Welcome to CSE 355!

My name is Tanay 😁

My recitation timings:

- Monday 3:00 to 3:50 PM
- Friday 12:20 to 1:10 PM

Classes I'm taking right now:

- CSE 434 : Computer Networks
- CSE 445 : Distributed Software Development
- CSE 466 : System Security
- CSE 475 : Foundations of Machine Learning
- CSE 485 : Capstone
- CSE 463 : Human-Computer Interaction

What do we do in Recitations?

- Usually I will have an agenda for the day where I go through concepts discussed in the classes or things that will appear in the future modules
- In the remaining time, I can take up examples/problems or any questions that you guys may have

Are recitations mandatory?

- NO! Recitations are completely optional and we do not record attendance. They are however, still an important resource and is highly recommended and encouraged.

Today’s Agenda

- Installing python
- Installing package (cse355-machine-design)
- Demo program using above-installed package

If time remains, I can go over your problems/questions

Installing Python on Mac

1. Open terminal and run “python3 –version” or “python –version” to check if the installed version is >3.10. If it is already installed, you may skip the installation.
2. Go to https://www.python.org/downloads/ and click on Download python 3.12.5 for Mac. You can choose your version of choice by scrolling down but it should >3.10
3. After the download has completed, run the installer. You should see it in Downloads folder or in your browser’s download section. Please ensure that you checked the option to add python to PATH variable.

Installing Python on Windows

1. Open CMD and run “python3 –version” or “python –version” to check if the installed version is >3.10. If it is already installed, you may skip the installation.
2. Go to https://www.python.org/downloads/ and click on Download python 3.12.5 for Windows. You can choose your version of choice by scrolling down but it should >3.10
3. After the download has completed, run the installer. You should see it in Downloads folder or in your browser’s download section. Please ensure that you checked the option to add python to PATH variable.

Something went wrong?

Try running “python –version” or “python3 –version” to check if the installed version is >3.10

It’s okay if you didn’t get it right the first time. Try again by following the instructions in this video : https://www.youtube.com/watch?v=YYXdXT2l-Gg and make sure that the instructions match with your operating system.

Feel free to ask any questions about the installation steps.

Installing cse355-machine-design

Depending on the command that you used to check the version earlier, use the command “pip install cse355-machine-design” for python or “pip3 install cse-355-machine-design” for python3 in your machine’s terminal or CMD.

- Mac : “pip3 install cse-355-machine-design”
- Windows : “pip3 install cse-355-machine-design”

Try using pip instead of pip3, if it doesn’t work with the latter.

Problem Demo with cse355-machine-design

Let’s look at a demo where we use the installed package & python to create a DFA (Deterministic Finite Automata) that recognizes the language L.

$L = \{ w \in \{a,b\} \mid w \text{ starts with aba} \}$

WHAT’S DFA OR LANGUAGE?

- Don’t stress about the details for now. I’ll walk you through it as we code.
- For now, know that DFAs are machines that recognize a language & a language is basically the strings that are accepted by the DFA.

DFA (Deterministic Finite Automata)

- DFA in a nutshell is like a machine represented as a graph (the data structure) that has nodes and edges. Every node represents a state that the machine is in and goes to another state when a character is consumed at an edge.
- You define a DFA using
  - Q : Set of states (set of nodes)
  - ∑ : Alphabet set (set of acceptable alphabets)
  - δ : Transition Function (defines relationships between states using the input consumed)
  - q_0 : Start state (starting node - head)
  - F : Final states set (set of states that accept a string)

DFA (Deterministic Finite Automata)

- It’s okay if you find this overwhelming right now, what’s important is you know that you need 5 things to define a DFA - (Q, ∑, δ, q_0, F).
- Do not hesitate to ask questions if something is unclear but know that this is yet to be covered in class.

Back to the problem

For the language L = { w ∈ {a,b} | w starts with aba }, the DFA will be defined as

Q = {q_0, q_1, q_2, q_3, q_4, q_5}

∑ = {a, b}

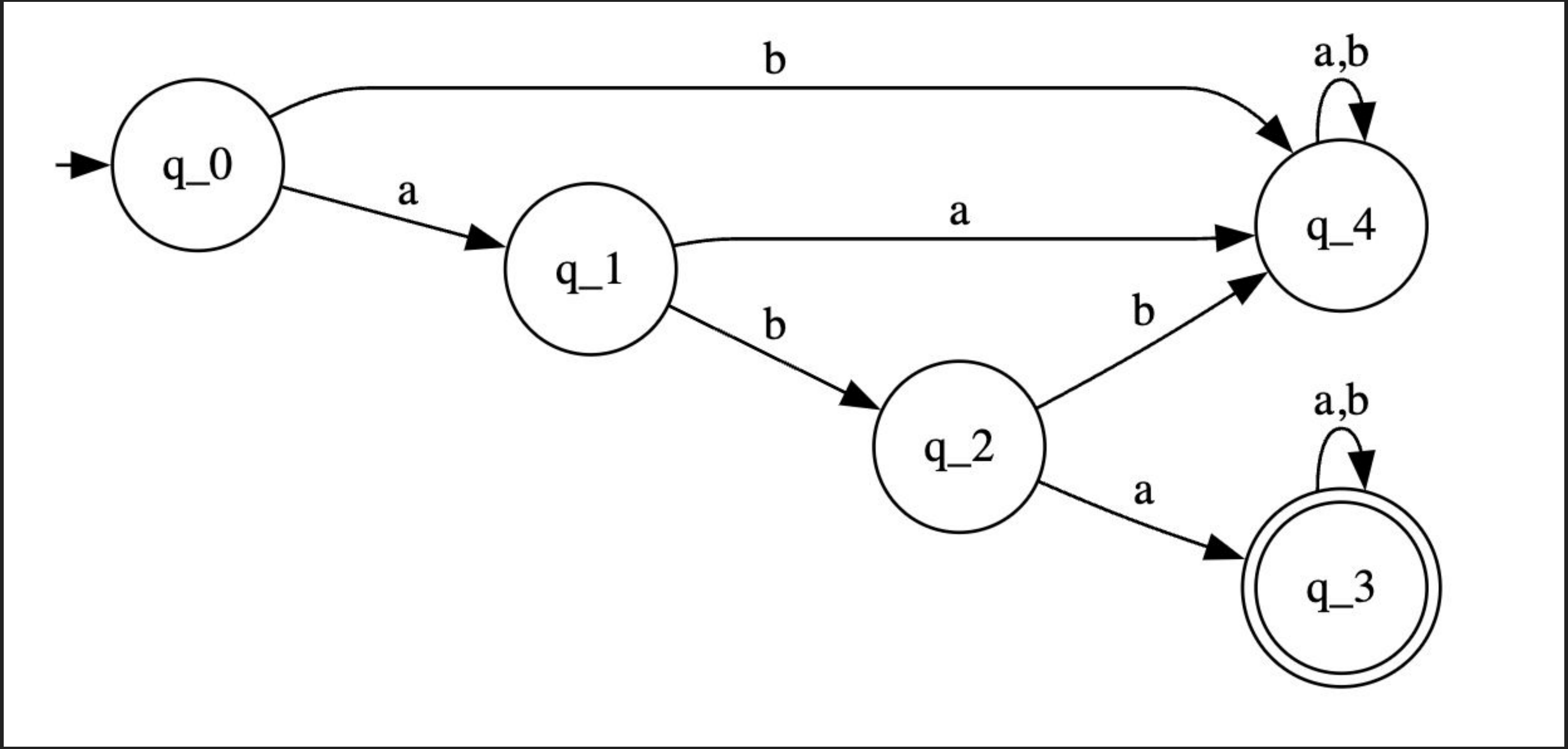
δ =

	q_0	q_1	q_2	q_3	q_4
a	q_1	q_4	q_3	q_3	q_4
b	q_4	q_2	q_4	q_3	q_4

q_0 = q_0

F = {q_3}

DFA as a graph (try “ababa” and “aab”)



Let’s construct this in Python

Open your favorite IDE and follow along as we reconstruct the DFA in python using the machine design package we installed.

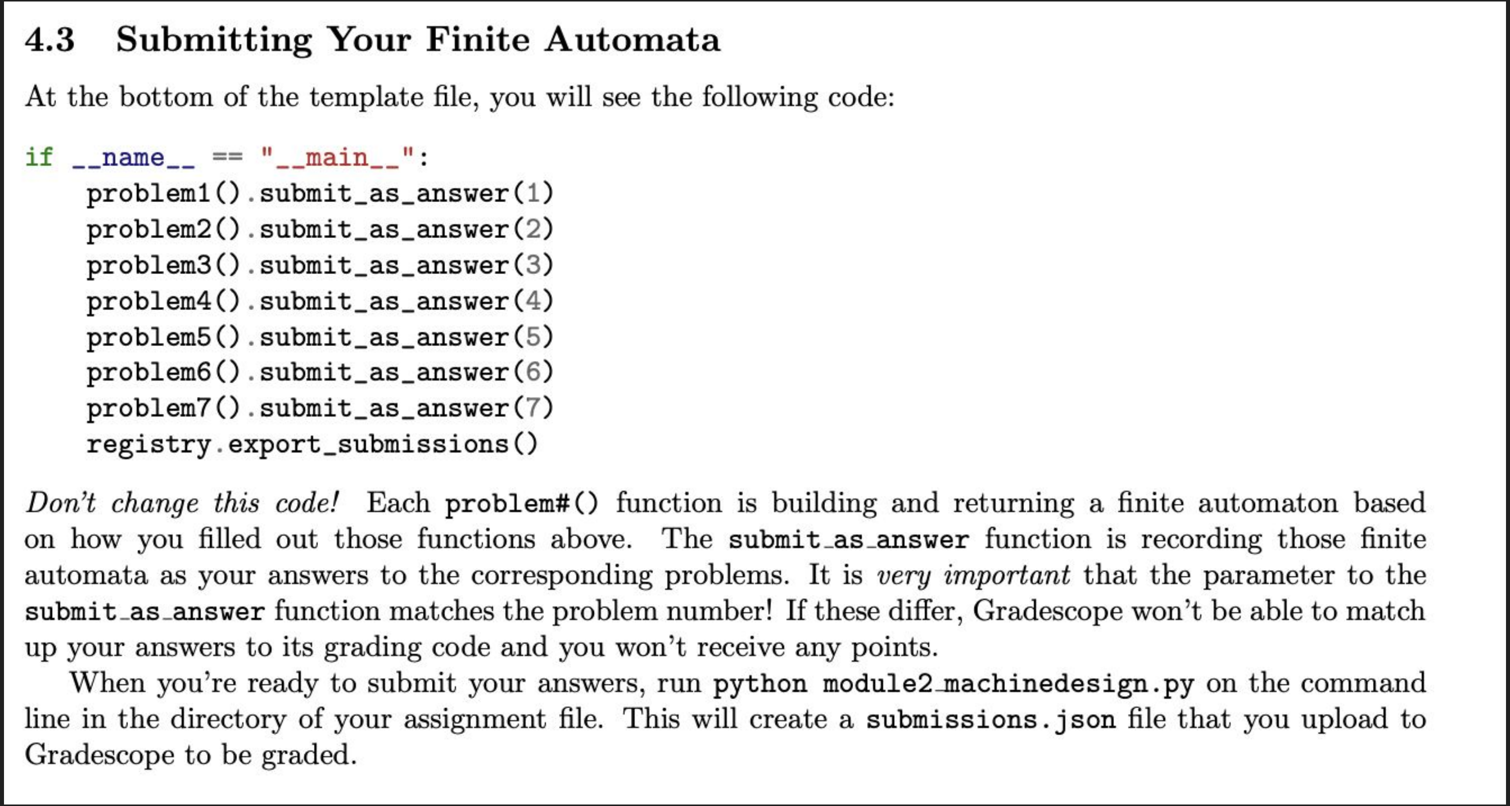
1. Import necessary modules
    a. In our case, “from cse355-machine-design import DFA, registry”
2. Create a function that returns a DFA “object”
3. Run operations on your DFA like display_state_diagram(), and evaluate(string) with enable_trace.

Congratulations!

You just created a DFA! If you had any problems going through the demo. You may look at the slides again for reference or contact me for help.

You may also look at this github repo for solution and expected result : https://github.com/tanay-jaiman/CSE355_Recitations/tree/main/Module1

How to submit DFAs in assignments?



4.3 Submitting Your Finite Automata

At the bottom of the template file, you will see the following code:

```
if __name__ == "__main__":
    problem1().submit_as_answer(1)
    problem2().submit_as_answer(2)
    problem3().submit_as_answer(3)
    problem4().submit_as_answer(4)
    problem5().submit_as_answer(5)
    problem6().submit_as_answer(6)
    problem7().submit_as_answer(7)
    registry.export_submissions()
```

Don't change this code! Each **`problem#()`** function is building and returning a finite automaton based on how you filled out those functions above. The **`submit_as_answer`** function is recording those finite automata as your answers to the corresponding problems. It is *very important* that the parameter to the **`submit_as_answer`** function matches the problem number! If these differ, Gradescope won't be able to match up your answers to its grading code and you won't receive any points.

When you're ready to submit your answers, run **`python module2_machinedesign.py`** on the command line in the directory of your assignment file. This will create a **`submissions.json`** file that you upload to Gradescope to be graded.

Any questions, concerns, queries?