	Firstname & lastname:			
	Group:	1CTAI		
	Bachelor:	CTAI	**Module:**	Basic Programming
	Lector(s):	Marie Dewitte	**Controlled by:**	Joerie Claerhout Frederiek Berthier Stijn Walcarius
	Year:	2023-2024	**Semester:**	1
	Date:	19-01-2024	**Exam period:**	1

1 Lab-exam

1.1 Info

- First read this exam calmly and carefully.
- Program the different exercises as described in the task.
- A program that deviates from this task may result in the loss of marks.
- You must:
  - Use the required naming convention if specified in the task.
  - Use the Python programming language.
- You may:
  - Use any teaching material on your laptop
  - Use the official documentation available at: https://docs.python.org/3/
- You are **NOT** allowed to
  - Use Copilot, chatGPT or any other AI system.
- Any irregularity (including GSM, cheating) will be reported to the student concerned and to the Chairman of the Examination Board in accordance with the OER.

1.2 Start - Github

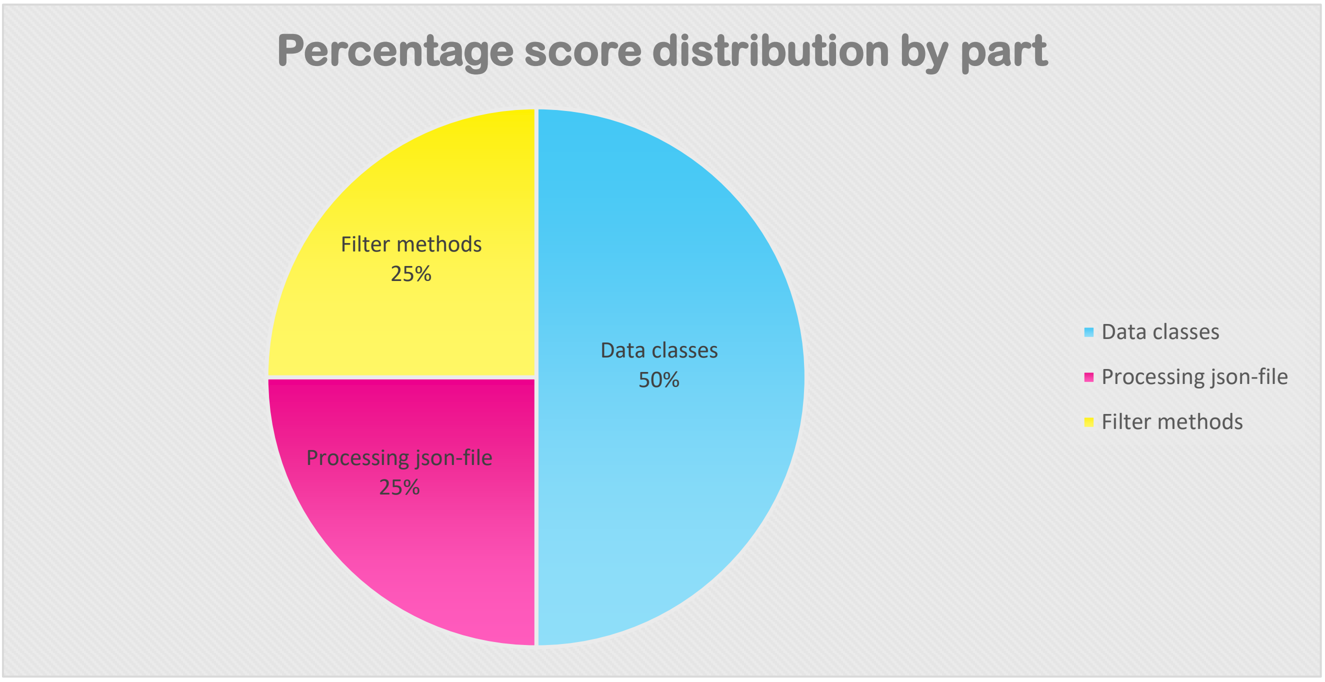
- Clone the repository: you can find the url on Leho.
- Put your name, first name and group in the comments at the top of each file.
- Continue working on the source files. **Make at least four addtional meaningfull commits.**

1.3 Submission

When you are finished, or when the specified time period has expired:

- Step 1 (GitHub): The version on Github will not be graded.
  - Save everything, commit & push everything to your repository one last time.
  - We will check if you have posted enough (meaningful) commits.

- Step 2 (Leho): The submitted version on Leho will be graded.
  - Locate the folder using explorer. Check that the folder contains all the necessary files.
  - Zip the folder. Make sure that everything is included in the zip file.
  - Submit your project on Leho for module assignments.
  - Check that you have submitted correctly by re-downloading your own solution.
    This is your own responsibility! *We will not accept any additional versions once the exam is finished under any circumstances.*
  - When you have completed all the steps, you need to hand in this paper to a teacher.
  - Do not forget to sign the attendance cheat.


Percentage score distribution by part
Filter methods 25%
Data classes 50%
Processing json-file 25%
Data classes
Processing json-file
Filter methods

1.4 Exercise

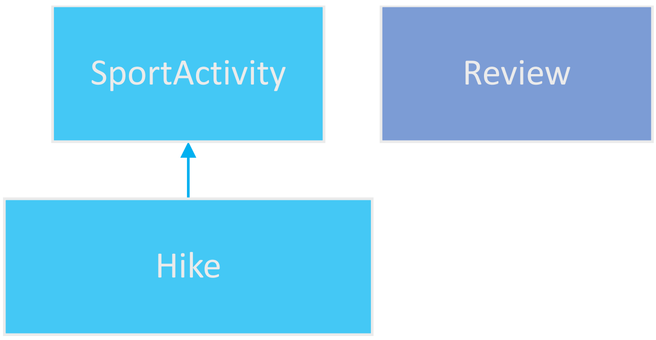
Study the json-file 'hike.json' This file contains the details of several well-known hikes in Flanders. Reviews are listed for each hike. We write a small application in Python that makes it easy to search for this information. We will write the data classes and the repository class.

Step by step: You start with the source material. Do not change the folder structure. The files test_x_x.py should not be changed and should work fine (after you have added the necessary imports).

1.4.1 A general exploration of the exercise

Create following classes:

- Class **Review** which stores the data of a review.
- Class **SportActivity** which holds some basic data about a sport activity.
- Class **Hike** which inherits from the SportActivity class.



- class **HikeRepository** which is responsible for reading the json file and can search/filter it.

Provide the **Review, SportActivity** and **Hike** classes with the correct properties (getters & setter), methods,… (see next step).

Pay attention to the inheritance between the classes.

Apply exception handling in the correct places:
- Is the property an integer or a float? Then throw an exception if the value is of the wrong data type.
- Is the property a string? Throw an exception if the value is of the wrong data type, or if the string is empty.
- Is the property a boolean or a datetime? Then throw an exception if the value is of an incorrect data type.

1.4.2 Details classes

Specific to the **Review** class:

- Provide the class with following properties (and their corresponding attributes):
    - *user, comment, ratingdate, rating*

  Some extra details:
    - *user* and *comment* are of type string.
    - *Ratingdate* is of type datetime.
      (Note: if you have problems with this, choose string as the datatype).
    - All properties have a getter and a setter.
    - The *rating* property only allows int-values between 1 and 5 (limits included).
      Add a check in the setter, and raise an appropriate error message if necessary.
- The **constructor** has 4 parameters: user, comment, ratingdate, rating
  Make sure that the rating parameter an optional parameter in the constructor. In other words, the user can choose whether or not to include this parameter. If this parameter is not included, the default value is 3.
  The ratingdate parameter is an optional parameter as well in the constructor. If this parameter is not included, the default value is today.
- Objects of the class Review are the **same** if the user and the comment are the same. Add the correct method to check this.
- Objects of the Review class can be **sorted** by their rating. The lowest rating is first. Add the correct method to do this.
- An object of the Review class can be **printed**. Output: `Review user [date]`.
- Objects of the class Review can be **displayed in a list**. Output: `Review user`
- Write an extra method *get_detail_info()*: this method returns following details of a review object as a string. Consider new lines and tabs.

```
Review from MARIE:
        - Comment: Nice walk in the forest
        - Rating: 4
        - Date: 2022-09-20 00:00:00
```

Now you can test your code via the method create_reviews() in test_without_json.py.

Specific to the **SportActivity** class:

- Provide the class with following properties (and their corresponding attributes):
    - *id, city*

  Some extra details:
    - *id* is of type int and has only a getter.
    - *City* is of type string and may not be empty.
- The **constructor** has 2 parameters: id, city.
    - The constructor creates an additional private attribute: *images*, which is an empty list by default. Create a new getter property *images* for this new attribute.
- Write a method ***add_image()***:
    - This method takes a string as a parameter.
    - If the user provides a correct parameter, you add the object to the list images.
    - If the user provides a parameter that is not of type string or an empty string, you will generate an error.
    - If the user tries to add a String object that already exists in the list, you will generate an error.
- Objects of the class SportActivity are the **same** if the id and the city are the same. Add the correct method for this.
- Objects of the SportActivity class can be **sorted** by their id. The smallest id comes first. Add the correct method to do this.
- An object of the SportActivity class can be **printed**. Output: `id [city]`.
- Objects of the class SportActivity can be **displayed in a list**. Output: `id`

Specific to the **Hike** class:

- The class **Hike** inherits from the class **SportActivity**.
    - Provide the class with following extra properties (and the corresponding attributes): *name, description, distance, difficulty, duration*

  Some extra details:
    - *name*, *description*, *difficulty* are of type string and they can't be empty.
    - distance is of type float and must be greater than 0.

    - duration is of type int and represents the walking time in minutes. This value must be greater than 0.
- Add all necessary parameters to the **constructor**.
    - The constructor creates an additional private attribute: *reviews*, which is an empty list by default. Add a getter property for this new attribute.
- Write an additional method *add_review()*:
    - This method takes an object of the Review class as a parameter.
    - If the user provides a correct parameter, you add the object to the reviews list.
    - If the user provides a parameter that is not of the Review class, you raise an error.
    - If the user tries to add a Review object that already exists in the list, you raise an error.
- Provide a **calculated property average_speed** that returns the average speed (in km/h). To do this, first convert the duration (walking time) of the hike from minutes to hours. Then divide the distance-value by the duration.
  ```
  Example: Distance: 15.0 km, Duration: 180 min → 3h, Average Speed: 15/3 = 5 km/h
  ```
- Provide a second **calculated property rating** that calculates and returns the average of the ratings of all reviews in the list. If there are no reviews yet, return the value -1.
- Write an extra method ***get_detail_info()***: this method returns following details of a review object as a string. Consider new lines and tabs.
  ```
  Hike 'Groene Gordel' in City Brussel
          - Distance: 15.0 km, Duration: 180 min, Average Speed: 5.00 km/h
          - Number of reviews: 2
          - Average rating: 3.50
  ```
- Objects of the Hike class can be **sorted** by their distance.
- The **equality** between two objects is identical as the base class.
- An object of the class Hike can be **printed**.
    - `Hike name (city) – distance km`
- Objects of the class Hike can be **displayed in a list**. Output:
    - `Hike name (city)`

At this point, you can test the remaining functions in test_without_json.

Specific to the **HikeRepository** class:
- Study the given json file (hike_EN) carefully.
- Write a public static method **load_hikes** that can read the source file and return **a list of Hike-objects with their reviews**. Don't forget to load the images as well. Include exception handling. Do a conversion from string to float/int/datetime where necessary.
    - Converting a string to a datetime can be done as follows (you can test this code in hint.py)
      ```
      from datetime import datetime
      test = "2023-01-20"
      date_time_obj = datetime.strptime(test, '%Y-%m-%d')
      ```
- A public static method ***search_in_hikes*** with 2 parameters: a list of objects from the class Hike and a search value. This method returns a list of those Hike objects where the search term appears in the *city* or *name* of the Hike. The user should not worry about case sensitivity.

- A public static method ***sort_hikes_by_difficulty*** with 1 parameter*:* a list of objects from the class Hike. This method returns a dictionary. For each item in the dictionary
    - The key: the difficulty
    - The value: a list of hikes that have that level of difficulty.

- A public static method ***filter_hikes_by_difficulty_duration*** with 3 parameters*:* a dictionary of objects of the class Hike grouped by difficulty (see previous method for structure), a requested difficulty and a maximum duration in minutes. This method returns a list of objects of the Hike class. This list will contain only those hikes where
    - the difficulty is equal to the requested difficulty,
    - the duration is less than or equal to the duration passed.

Use the test file 'test_with_json.py' to test the various functionalities in detail. Afterwards, also test your code with the other json source file.

Good luck!

Output of the test classes:

test_without_json.py

```
*** 1a) creating a review
Review from STIJN [2024-01-15 11:42:10.487962]
*** 1b) printing details of a review
Review from STIJN:
        - Comment: Nice walk in the forest
        - Rating: 3
        - Date: 2024-01-15 11:42:10.487962
*** 1c) testing equality between two reviews
Review2 and review3 are equal
** 1d) Sorting and printing of a list
Not sorted: [Review STIJN, Review MARIE]
Sorted on rating: [Review MARIE, Review STIJN]

*** Test 2 ***
*** 2a) creating a hike
Hike Discover Howest (Kortrijk) - 12.5 km
Hike 'Discover Howest' in City Kortrijk
        - Distance: 12.5 km, Duration: 120 min, Average Speed: 6.25 km/h
        - Number of reviews: 0
        - Average rating: -1.00
** 2b) Sorting and printing of a list
Not sorted: [Hike Discover Howest (Kortrijk), Hike Explore city Gent (Gent)]
Sorted on rating: [Hike Explore city Gent (Gent), Hike Discover Howest (Kortrijk)]

*** Test 3 ***
*** 3a) creating a hike with reviews
Hike 'Discover Howest' in City Kortrijk
        - Distance: 12.5 km, Duration: 120 min, Average Speed: 6.25 km/h
        - Number of reviews: 2
        - Average rating: 2.50
*** 3b) printing reviews of a hike
Review from STIJN:
        - Comment: Boring walk
        - Rating: 1
        - Date: 2022-11-10 00:00:00
Review from MARIE:
        - Comment: Nice walk in the forest
        - Rating: 4
        - Date: 2022-09-20 00:00:00

*** Test 4 - Wrong input ***
Error: Incorrect value rating
Error: Error on adding review
```

test_with_json.py

```
Number of walks: 12
[Hike Castles Route (Gavere), Hike Historical City Walk (Ghent), Hike Nature Park Brielmeersen (Deinze), Hike
City Walk (Maldegem), Hike Old bare valley (Vinderhoute), Hike Roger Raveel Walking Route (Machelen-aan-de-Leie),
Hike Green Belt (Brussels), Hike Dunes Walk (De Panne), Hike Sonian Forest (Tervuren), Hike Coastal Path
(Ostend), Hike Heuvelland (Ypres), Hike Kalmthoutse Heide (Kalmthout)]

****2) sorting hikes + print detail info****
Hike 'Roger Raveel Walking Route' in City Machelen-aan-de-Leie
        - Distance: 3.7 km, Duration: 45 min, Average Speed: 4.93 km/h
        - Number of reviews: 2
        - Average rating: 4.50
Hike 'Historical City Walk' in City Ghent
        - Distance: 5.0 km, Duration: 90 min, Average Speed: 3.33 km/h
        - Number of reviews: 0
        - Average rating: -1.00
Hike 'City Walk' in City Maldegem
        - Distance: 5.0 km, Duration: 90 min, Average Speed: 3.33 km/h
        - Number of reviews: 1
        - Average rating: 5.00
Hike 'Old bare valley' in City Vinderhoute
        - Distance: 8.0 km, Duration: 100 min, Average Speed: 4.80 km/h
        - Number of reviews: 4
        - Average rating: 3.00
Hike 'Castles Route' in City Gavere
        - Distance: 10.0 km, Duration: 120 min, Average Speed: 5.00 km/h
        - Number of reviews: 2
        - Average rating: 3.00
Hike 'Kalmthoutse Heide' in City Kalmthout
        - Distance: 10.0 km, Duration: 120 min, Average Speed: 5.00 km/h
        - Number of reviews: 2
        - Average rating: 3.50
Hike 'Dunes Walk' in City De Panne
        - Distance: 10.0 km, Duration: 120 min, Average Speed: 5.00 km/h
        - Number of reviews: 2
        - Average rating: 4.50
Hike 'Sonian Forest' in City Tervuren
        - Distance: 12.0 km, Duration: 150 min, Average Speed: 4.80 km/h
        - Number of reviews: 0
        - Average rating: -1.00
Hike 'Green Belt' in City Brussels
        - Distance: 15.0 km, Duration: 180 min, Average Speed: 5.00 km/h
        - Number of reviews: 2
        - Average rating: 3.50
Hike 'Heuvelland' in City Ypres
        - Distance: 15.0 km, Duration: 120 min, Average Speed: 7.50 km/h
        - Number of reviews: 2
        - Average rating: 4.50
Hike 'Coastal Path' in City Ostend
        - Distance: 20.0 km, Duration: 180 min, Average Speed: 6.67 km/h
        - Number of reviews: 2
        - Average rating: 4.50
Hike 'Nature Park Brielmeersen' in City Deinze
        - Distance: 20.0 km, Duration: 180 min, Average Speed: 6.67 km/h
        - Number of reviews: 3
        - Average rating: 5.00

****3) filter hikes ****
Enter a (part of a) city or name of the walk:> heuvel
These are the search results:
Hike 'Heuvelland' in City Ypres
        - Distance: 15.0 km, Duration: 120 min, Average Speed: 7.50 km/h
        - Number of reviews: 2
        - Average rating: 4.50

****4) sort hikes on  difficulty****
* Difficulty Easy:
[Hike Roger Raveel Walking Route (Machelen-aan-de-Leie), Hike City Walk (Maldegem), Hike Castles Route (Gavere)]
* Difficulty Average:
[Hike Historical City Walk (Ghent), Hike Kalmthoutse Heide (Kalmthout), Hike Dunes Walk (De Panne), Hike Sonian
Forest (Tervuren), Hike Green Belt (Brussels), Hike Coastal Path (Ostend), Hike Nature Park Brielmeersen
(Deinze)]
* Difficulty Difficult:
[Hike Old bare valley (Vinderhoute), Hike Heuvelland (Ypres)]

****5) filter hikes on difficulty and duration****
Enter a maximum duration:> 100
Enter a difficulty:> Difficult
These are the search results:
Hike 'Old bare valley' in City Vinderhoute
        - Distance: 8.0 km, Duration: 100 min, Average Speed: 4.80 km/h
        - Number of reviews: 4
        - Average rating: 3.00
```