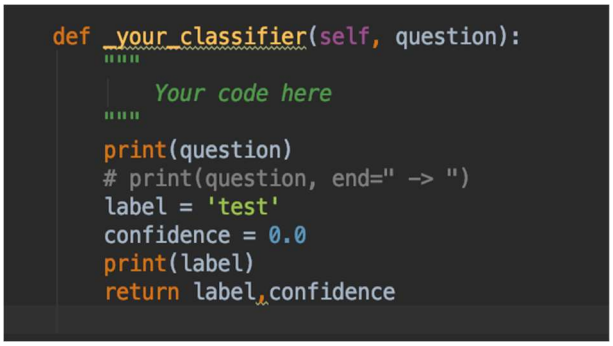
CS4635-01, CS7637-01, CS4635-A, CS7637-A Project 3

This class has three projects which walk you through building a simple Jill Watson. Jill Watson is a virtual teaching assistant with the goal of answering syllabus questions (this semester) in a virtual classroom. Jill Watson is trained as opposed to coded on the course specific material; for project 3 you are to create a classifier of your own to be trained to classify questions about a course syllabus.

The Jill Watson runtime supports multiple classifier plug-ins, most of which are ML but this project will concentrate on an AI based classifier. The file TestAdapter.py is a plug-in wrapper that we use when we want to test new classifiers in the Jill Watson framework. For this project, you will develop your code in TestAdapter.py so that later we can potentially test your training algorithm and the resulting classifier alongside other classifiers as part of our research.

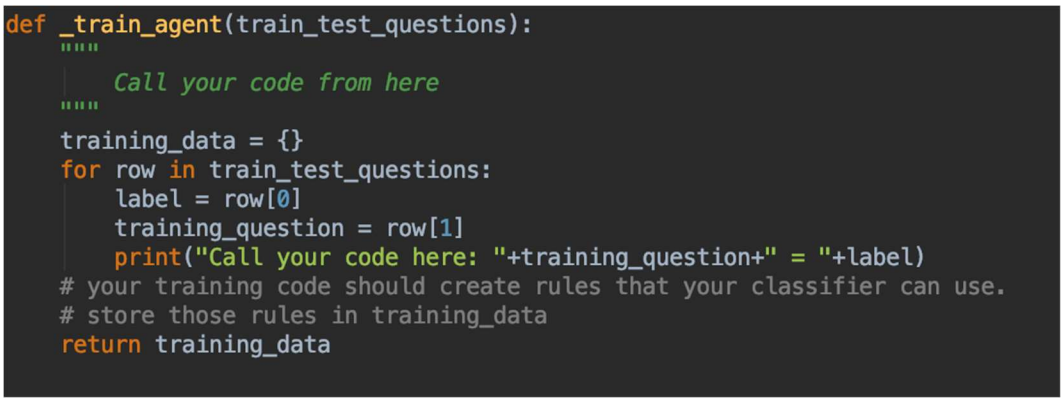
You are given a set of labeled questions of which your agent must learn patterns (supervised learning); plus a set of questions to test your trained agent. After you submit you agent, it will be trained on a **completely different set of training questions and tested with a different set of test questions**.

There are two files to add your solutions to: TestAdapter.py and Project3.py. There are comments that indicate where you should be able to start your work. (Please look at the screenshots below).



```
def _your_classifier(self, question):
    """
        Your code here
    """
    print(question)
    # print(question, end=" -> ")
    label = 'test'
    confidence = 0.0
    print(label)
    return label,confidence
```

(TestAdapter.py)



```
def _train_agent(train_test_questions):
    """
        Call your code from here
    """
    training_data = {}
    for row in train_test_questions:
        label = row[0]
        training_question = row[1]
        print("Call your code here: "+training_question+" = "+label)
    # your training code should create rules that your classifier can use.
    # store those rules in training_data
    return training_data
```

(Project3.py)

Steps

1) Download the project zip file (which contains project3.py, TestAdapter.py, and training_test_data.txt) from Canvas.
2) Call your training algorithm from project3._train_agent()
3) Call your trained classifier from TestAdapter._your_classifier()
4) Test your code using the command: **python project3.py training_test_data.txt**
5) Check results
6) When you are happy with your results, zip up all the files used by your project into a file using the following template: project3_<your name>.zip

Constraints

- Code must work with python 3.7.x
- No additional libraries are allowed or needed
    - Only libraries included with Python are allowed
- The vocabulary is limited to the words (and their **base (hint)**) in the training questions
- The questions will not contain commas

Your Agent Needs to

- Learn a relation between training questions within a label and possibly difference between labels.
    - Note: your agent is learning
- Classify a test question to the appropriate label.
- The returned label must be one of the labels from the training question/labels pairs.
- The labels, training questions, and test questions will be completely different when we grade your agent.
    - The relationships between training questions may be different when grading.
    - The vocabulary used when grading will be different.
    - The training questions and test questions used when grading will use proper grammar and be similar in complexity to the training/test questions provided as part of the project.

Report Rubric (50% of grade) – Full description on Canvas

Item #	Points	Address in your Report
1	10	Draw a block diagram of your agent.
2	10	Explain each block in your diagram.
3	10	Explain how your design uses the KBAI concepts taught in class
4	10	How does your agent's design relate to human cognition?
5	10	What are the strengths and weaknesses of your agent?

Agent Grading

Your agent will be executed using a training set and training questions you have never seen. The unseen questions have been chosen to evaluate your agent in a fair manner; we trained a Watson model to filter out any questions that are too tough, and trained an agent written by an instructor to make sure the questions were reasonable.

File Structure

File	Modify
project3.py	Contains the code to train the agent and execute the TestAdapter
TestAdapter.py	Contains the code to test the agent
train_test_questions.csv	Contains the test and training questions as well as their labels.