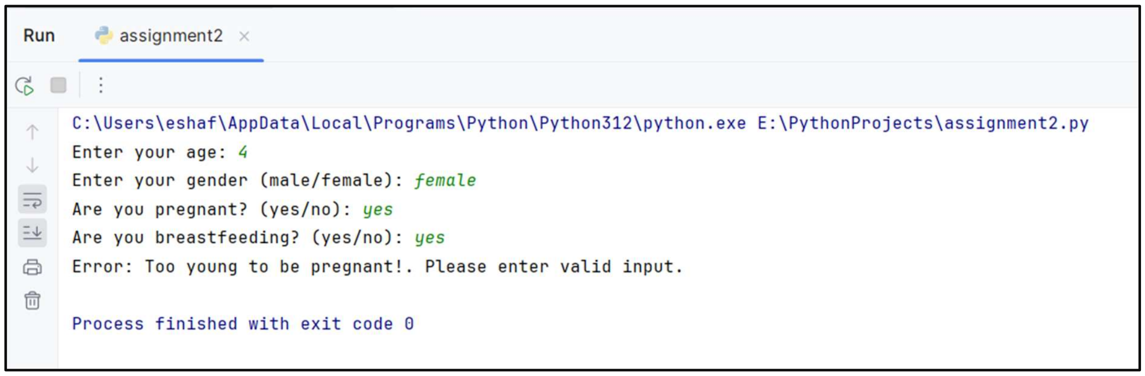
```
Dietary Recommendations

Vegetables: 7.5 servings per day
Fruits: 2 servings per day
Grains: 9 servings per day
Meats: 2.5 servings per day
Dairy: 2.5 servings per day

Additionally, here are detailed descriptions of what constitutes a single serving for each food category:

VEGETABLES
Single serve of vegetable is 75g (100-350kJ). Here are some examples of single serve of Vegetable:
• 0.5 cup of cooked green or orange vegetables (like broccoli, spinach, carrots, or pumpkin)
• 0.5 cup of cooked dried or canned beans, peas, or lentils
• 1 cup of green leafy or raw salad vegetables
• 0.5 cup of sweet corn
• 0.5 of a medium potato or other starchy vegetables (such as sweet potato, taro, or cassava)
• 1 medium tomato

FRUITS
Single serve of fruit is 150g (350kJ). Here are some examples of single serve of fruit:
• 1 medium apple, banana, orange, or pear
• 2 small apricots, kiwi fruits, or plums
• 1 cup of diced or canned fruit (with no added sugar)

GRAINS
Single serve of grain is (500kJ). Here are some examples of single serve of grain:
• 1 slice (40g) of bread
```



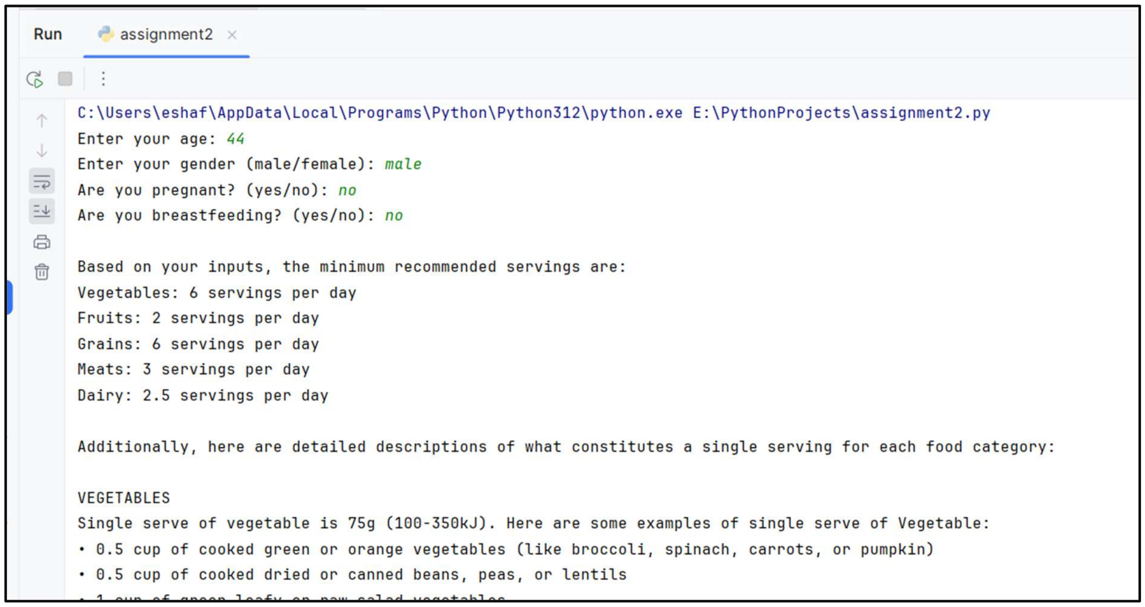
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 34
Enter your gender (male/female): male
Are you pregnant? (yes/no): yes
Are you breastfeeding? (yes/no): yes
Error: Men cannot be pregnant and breastfeeding at the same time.. Please enter valid input.

Process finished with exit code 0
```



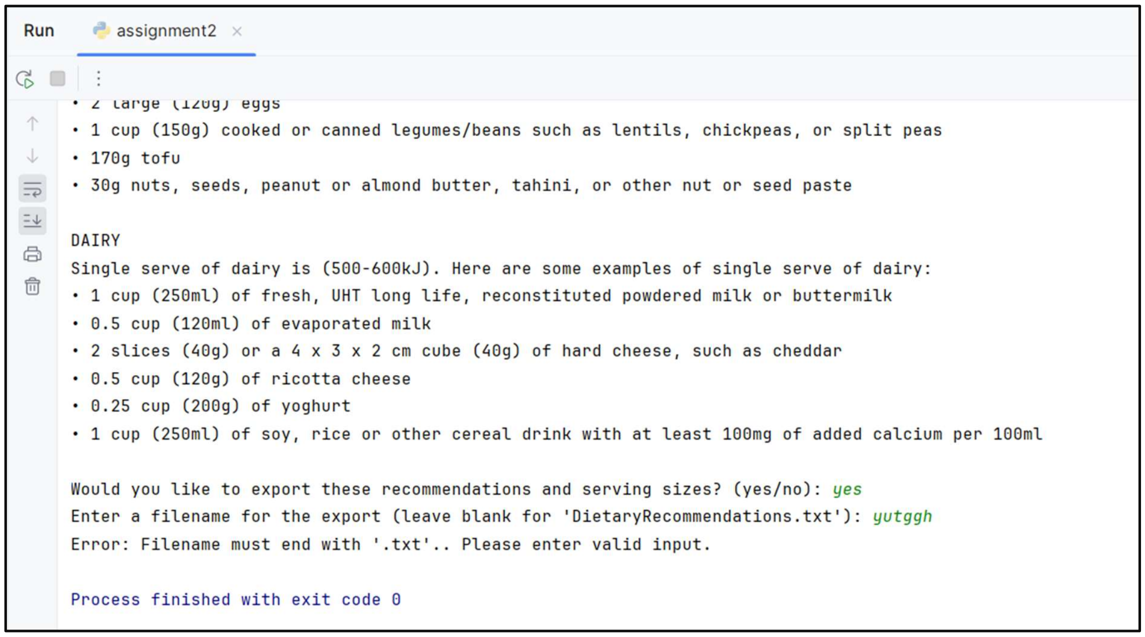
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: tuj
Error: invalid literal for int() with base 10: 'tuj'. Please enter valid input.

Process finished with exit code 0
```



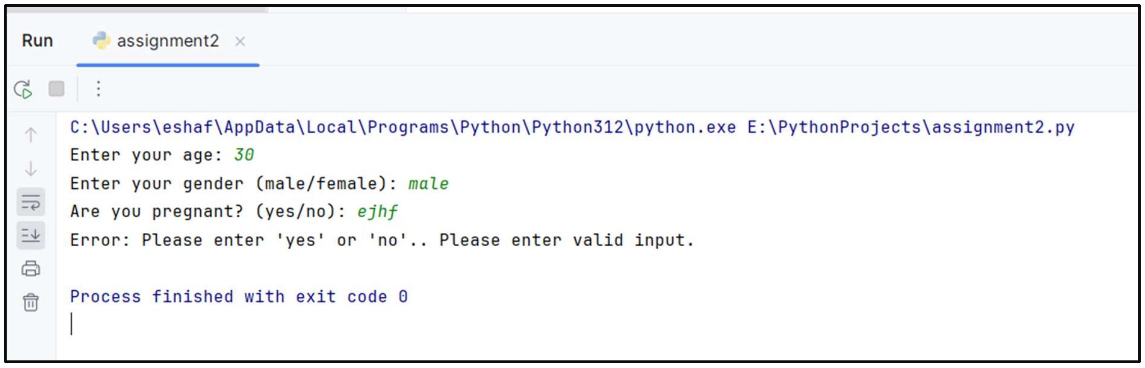
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 4
Enter your gender (male/female): female
Are you pregnant? (yes/no): yes
Are you breastfeeding? (yes/no): yes
Error: Too young to be pregnant!. Please enter valid input.

Process finished with exit code 0
```



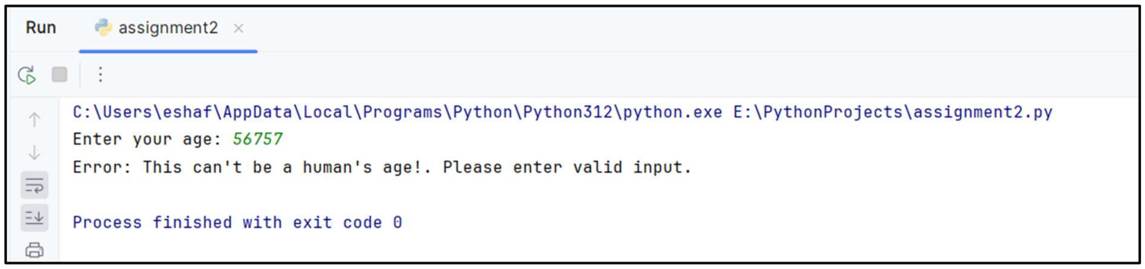
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 44
Enter your gender (male/female): male
Are you pregnant? (yes/no): no
Are you breastfeeding? (yes/no): no

Based on your inputs, the minimum recommended servings are:
Vegetables: 6 servings per day
Fruits: 2 servings per day
Grains: 6 servings per day
Meats: 3 servings per day
Dairy: 2.5 servings per day

Additionally, here are detailed descriptions of what constitutes a single serving for each food category:

VEGETABLES
Single serve of vegetable is 75g (100-350kJ). Here are some examples of single serve of Vegetable:
• 0.5 cup of cooked green or orange vegetables (like broccoli, spinach, carrots, or pumpkin)
• 0.5 cup of cooked dried or canned beans, peas, or lentils
```



```
• 2 large (120g) eggs
• 1 cup (150g) cooked or canned legumes/beans such as lentils, chickpeas, or split peas
• 170g tofu
• 30g nuts, seeds, peanut or almond butter, tahini, or other nut or seed paste

DAIRY
Single serve of dairy is (500-600kJ). Here are some examples of single serve of dairy:
• 1 cup (250ml) of fresh, UHT long life, reconstituted powdered milk or buttermilk
• 0.5 cup (120ml) of evaporated milk
• 2 slices (40g) or a 4 x 3 x 2 cm cube (40g) of hard cheese, such as cheddar
• 0.5 cup (120g) of ricotta cheese
• 0.25 cup (200g) of yoghurt
• 1 cup (250ml) of soy, rice or other cereal drink with at least 100mg of added calcium per 100ml

Would you like to export these recommendations and serving sizes? (yes/no): yes
Enter a filename for the export (leave blank for 'DietaryRecommendations.txt'): yutggh
Error: Filename must end with '.txt'.. Please enter valid input.

Process finished with exit code 0
```



```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 30
Enter your gender (male/female): male
Are you pregnant? (yes/no): ejhf
Error: Please enter 'yes' or 'no'.. Please enter valid input.

Process finished with exit code 0
```



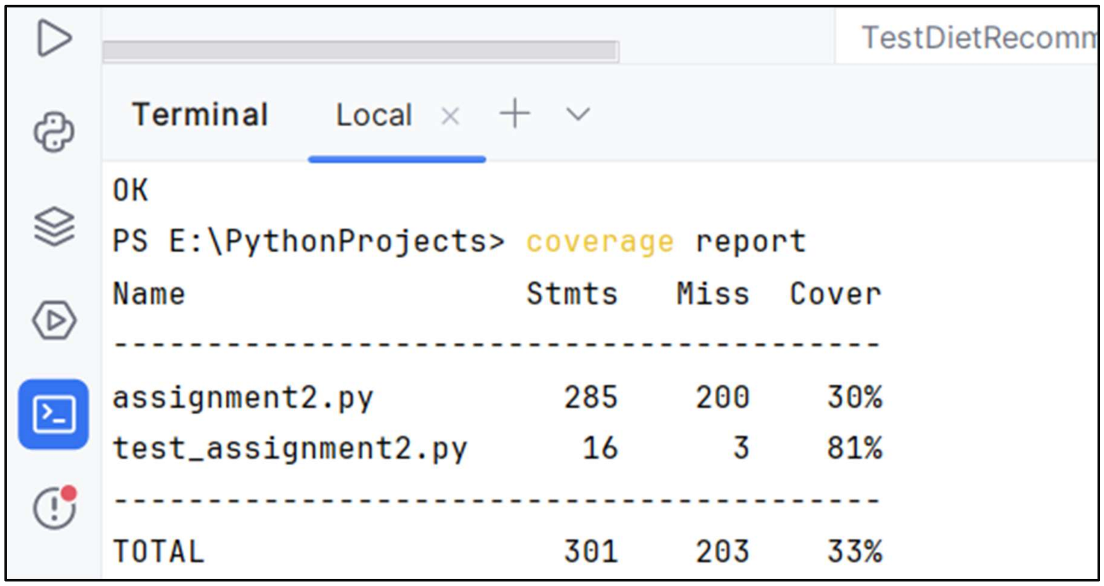
```
C:\Users\eshaf\AppData\Local\Programs\Python\Python312\python.exe E:\PythonProjects\assignment2.py
Enter your age: 56757
Error: This can't be a human's age!. Please enter valid input.

Process finished with exit code 0
```

Test_assignment.py code:

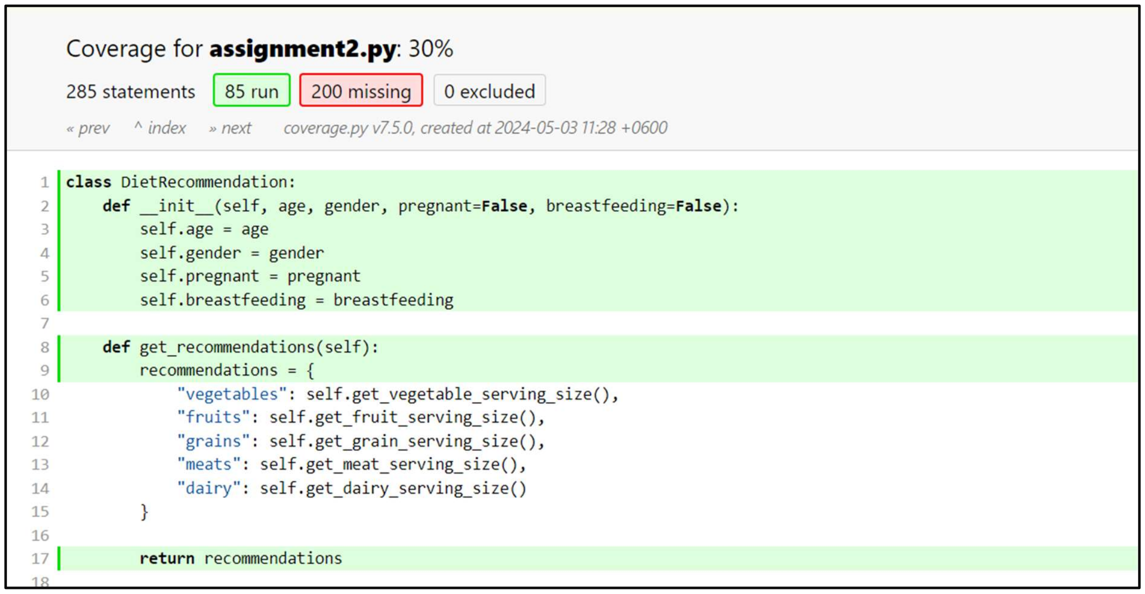
```
import unittest
from assignment2 import DietRecommendation

class TestDietRecommendation(unittest.TestCase):
    def test_get_recommendations(self):
        # Test case 1: Male, age > 19, not pregnant, not breastfeeding
        recommendation1 = DietRecommendation(35, "male", pregnant=False,
breastfeeding=False)
```

```
        expected_recommendations1 = {
            "vegetables": 6,
            "fruits": 2,
            "grains": 6,
            "meats": 3,
            "dairy": 2.5
        }
        self.assertEqual(recommendation1.get_recommendations(),
expected_recommendations1)

        # Test case 2: Female, age < 19, not pregnant, not breastfeeding
        recommendation2 = DietRecommendation(17, "female", pregnant=False,
breastfeeding=False)
        expected_recommendations2 = {
            "vegetables": 5,
            "fruits": 2,
            "grains": 7,
            "meats": 2.5,
            "dairy": 3.5
        }
        self.assertEqual(recommendation2.get_recommendations(),
expected_recommendations2)


        #Test case 2: Male, age>19 , pregnant ,breastfeeding
        try:
            DietRecommendation(34, "male", pregnant=True,
breastfeeding=True).get_recommendations()
        except ValueError as e:
            self.assertEqual(str(e), "Men cannot be pregnant and
breastfeeding at the same time.")


if __name__ == '__main__':
    unittest.main()
```

Code Coverage Report:



```
OK
PS E:\PythonProjects> coverage report
Name                  Stmts   Miss  Cover
-----------------------------------------
assignment2.py          285    200    30%
test_assignment2.py      16      3    81%
-----------------------------------------
TOTAL                   301    203    33%
```

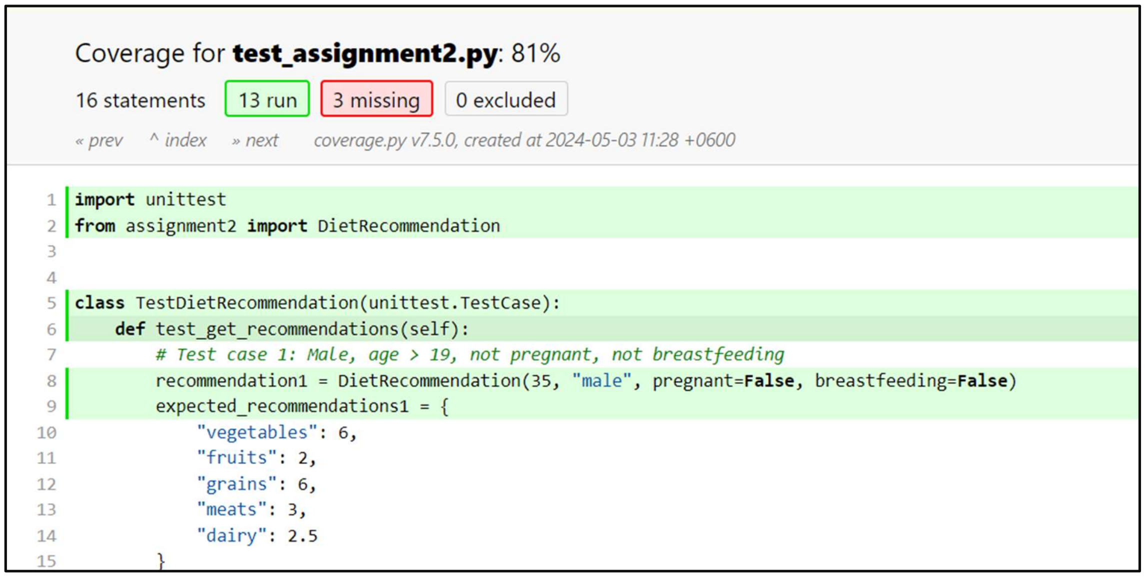


Coverage for **assignment2.py**: 30%

285 statements | 85 run | 200 missing | 0 excluded

« prev ^ index » next *coverage.py v7.5.0, created at 2024-05-03 11:28 +0600*

```python
class DietRecommendation:
    def __init__(self, age, gender, pregnant=False, breastfeeding=False):
        self.age = age
        self.gender = gender
        self.pregnant = pregnant
        self.breastfeeding = breastfeeding

    def get_recommendations(self):
        recommendations = {
            "vegetables": self.get_vegetable_serving_size(),
            "fruits": self.get_fruit_serving_size(),
            "grains": self.get_grain_serving_size(),
            "meats": self.get_meat_serving_size(),
            "dairy": self.get_dairy_serving_size()
        }

        return recommendations
```

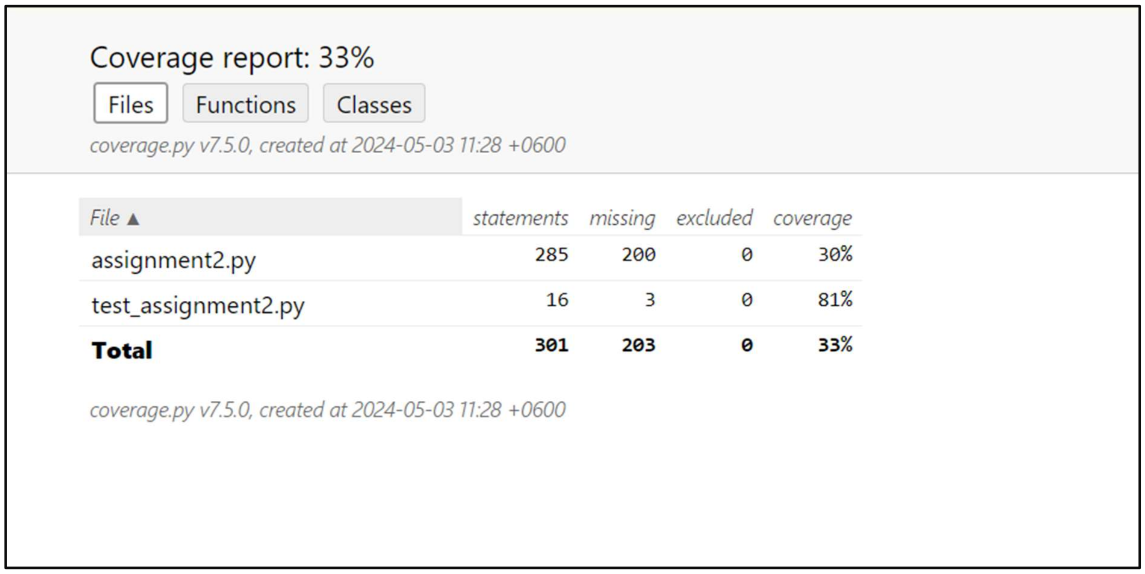


Coverage for **test_assignment2.py**: 81%

16 statements | 13 run | 3 missing | 0 excluded

« prev ^ index » next *coverage.py v7.5.0, created at 2024-05-03 11:28 +0600*

```python
import unittest
from assignment2 import DietRecommendation


class TestDietRecommendation(unittest.TestCase):
    def test_get_recommendations(self):
        # Test case 1: Male, age > 19, not pregnant, not breastfeeding
        recommendation1 = DietRecommendation(35, "male", pregnant=False, breastfeeding=False)
        expected_recommendations1 = {
            "vegetables": 6,
            "fruits": 2,
            "grains": 6,
            "meats": 3,
            "dairy": 2.5
        }
```

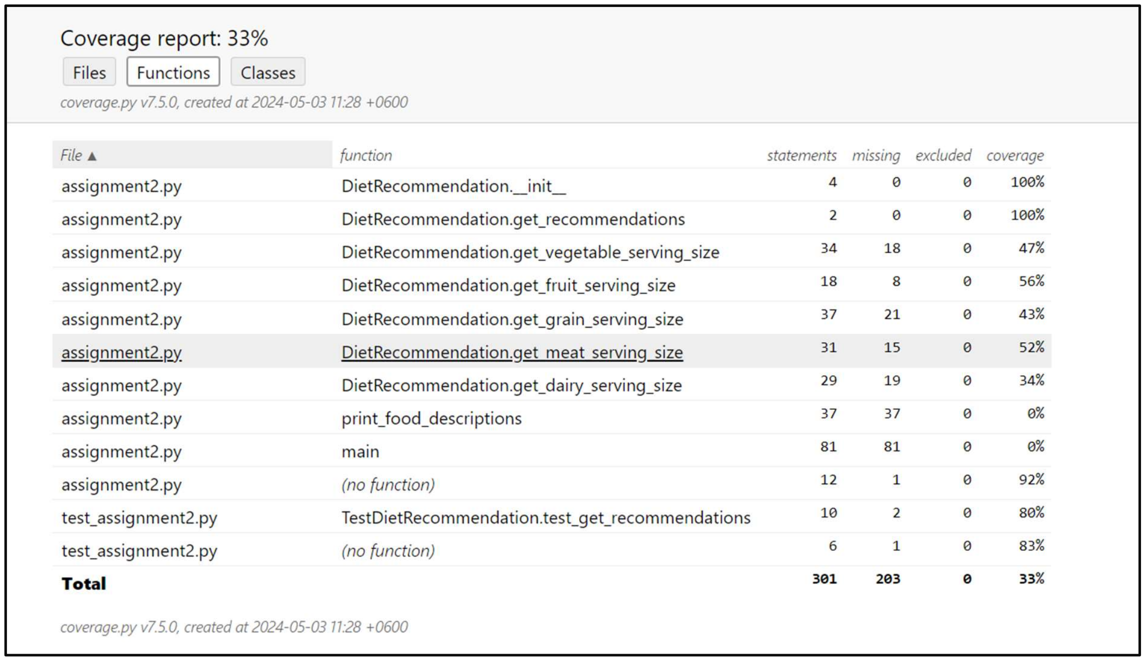


Coverage report: 33%

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600

File ▲	statements	missing	excluded	coverage
assignment2.py	285	200	0	30%
test_assignment2.py	16	3	0	81%
Total	**301**	**203**	**0**	**33%**

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600

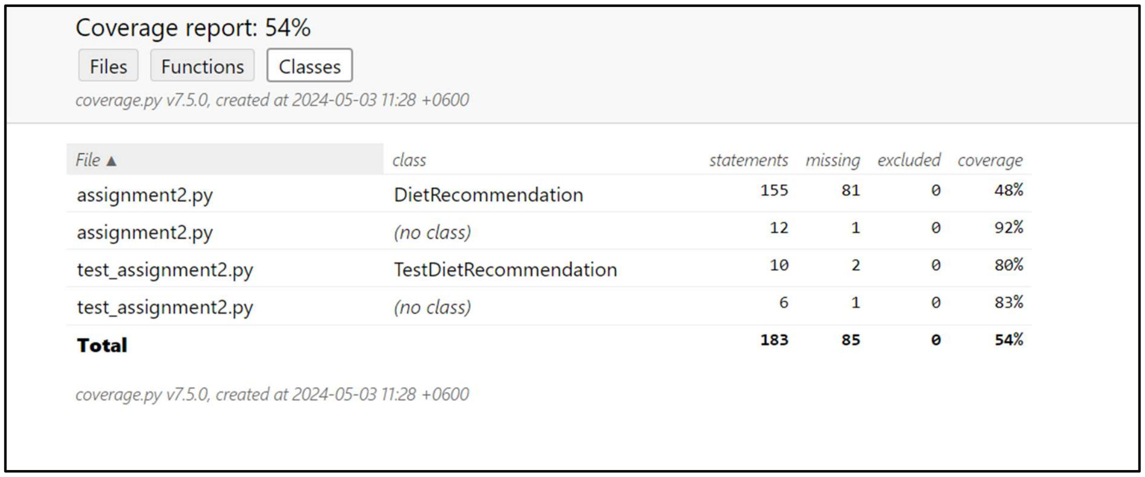


Coverage report: 33%

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600

File ▲	function	statements	missing	excluded	coverage
assignment2.py	DietRecommendation.__init__	4	0	0	100%
assignment2.py	DietRecommendation.get_recommendations	2	0	0	100%
assignment2.py	DietRecommendation.get_vegetable_serving_size	34	18	0	47%
assignment2.py	DietRecommendation.get_fruit_serving_size	18	8	0	56%
assignment2.py	DietRecommendation.get_grain_serving_size	37	21	0	43%
assignment2.py	DietRecommendation.get_meat_serving_size	31	15	0	52%
assignment2.py	DietRecommendation.get_dairy_serving_size	29	19	0	34%
assignment2.py	print_food_descriptions	37	37	0	0%
assignment2.py	main	81	81	0	0%
assignment2.py	*(no function)*	12	1	0	92%
test_assignment2.py	TestDietRecommendation.test_get_recommendations	10	2	0	80%
test_assignment2.py	*(no function)*	6	1	0	83%
Total		**301**	**203**	**0**	**33%**

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600



Coverage report: 54%

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600

File ▲	class	statements	missing	excluded	coverage
assignment2.py	DietRecommendation	155	81	0	48%
assignment2.py	*(no class)*	12	1	0	92%
test_assignment2.py	TestDietRecommendation	10	2	0	80%
test_assignment2.py	*(no class)*	6	1	0	83%
Total		**183**	**85**	**0**	**54%**

coverage.py v7.5.0, created at 2024-05-03 11:28 +0600