

I. Assessment Requirements

You are expected to submit a collection of source codes, documentations and test results related to the design of a chatbot. The chatbot must be modular and demonstrate the use of several major AI techniques, as well as their integration into one user friendly system with a common topic.

Tasks
Tasks a and b: Chatbot with rule-based and similarity-based conversation and logical reasoning components

Chatbot topic: Make sure that the chatbot topic, as well as the individual functionalities, are reasonably specific, so it's unlikely someone else in the class creates very similar functionalities.

Deliverables:

- **Source codes and supplementary files:** The source codes are to be programmed in Python, and submitted in .py or .ipynb. For multiple source code files, and where you have any supplementary files (e.g., AIML and CSV) you may compress and submit them together with your source codes as a single .zip file. Add in-line comments and indent your source code for maximum readability. You can use other Python packages if they are on pypi.org and you find them useful for your chosen domain of application, but you must not use them as a replacement for any of the specified parts of the assignment (for example, you must use AIML, not any other method to program chatbot responses).
- **Documentation:** The submission of each part of the coursework must have a documentation file included. This must be submitted in single Word or pdf file using the **provided template**. This document should include:
 - (1)**Design notes:** The general explanations of the system and its goals, the system requirements, i.e., the list of what the system should do/have from a user's perspective, explaining the employed AI techniques, and the explanation of your programs, i.e., details of what parts of your program do what. You do not need to do a research or reference your work. No word count, but the details should be reasonably enough for a reader to understand your design.
 - (2)**Conversation log:** The submission of each part of the coursework must have a conversation log (showing at least 20 conversation pairs for each stage). For each conversation log, record an actual conversation between you and the chatbot that demonstrates the implemented features. Where appropriate, annotate it with brief comments that explain which feature/component generated this, and how, for any particularly remarkable output. If the conversation included non-textual input (e.g., images), this should also be included to the degree possible.

II. Assessment Scenario/Problem

- **Task a - Rule-based and similarity-based conversation features:** This submission should consist at least one Python file that implements the chat bot, one AIML file that implements the rules, and one file of pre-defined Q/A pairs. For the rule-based conversation, you may use the Python and AIML files provided on NOW as a starting point, but you should extend and customise them towards your design specification. The similarity-based conversation should be added based on the bag-of-words model, tf/idf, and cosine similarity. If the user inputs a sentence than is matched to an AIML entry, the answer is provided accordingly. Otherwise, the QA pairs should be searched for the closest match, and the relevant answer is to be returned to the user. See the notes on week 7 lab sheet for more information.
- **Task b - Add logical reasoning extension:** The aim of this task is to build a simplistic first order logic (FOL) knowledgebase and inference engine using NLTK library, that can be updated or queried by the user. This component could ideally be implemented using full grammatical analysis and NLP of the user input, but this is out of the scope of this coursework stage. Therefore, the user inputs for this component are limited to two simple patterns of: "I know that ... is ..." and "Check that ... is ..." or similar. For example, user types "I know that Tim is British" or "Check that Tim is European". You must make a KB file with a number of initial statements (at least 10) about your chosen chatbot topic. Each fact is written in first-order logic structured in the NLTK's FOL syntax, such as "British (Tim)" and "British(x) -> European(x)". In your program, first import the initial knowledgebase file and check it for any contradiction. Then, if the user inputs "I know that ... is ...", first check if the new expression is not in contradiction with the knowledgebase. If not, add it to the knowledgebase (in memory not in the file) and respond like "OK, I will remember that ... is ...". If the user inputs "Check that ... is ..." then you respond with "Correct", "Incorrect" or "Sorry, I don't know" by applying the NLTK's resolution algorithm. See the notes on week 9 lab sheet for more information.

Submission Stage-1 - Includes tasks a and b.

Files to submit: A zip file containing Python program (py or ipynb), AIML file (xml), Q/A pairs (csv), KB file (csv). Separately submitted documentation (doc, docx or pdf) that also includes demo video URL.

What is an “Extra Functionality”?

The extra functionalities for each task, is implementing AI techniques related to the basic task, so that it shows you have explored some new techniques on your own, and that you showed how to program them in order to extend the minimum required functionality. Remember that the extra works are considered only if all the basic tasks are implemented successfully. Of course, there is no definite list of the extra functionalities you may go for, however here are some suggestions:

Basic requirements	Extra functionalities: Think of...
Task a: A chatbot that uses AIML for pattern-based conversation and tf/idf for similarity matching.	<ul style="list-style-type: none"> - Other methods for communication rather than text-based. - Other NLP techniques that makes your chatbot smarter in answering questions - Other web services (than what given) that you may use to extend the chatbot answering capabilities - etc.
Task b: Adding logical reasoning based on FOL knowledgebase, NLTK library and resolution inference.	<ul style="list-style-type: none"> - Other libraries or reasoning techniques. - Other logics than FOL, e.g., fuzzy or multi-valued logics - Other NLP techniques to support extended forms of logical conversations than the basic ones, e.g., support for multi-valued predicates, or designing a logical game. - etc.