BMI Calculator Documentation

User Guide

Given a JSON file in the following format containing at least these column names:


```
{
        {"Gender": "Male", "HeightCm": 171, "WeightKg": 96 },
        { "Gender": "Male", "HeightCm": 161, "WeightKg": 85 },
        { "Gender": "Male", "HeightCm": 180, "WeightKg": 77 },
        { "Gender": "Female", "HeightCm": 166, "WeightKg": 62},
        {"Gender": "Female", "HeightCm": 150, "WeightKg": 70}, {
        "Gender": "Female", "HeightCm": 167, "WeightKg": 82}
}
```


the application performs the necessary calculation and adds 3 columns: 'BMI', 'BMI Category', 'Health Risk'. The base functionality involves a csv (of JSON) output of the original data including the additional 3 columns.

The BMI is calculated according to the following formula:

$$BMI(kg/m^2) = mass(kg) / height(m)^2$$

The BMI (Body Mass Index) in (kg/m^2) is equal to the weight in kilograms (kg) divided by your height in meters squared (m^2). For example, if you are `175cm (1.75m)` in height and `75kg` in weight, you can calculate your BMI as follows: `75kg / (1.75m²) = 24.49kg/m²`

The BMI Category and Health Risk are calculated according to the following criteria:

Table 1 - BMI Category and the Health Risk.

BMI Category	BMI Range (kg/m2)	Health risk
Underweight	18.4 and below	Malnutrition risk
Normal weight	18.5 - 24.9	Low risk
Overweight	25 - 29.9	Enhanced risk
Moderately obese	30 - 34.9	Medium risk
Severely obese	35 - 39.9	High risk
Very severely obese	40 and above	Very high risk

Functionality

After cloning the contents of this repo onto a server with python installed, you can perform the following via the terminal:

1- *>> python main_program.py -h*
   This provides a list of arguments to the application and their function
2- *>> python main_program.py -generatedatasize 10000*
   This generates a JSON file containing 10000 random patient data adhering to:
   a. half female and half male

b. female heights sampled from a normal distribution with mean 162cm and male heights sampled from a normal distribution with mean 176cm. Both with standard deviation 7cm. These were obtained from UK averages
c. female weights and male weights sampled from normal distributions with means 70.6kg and 83.9kg respectively with standard deviation 5kg. These were obtained from UK averages

3- *>> python main_program -loadpath 'health_parameters.json'*
This utilises the base functionality of the application by providing the path to a JSON file containing the patient data. The output will by default be a JSON file called 'health_parameters_output.json'

4- *>> python main_program.py -loadpath 'health_parameters.json' -category 'normal weight' -savefiletype 'csv'*
Extended usage. This runs the application by providing a JSON file called 'health_parameters.json' found locally to the code and the output file will be of type csv. Additionally, we request some additional information on the 'normal' BMI category. The additional information specifies that 22% of the population is 'Normal Weight' and the save location of the output file is '…/health_parameters_output.csv'. It looks like this:

```
(venv) (base) tanselarif@Tansels-MacBook-Pro pythonProject % python main_program.py -loadpath 'health_parameters.json' -category 'normal weight' -savefiletype 'csv'
The number of patients in category normal weight is 2200. This is 22.0% of the total population.
File saved to location /Users/tanselarif/PycharmProjects/pythonProject/health_parameters_output.csv
```

Technical Specifications

The project contains 4 modules and a requirements.txt file (all written in Python 3):
requirements.txt: The libraries the application depends upon. If creating a virtual environment, this can be used to match the python libraries with the versions used in the application development.
BMI_Calculator.py: contains the core functions for the application
Main_program.py: contains the source code for the main application functionality. Users interact with this modules directly
Generate_data.py: contains a function to generate dummy/random data as a JSON file
Unit_Testing.py: currently contains 3 unit tests. Running the module directly in a python terminal checks the stable functionality of the application:

a) **Unit Test 1**: Ingests the data shown at the top of this document and tests that the calculated BMIs match [24.9, 24.9, 18.5, 24.9, 24.9, 18.5] rounded to 2d.p. Also checks that the application returns the expected number of each BMI category
b) **Unit Test 2**: This is a test that the application is able to cater for large files (1 million patients) and give the correct results
c) **Unit Test 3**: This is a test that the application deals with edge cases according to the defined functionality. i.e. someone with a BMI of 24.9 should be defined as 'Normal Weight'