Code Explanation

In this document, we will thoroughly examine and comprehend the assumptions made, the flow of the code, the user inputs required, and the constants utilized.

Below is a breakdown of the code structure followed for this project:

HX_Project/

├── tests/

│ └── test_initial_file_modelling_case_study.py

├── Initial_File_modelling_case_study.py

└── parameters.json

└── modelling_case_study_functions.py

└── Code_Explaination

└── Output.txt

└── Modelling Case Study Given.zip

- **HX_Project/:** This is the root directory of the project.
- **Initial_File_modelling_case_study.py:** This is a Python file that appears to be the main script or entry point of your project. It likely contains the code for your case study or modelling task.
- **tests/:** This folder contains test files related to project. It includes test_initial_file_modelling_case_study.py, which contains unit test for the Initial_File_modelling_case_study.py file.
- **parameters.json:** This file is a JSON file that stores parameters given in Parameters spreadsheet **in "Initial File - hx Interview Model" excel file**. It is used to provide inputs to main script.
- **modelling_case_study_functions.py**: This file contains additional functions related to modelling case study. It includes helper functions which are used in Initial_File_modelling_case_study.py script.
- **Code_Explanation:** This is a file that contain notes about the code.
- **Output.txt:** This file stores the output of script for Base version.

Below is a breakdown of a code in terms of flow of program and functions used:

The code begins by importing the necessary modules and loading the **parameters** from a **JSON** file using the json module.

- The **riebesell** function calculates the **Riebesell ILF** based on the provided formula in given excel and the parameters loaded from the JSON file.

- The **calculate_hull_values** function calculates various values related to the **Hull Table** for each drone in the model data.

- The **calculate_tpl_values** function calculates various values related to the **TPL (Third-Party Liability) Table** for each drone in the model data.

- The **calculate_detachable_camera_values** function calculates values related to **Detachable cameras** table based on the model data.

- The **extension1_recalculate_drone_premium** function implements extension 1 of the case study, which involves charging different premiums for a subset of drones based on a maximum number of drones in the air.

- The **extension2_recalculate_camera_premium** function implements extension 2 of the case study, which involves charging different premiums for a subset of cameras based on the number of drones with detachable cameras.

- The **calculate_net_premium** function calculates various values related to the Net column for the Premium Summary.

- The **calculate_gross_premium** function calculates various values related to the Gross column for the Premium Summary.

- The **main** function is the **main entry point of the code**. It performs the rating calculations on the provided model data, including applying the extensions if enabled.

- Finally, the **if __name__ == '__main__':** block demonstrates how to use the functions by calling the main function with some example data and printing the calculated model data.

Note:

- The serial number for third drone in excel is **"CCC-333**" and that in provided data structure is **"AAA-123**" are with different numbers. To match the calculations given in the excel spreadsheet, **values of tpl_limit and tpl_excess are mapped in code for the drone "AAA-123**".
- The values **of tpl_ilf**,**tpl_layer_premium** in **drones** list of dictionary, **drones_hull, drones_tpl, cameras_hull** in **gross_prem** dict , **drones_hull, drones_tpl, cameras_hull in net_prem** dict are not rounded off in code intentionally as it would create a difference in final net and gross premium. (It occurs the values are rounded by excel and not in original calculations.
- User has to pass the values with **serial number, tpl_limit** and **tpl_excess** in a list of dictionary in main function as it is a User inputs. In case user does not want to provide value of **tpl_excess** it can be set to **None**.
- Gross amount is referred as the total amount before any deductions/fees are applied.
- Net Amount is referred to the amount remaining after deducting any fees/charges.
- Brokerage Fee is the percentage of the gross amount that is charged by a broker for their services.
- Extension 1 in code is referred as **enable_extension_1,** to enable this user has to pass enable_extension_1=True. (By default it is kept disabled in code.
- Extension 2 in code is referred as **enable_extension_2,** to enable this user has to pass enable_extension_2=True. (By default it is kept disabled in code.
- Both extensions could be enabled together as well.
- If the extensions are enabled it would affect the total net and gross which means it would not reflect the same values as provided in excel.
- It is assumed that inputs are valid, greater than 0 and positive.
- The code has been divided into two files to provide better readability.
- All calculations are implemented based on the formulas in excel.
- The values of **"Initial File - hx Interview Model.xlsx**" parameters spreadsheet are dumped into parameters.json in program.

Assumptions in Extension 1 (extension1_recalculate_drone_premium):

- For extension 1, values of list of dictionary of **drone** of **hull_premium** are replaced according to the provided summary. The net and gross will be based on the newly calculated values of **hull_premium**.

Assumptions in Extension 2 (extension2_recalculate_camera_premium):

- For extension 2, values of list of dictionary of **detachable_cameras** of **hull_premium** are replaced according to the provided summary. The net and gross will be based on the newly calculated values of **hull_premium.** If **total_drones_with_detachable_camera** is less than **max_camera_in_air** then **total_drones_with_detachable_camera** are taken as **max_camera_in_air otherwise** it is taken as **max_drones_in_air.**