“Spy-School” Assignment

This assignment is aimed at assessing students’ skills and knowledge in Python programming and software development and implementation. Occupies 50% of the total grade of the course Intelligent Control, minimum passing grade 5.5.

There are in total 2 tracks available to be selected, each student group is supposed to **choose only 1 track** to complete.

Deliverables:

- a report with code as appendix,
- original .py file.

General requirements:

1. Give proper reference with consistent referencing format in both the report and the code. Particularly, using others’ code without giving proper reference would result in plagiarism. In that case, the student is instantly failing this part of the course and the action of plagiarism will be reported.
2. Please include your name, student numbers, date in your code as comment text in the first line of your script.
3. Please do proper packaging of your code. Make sure one-line of command execution of your main script file. You can include multiple python scripts in your submission, but it is required to only run one file by *“python3 your_file_name.py”*. Self-defined arguments are allowed but requires proper documentation in your script and your report.
4. Making your python file executable is not recommended, this could impose security risk for your operating system.
5. If any major bug discovered by students and could not be solved by the time of submission, please do not worry. But please provide proper documentation.
6. Include a readme file in your submission for instruction manual of your code. Should be short and to the point.
7. After the submission, plagiarism-prevention interviews will be randomly held for each group or student. Students are supposed to explain everything he/she submitted (action-wise but not knowledge-wise). Which means no points will be deducted if the interviewed students provide false answer in the perspective of knowledge taught. Students only need to justify the action and show he/she know everything submitted.
8. Grades will be announced in the form of one virtual group meeting with the lecturer. Students can object to grades given by the end of the meeting. The grade distribution among the group members will be determined by the group members themselves with the possibility of taking the plagiarism-prevention interviews as a reference (only in case of suspected plagiarism activity).
9. Source of reference should only be trustworthy source. Ideally, peer-reviewed.

Requirement for report:

1. Use proper, meaningful report structure. You have the freedom to pick what chapters to include, but please follow a meaningful and logical order. Make sure your report structure can properly present your work.
2. Provide justification for all your statements.
3. If pictures, diagrams, tables, etc. are used, make sure they are of high quality and readable.
4. To the point. Text like *"In the modern world, technology is evolving fast. We are at the crossroads of……."* are not appreciated.
5. Necessary points to explain in the report:
    a. Cover page, title, student numbers, date, page numbers, table of contents
    b. Necessary background information
    c. The problem formulation, goal of your software
    d. How you came to your solution
    e. Specific design of your solution, make sure to document carefully if extra rules or assumptions made in your report. Specify the inputs and outputs, use visual tools/diagrams to demonstrate if necessary.
    f. How did you implement your solution
    g. Pros and cons of your solution
    h. Performance of your solution
    i. Reflection, possible improvements, etc.
    j. Reference
    k. Code in appendix
6. Main body of the report is no shorter than 3 pages but should not exceed ~21 pages in principle. Use professional formatting, do not compress the font size or paragraph spacing too much in order to fit in the page limit.

    If you really have difficulties meeting the page limit, either put things in the appendix without sabotaging the story telling in the main body or you can compress the format a little bit. The bottom line of the format compression is the format of this document. This document is using Calibri(Body), font size 11, multiple line spacing *1.08, 8 pt spacing after each paragraph. Based on the bottom-line of this most compact format, students have the freedom to select the formatting.

Quality of code:

1. Readable, with consistent style, easy to understand
2. Well-documented
3. Reusable
4. Can be tested
5. Secure to use, with essential input/output control and error tolerance (rules upon dealing with errors can be self-defined but please document this in the report)
6. Provide change logs and version control if necessary. (optional)

Assessment:

1. Report (~45%)
2. Code(~45%)
3. A simple defence session will be held prior to the grade announcement session.(~10%)

Track 1 – *"The Imitation Game"*

(9 points) Create a software tool for Caesar ciphers using Python that have 3 functions:

- Encrypt with a given key k
- Decrypt with a given key k
- Find the key k

We are only considering the English language. Thus, the alphabet space is 26 English letters. The runtime of each test case should be below 4.5 seconds on my laptop. (My laptop: 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz; 16GB RAM; x64 Windows 10 21H2)

BONUS(2 points): try to do the same thing for the substitution cipher. The mapping rule for the substitution cipher can be self-define, you are also allowed to shrink the alphabet space.

Tips: present your thoughts, research, efforts, validation and justification helps you get better grades.

Grading rubric: see Appendix A in a separate document.

Track 2 – *"The experienced tracker"*

(9.5 points)In this assignment you act as a stalker that performs stalking mission on one HVT(high-value target). The stalking lasts 90 seconds. To simplify this process, we set the basic time interval as 1 second. So there are 91 time instances, starts from 0 ends at 90 second mark. The unit of the distance is omitted for the convenience of this assignment. Below are some essential parameters:

- Stalker velocity: can be any integer in 1,2,3,4,5, you are controlling your own speed. But to avoid being suspected by the HVT, you are not allowed to stop. You can change your velocity at precisely every time instance and keep it for at least 1 second.
- HVT velocity: follows a gaussian distribution with mean v_{avg} and standard deviation of 1. But the HVT is very experienced and a bit tricky to handle, he can sometimes, step back 1 unit distance with probability p. The v_{avg} and p values are different for each group.
- Initial distance between the stalker and the HVT: 50
- Desired distance between the stalker and the HVT: 27 +/- 3
- The HVT starts statically, he starts moving at precisely the 5 second mark.

The speed of HVT is provided as a sequence in a text file with 91 values by lecturer so it's not mandatory to write script to generate this sequence. 4 training sequences will be provided to each group.

If for 3 consecutive seconds your distance to the HVT is less than but not equal to 27, or for one time instance the distance is smaller than but not equal to 20 the HVT can feel you and the stalking mission fails. You should stop and the program should end with appropriate message of your choice.

If for 3 consecutive seconds your distance to the HVT is more than or equal to 35 or for one time instance the distance is more than or equal to 41, you lost the HVT and the stalking mission fails. You should stop and the program should end with appropriate message of your choice.

(Grading rubric seen Appendix B in a separate document)

Finish the mission, good luck agent.

Additional requirements for Track 2:

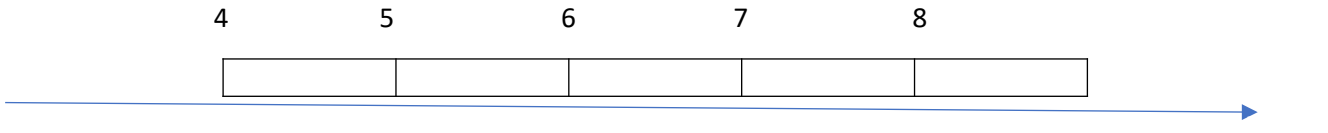
1. Try build your own algorithm for controlling the tracking mission.
2. Log and plot the distance between the agent and the HVT at least during validation of your final script.
3. Explicitly explain your algorithm. If you can, try to use pseudo code. Otherwise, use whatever you like, make sure demonstrate clearly.
4. Test the performance of your program. Think about parameters to represent the performance of your program in terms of efficiency (speed of your program), stability, error tolerance, etc.(you are free to evaluate other performance aspects) Provide at least one parameter for each aspect. In total 6 parameters needed to be reported. Two examples: total runtime of your program, mission success rate. You can include this example as two of your 6 parameters. They're free 😉
5. Do not simulate the process real time! (Which means your script should never be running 90 seconds!!)
6. Although the entire sequence is provided, do not adjust the agent speed based on future speed data of the HVT. If this is discovered in your code, it's cheating, and you fail.

Bouns(1 point): try to use Python to automate the recording of your parameters, save the raw data and the parameters in files, csv or txt. Try to plot the graphs using Matplotlib module.

(On the next page)

Some additional explanation on the data sequence: (the velocity sequences here are only examples)

v_t_n is the velocity of the HVT at time instance n, this velocity marks the average speed of the HVT over the past second(from time instance n-1 to n). v_s_n is the velocity you have selected or will be selecting for the agent in the next second(from time instance n to n+1).



HVT velocity sequence	v_t_4 = 0	v_t_5 = 0	v_t_6 = 3	v_t_7 = 5	v_t_8 = -1
Distance between stalker and HVT	30	25	26	29	19
Stalker velocity sequence	v_s_4 = 5	v_s_5 = 2	v_s_6 = 3	v_s_7 = 5	v_s_8 = 1