INSTRUCTIONS TO INSTALL AND RUN THE SIMULATION of the PARTAKER-SHARER ADVISING FRAMEWORK

- Simulation implemented in Python:
  - ➔ we recommend installing 'python 3.10' or more recent versions (https://www.python.org/downloads/)

- Required PACKAGES (to be installed using the 'pip install <package_name>' command from command line (after having installed Python and pip):
  - numpy (to download and install it: 'pip3 install numpy', from command line)
  - matplotlib (follow same procedure to install)
  - pickle
  - time
  - os
  - random
  - math

In the project folder there are 2 sub-folders containing Python files:
- 'one_agent' --> contains code for the single agent/predator model
- 'two_agent' --> contains code for the model with 2 predators/agents

---------- **STRUCTURE of the 'two_agent' sub-folder** ----------

Inside the 'two_agent' sub-folder (same structure for the 'one_agent' sub-folder):

➔ there is 1 Python file for each type of environment:
- 'environment_comm.py' defines the "environment" and "agent" Python classes for the case with a FIXED prey and partaker-sharer communication
- 'environment_comm_prey.py' defines the "environment" and "agent" Python classes for the case with a MOVING prey and partaker-sharer communication
- 'environment_no_comm'.py defines the "environment" and "agent" Python classes for the case with a FIXED prey and NO communication
- 'environment_no_comm_prey'.py defines the "environment" and "agent" Python classes for the case with a MOVING prey and NO communication

➔ 'utils.py' contains the definitions of the main functions used in the 'main.py' program: functions and parameters are described as comments

➔ 'main.py' contains the main program to RUN (it just calls the functions of 'utils.py')

---------- **RUN SIMULATION** ----------

TO RUN A SIMULATION JUST run the 'main.py' (which imports the environments and 'utils.py', so be careful to have everything on the same folder):

- in the 'main.py' define the grid size of interest and the training parameters ("n_episodes" and "max_steps", already setted)
- then create the environment of interest by calling the corresponding class constructor (just un-comment one of the 4 possible environments, we already un-comment the most interesting one)
- select the name of the chosen model (un-comment the desired one, we already un-comment the most interesting one)
- To see/run the simulation, un-comment the function 'utl.run_training_simulation(...)' (already un-commented) which will run a full training of n_episodes on the chosen model visualizing the grid world environment with moving agents and prey on the standard output (console/terminal)

- All the other functions are left commented (except for some plotting ones), since useful to perform plots and comparisons (used for performing the tests)
- We leave uncommented the functions performing the plots so that, after 'utl.run_training_simulation(...)' is finished, the resulting plots are saved into the program folder.
- Check plots to visualize the learning process (Time To Goal and Budget usage)

IMPORTANT NOTE!

- ➔ The program is ready to run a simulation of a PSAF model with 2 agents and moving prey on a grid word of size 10.
- ➔ Simply run the 'main.py' file and look at the standard output to visualize the simulation (be careful to be inside the folder containing the environments and the 'utils.py')
- ➔ Modify the main only for making other experiments

Notice that all functions and procedures are commented and described in detail inside the scripts,

- ➔ -in case of problems and for further details, please contact at: "denaldo98@gmail.com" or "denaldo.lapi@mail.polimi.it" or "hailey.shakespear@estudiantat.upc.edu"

- ➔ -The code is available also on GitHub on the following repository: "https://github.com/denaldo98/SOAS_project"