Fusemachines Research and Training Center

Syllabus

Basics of Computer Science for Artificial Intelligence

Foundations in Artificial Intelligence Program

Version	Significant Changes (Marked with a Symbol)	Modified by	Modification Date
1.0	First Version of the course	Fuse.ai Team	Nov 2019
2.0	2.0 Changes marked as Green	Fuse.ai Team	

Contents

Introduction

All the syllabuses are reviewed regularly so that it will be able to reflect the latest thinking and current best practices employed in industries and take into account different national and international contexts in which these courses will be taught.

Syllabus Aims

The Syllabus aims to:

- Provide a worthwhile learning experience for all learners and enable them to acquire sufficient Computer science knowledge and skills to get started in the domain of AI
- Facilitate and Standardise Course Content Development and Delivery

Introduction to the Course

This course provides the fundamental concepts of computer science required to master the field of Machine Learning. Our course starts with the basic introduction of artificial intelligence and machine learning, it's history and applications. Later, you will learn about basic undergraduate computer science, programming in python, data structure and database. You will also study software development life cycle, agile methodology, web frameworks, rest api and git. Thus, after studying this course, you will gain both theoretical and hands-on knowledge about the tools used in software engineering in industry today.

After completing this course, students will be able to

- Explain the concepts of AI and its applications
- Understand the differences between AI, ML and data science.
- Understand how data is represented and stored in computer; learn about processors and how computers communicate with each others.
- Use Linux operating system, understand how file system is organized in it and concepts of virtual machines.
- Build programs in python and and use powerful libraries such as numpy, pandas, matplotlib to perform numerical calculations, store data and visualization.
- Compare and contrast different sorting algorithms such as bubble sort, insertion sort and merge sort.
- Compare greedy search algorithm with A* search algorithm and show complexity as well.
- Compare the working and complexities of BFS and DFS
- Discuss different data structures such as array and list, stack and queue, linked lists, tree, graph and hash.
- Contrast databases with file systems.
- Differentiate between SQL and NoSQL database management systems.
- Understand what software engineering is and why it is important.

- Apply important agile development practices and principles such as customer involvement, Incremental planning and delivery, continuous integration, refactoring, pair programming, small releases and test-first development.
- Make use of basic Git workflows [including issue tracking] to coordinate parallel development on a code base and to maintain the quality of code scheduled for release.
- Describe the advantages of a containerized software development & deployment; Use Docker engine features necessary for running containerized applications.

In this course, students will be working on python projects

-

Immediate Job market Outcomes

- Machine Learning Intern

Course Prerequisites

- No any prerequisites required.

Target Audience

Anyone without Programming and CS background.

Guided Hours

12 weeks course maximum of 4 hours per week in class.

Modules	**Class Time(hrs)**	**Online Learning (hrs)**
Introduction to the Course	0.75	
Basics of CS	6.25	
Python Programming	15.25	
Data Structure and Algorithms Analysis	10	
Database	4	
Building Applications	11.75	
Total	48	

Distinct Features Used in the Syllabus

Bold Outcomes refers to **Must Have** learning outcomes (Bare Minimum Criteria),

Normal Text refers to Should Have learning outcomes, Orange refers to Good to Have or Higher Level Learning Outcomes for high achieving students.

[P], [P: Numpy] refers to Implementation Learning Outcomes (additional details eg: using Numpy tool)

Red Outcomes refers to kept in the course but optional Outcomes completely ignorable.

Chapter Number #.#.#, Chapter Name	Learning Objectives follows Students should be able to: A. Answer this question Can Include additional contents that are available in platform or for teaching	Resources for reference Also include guide for teaching the particular content

Assessment

The Assessment will be based on the cognitive domain of Bloom's Taxonomy to classify learning objectives into different levels of complexity and specificity, viz. Remember, Understand, Apply, Analyse, Evaluate, Create

Assessment Objectives	**Categories**	**Objective**	**Action Words**
AO1	**Remember**	Recall Facts and basic concepts	Define, list, memorise, repeat, state, recognize
AO2	**Understand**	Explain Ideas or Concepts	Classify, describe, discuss, explain, identify, locate, recognize, report, select, translate, interpret, exemplify,
AO3	**Apply**	Use Information in a new situation, problem-solving and programming skills	Execute, implement, solve, use, demonstrate, interpret, operate, schedule, sketch
AO4	**Analyse**	Draw connections among ideas	Differentiate, organise, relate, compare, contrast, distinguish, examine, experiment, question, test
AO5	**Evaluate**	Justify a stand or decision, Choose between different methods or options	Check, Appraise, argue, defend, judge, select, support, value, critique, weigh
AO6	**Create**	Produce new or original Work, work on a whole project in original manner	Design, assemble, construct, conjecture, develop, generate, plan, produce, formulate, investigate

Source: https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/

Weight Distribution on

Components	**AO1**	**AO2**	**AO3**	**AO4**	**AO5**	**AO6**	**Weights**
Quizzes	20%	20%	20%	20%	10%	-	
Programming Assignment	10%	20%	20%	20%	10%	10%	
Projects	-	-	10%	20%	20%	50%	
Classroom Assessment							

Course Contents

By the end of this course, students will be able to:-

- Apply job as a software developer
- Describe different Computer Science Terminology as listed in the index
- Take Foundations in AI Maths class and Microdegree in ML

Module 1. Introduction to the Course

This course should include basics of programming with concepts on fundamental data types and structures, followed by different data structures and algorithms. Some basics of computer architecture, understand how CPU processes data. After that, students should understand the fundamental pipeline in software development. Then students should be able to learn about os and operations in os, followed by database and networking.

1.1. Introduction to the Course

Students should be able to :

1.1.1. Introduction to the Course	**A. Define Artificial intelligence in a broader sense and provide some examples** B. State what are the key foundational concepts important for learning AI a. Python, Data-structures and algorithms b. Mathematics required. **C. State what are the possible career tracks students can choose after completing this course.** Also Includes:- **1. Course Structure** **2. Road Map**	
1.1.2. Course Logistics	A. Plan and schedule their learning based on the course load, assessment and other criterias provided in this unit Also Includes:- **1. Course Load in Module Level** **2. Assessment and Evaluation Criteria** **3. Honour code and Violation Policy** This chapter should also explain how blended course will work and **set clear expectations from students**	

1.2. Introduction to AI

Students should be able to :

1.2.1. AI and its applications	A. Define Artificial Intelligence **B. Recognise applications of AI in different domains**	

	such as data-driven applications, Computer Vision, NLP C. Discuss the future opportunities and possibilities with AI	
1.2.2. A Brief History of AI	A. Discuss the origination history of AI including ideas a. Turing Test, b. AI Winter, c. Deep Learning Boom	
1.2.3. Types of AI	A. Define artificial intelligence into four categories and its current interpretation a. Thinking-Humanly b. Thinking-Rationally c. Acting-Humanly d. Acting-Rationally B. Define Different levels of Intelligence: a. Narrow, b. General, and c. Super Intelligence C. Recognise and understand the different methods used in AI such as search agents, symbolic AI or knowledge-based agents, Machine Learning, expert systems, other problems.	
1.2.4. Introduction to ML and DL	**A. Define Machine Learning** **B. Define** **a. Supervised,** **b. Unsupervised and** **c. Reinforcement learning** C. Define Deep Learning D. Differentiate between AI, ML, DL and data science	

Module 2. Basics of Computer Systems

2.1. Introduction to the Module

Students should be able to :

2.1.1. Introduction to the Module	A. Explain the importance of computer systems in any kind of software development Also includes 1. Unit Structure	

2.2. Digital information and Logic

Should understand boolean logics, gates, flip-flops, multiplexers, encoder, decoder, counter, state-machines. Optional: Complement system.

Students should be able to :

2.2.1. Number Representation in computers	A. Describe different number systems used to represent data in the computer system viz. binary, denary and hexadecimal number system B. convert a number from one number system to another C. Represent character data in its internal binary form depending on the character set used (ASCII or Unicode) D. Explain floating-point number representation	
2.2.2. Digital Information	**A.** Compare Digital and Analogue Signals linked to real sensor data B. Explain how bitmapped image data is represented in the computer system and the associated terminologies: pixel, image resolution, file header, file formats C. Explain how sound is represented and encoded along with the associated terminologies: sampling, sampling rate, sampling resolution, file formats D. Estimate the file size for uncompressed bitmap images and sound file given the appropriate parameters E. Explain how videos (streams) are represented along with the associated terminologies: frame rate, interlaced and progressive encoding, file formats	
2.2.3. Boolean Algebra	A. Explain the importance of boolean algebra B. Recall and use the following logic gates (and symbols) based on Truth tables: AND, OR, NOT, NAND, NOR, XOR to solve logic problems C. Recall and apply basic laws of Boolean Algebra including DeMorgan’s Law to simplify logical expressions D. Analyse the validity of logical propositions E. Discuss the importance of logic expression in logic-based AI [First order logic]	

2.3. Basics of Computer Organisation and Architecture

Instruction sets, Logical Unit, Arithmetic Unit, Memory, CPU, Memory Allocation, Architecture (RISC vs CISC).

Students should be able to :

2.3.1. Introduction to Computer Organization and Architecture	A. Describe structural and functional overview of the computer with reference to the Von-Neumann Architecture. B. Distinguish between computer organization and computer architecture	
2.3.2. Processing Unit and Instruction Set	A. Explain the functionality of three main components of a processor: The Control Unit, The Arithmetic And Logic Unit and Registers. B. Discuss the importance of Instruction Set Architecture with essential elements and characters of machine instructions. C. Explain the basic functionality of general types of instructions(Data Transfer, Data Manipulation and Program Control Instructions) used in microprocessors with some examples.	
2.3.3. Bus Organization	A. Describe the concept of interconnection within a computer system B. Explain the role of Data, Address and Control Bus	
2.3.4. Memory and I/O Unit	A. Describe the importance of memory hierarchy in Computer organization B. List various storage units. C. Describe briefly about the cache memory, primary memory and secondary memory D. Explain the basic function and use of I/O modules. E. Memory and I/O bottlenecks	
2.3.5. Parallel Architecture, Flynn's Taxonomy	A. Describe the limitations of Single Processor Architecture and why parallel architectures are needed B. Describe Flynn’s taxonomy: SISD, SIMD, MISD and MIMD C. Case Study of SIMD Architecture: GPU/applicability in ML	

2.4. Introduction to Linux Operating System

Shell scripting, Pipe, thread, deadlock, filesystem, groups, scheduling, IO handling.

Students should be able to:

2.4.1. Introduction to Operating System	A. Describe Operating System and its main functionalities B. Explain about OS kernels and the architecture of monolithic, micro and hybrid kernels.	
2.4.2. Introduction to Linux	A. Discuss the usefulness and popularity of linux with availability of different distros of linux B. Describe common Linux Terminologies: APT, sudo, repository, packages, dependencies, Terminal C. Get help through the command line using: echo, man, --help commands	
2.4.3. Linux File System	**A. Understand Linux file system organization** **B. Navigate and manipulate file system using:** ls, pwd, cd**,** cp, mv, rm, mkdir, chmod, rmdir, touch C. Search and edit text files using: grep, sed, gedit, cat, nano, less, head, tail and wildcard symbols D. Execute multiple commands using ; and && operators E. Build pipelines using |(pipe) operator F. Use VIM and basic Shell Scripting	
2.4.4. Process Control and Memory Management	A. Describe the Process Life cycle and data structure of a process B. Distinguish between threads and process C. Explain the following terminologies: multithreading, multiprocessing, multi-core, multi-CPU, and hyperthreading D. Examine the system and control the system process: ps, df, free, top, uname, kill, pkill E. Use Jobs control commands: jobs, fg, bg F. Describe address space and Virtual memory G. Monitor system resources using top, htop command	
2.4.5. Virtual Machine	A. Justify the need of virtualization B. Describe the concept of OS virtualization C. Distinguish between Virtual Machine and Container	

2.5. Fundamentals of Computer Networks

OSI, TCP/IP, SSL, TSL, HTTP, HTTPs, UDP, TCP, IP addressing, DH algorithm, RSA, SHA, SMTP, POP, IMAP

Students should be able to :

2.5.1. Introduction to Computer Network and Networking models	A. explain how the World Wide Web (WWW) and the Internet works B. explain the client-server and P2P model of networked computers and provide examples of applications	
2.5.2. Protocols and standards: TCP/IP	A. Distinguish between protocols and standards B. State 7 layers of OSI, and relate it with TCP/IP suite C. Describe how TCP/IP Protocol is used to interconnect network devices on the internet D. Describe the format of an IP address and how an IP address is used to associate a device on the network E. explain how a resource is uniquely identified on the World Wide Web (WWW) with reference to URL, URI, URN F. Use common networking tools: ping, traceroute, nslookup, ifconfig,	
2.5.3. Network Security	A. Describe the properties of secure communication B. Describe the basic principles and terminologies of cryptography C. Describe how encryption and decryption is carried using symmetric and public key D. Describe Authentications and HTTPS and its importance E. Use Secure Shell (ssh) to connect to a remote computer	
2.5.4. Cloud Computing [Class]	A. Explain the need of distributed computing. B. Describe what is meant by cloud computing and discuss reasons for its popularity. C. Describe resource scalability concept and distinguish between vertical scaling and horizontal scaling. D. Describe various service models of the Cloud E. Discuss on various IAAS service providers like AWZ, Azure, GCP etc.	

2.6. Module Summary

Students should be able to:

2.6.1. Module Summary	*A.* Summarise the contents covered in the module B. Relate the contents covered in this module to AI development *C.* Relate the contents covered in this module with other topics covered in the course	

Module 3. Python Programming

- Students should be able to code in python given their conceptual solution (i.e pseudocode).
- Students should be able to look through the documentation or docstrings to be able to use a particular function in different libraries such as in numpy, Pandas, sklearn, matplotlib.

3.1. Introduction to the Module

Students should be able to :

3.1.1. Introduction to the Module	A. Explain why Python is most popularly use for AI applications B. Describe python as a. Interpreted Language distinct from compiled language b. Dynamic Typed Language C. Use common statements in python such as arithmetic calculations and print	

3.2. Python Programming

Zen of Python, objects, data types, data structures. collections, abcs, functions conditional branching, loops,

Students should be able to :

3.2.1. Variables	A. Discuss the importance of variables B. Create and Assign values to variables, understanding variable name, operator and object) C. Recognise invalid variable names and suggest valid variable names D. Follow pep8 and words_separated_by_underscore convention E. Write basic expressions and operations with variables	
3.2.2. Range, Lists and Tuples	A. Define list from the need of sequential data structure point of view B. State the important characteristics of Range, Lists and Tuples C. demonstrate Slicing techniques a. left to right and right to left, b. negative indexing, c. stepping D. define and manipulate range, list and tuple E. Use method of list and tuples F. Explain difference between Lists and Tuples, with the concept of mutability G. Demonstrate different use-cases of membership test	
3.2.3. Dictionaries and Sets	A. Demonstrate use of `dict()` construct to create dictionaries B. Understand Important characteristics of DIctionaries and Sets	

	C. Learn how to define them and how to manipulate them D. Use methods of Dictionaries and Sets E. Explain difference between Dictionaries and Sets	
3.2.4. Control Flow Statements	A. Write if, else, elif conditional statements B. Use for loop through an iterator: List, dictionary, range etc C. Use while loop D. Use break and continue statements E. Apply nested control flow statements	
3.2.5. Functions	A. Define functions, B. Learn type of arguments to function (positional argument: args and keyword argument: kwargs) C. Know about variable scope (local, global and non local)	
3.2.6. **Iterators and** Generators	A. Clarify difference between iterable and iterator B. Know the iterator protocols (iter, next) and Iterator implementation as class (using __iter__, __next__) C. Simplifie creation of iterators: Generator D. Use of yield statement E. Learn to create generator expressions	
3.2.7. Lambdas	A. Learn about anonymous function and lambda B. Compare lambdas with regular function objects C. Know when to use and avoid lambda function	
3.2.8. Functional Programming: Maps/Filter	A. Learn functional programming concept B. Know use and importance of functional programming C. Learn about higher order function D. Use built-in higher order functions like map and filter	
3.2.9. List Comprehension	A. Know the importance of comprehension to make code more pythonic B. Use of list comprehension to improve the performance speed C. Motivate to use comprehension replacing map and filter where possible	
3.2.10. File handling	A. State different type of files that python can handle B. describe different mode of opening file (read, write, append) C. Use various operations of file handling(read, readline, readlines, wrtire, seek tell e.t.c)	
Assignment	Simple Gradient Descent	

3.3. Object-Oriented Programming

overloading, initialization, abstraction, composition, inheritance,

Students should be able to :

3.3.1. Object-Oriented Design	A. Distinguish between modular programming and object oriented programming B. State the major features of object oriented Paradigms C. Analyze object oriented design using Class Diagrams.	
3.3.2. OOP in Python	A. Describe the components of OOP: Object, class, attributes and methods B. Distinguish between object and class C. Initialize a classes using __init__ method D. Discuss the importance of self object. E. Use of classmethod and staticmethod decorators F. Learn data encapsulation and its importance	
3.3.3. Inheritance	A. Understand the concept of Code reusability B. Add methods and properties to existing class C. Distinguish between overriding and changing:use of super() method D. Types of inheritance (single, multiple and multilevel) E. Motivate to know about __mro__ (method resolution order) F. Appreciate the importance of DRY (Don’t repeat yourself) principle	
3.3.4. Polymorphism	A. Scope and signature of function B. Use of polymorphism in class methods. C. Use of polymorphism with: inheritance, function and object	
3.3.5. Operator Overloading	A. Learn about built-in dunder methods in python B. Appreciate the importance of overloading with use cases	
3.3.6. Exceptions Handling	A. Familiar with try, except, else and finally block B. Learn about different built-in Exceptions and its use cases C. Know about Generic Exception D. Use of “raise” keyword for raising error explicitly E. Learn to build custom exception handling	
3.3.7. Packages	A. Structure modules appropriately B. Manage absolute and relative paths	
3.3.8. Standard Library examples: Math, Datetime,	A. Learn importation of libraries B. Manipulating mathematical operation using math library C. Learn to parse and format the datetime/string D. Use of UTC datetime and use of timezones	
3.3.9. Strings and regular expressions	A. Different string literals (‘’, “”, ‘’’ ‘’’, “””” ‘’’) for single and multiline string B. Know Indexing, slicing and striding in string C. Learn different methods of strings (lower, upper, format etc..) D. Learn string concatenation and and formatting	

	E. Appreciate the importance of regex F. Use of re module G. Be familiar with MetaCharacters used in regex ([] . ^ $ * + ? {} () \ |) H. Learn to build regex patterns I. Know different methods available in regex (findall, split, sub, search etc...)	
3.3.10. Debugging	A. Appreciate the motivation of using debugging tools of like pdb B. Know to print a variable’s value using pdb C. Be familiar with python debugger commands	
3.3.11. OOP Design Principles	A. Recall and apply DRY B. SOLID Principle C. Import abc and create abstract class	

3.4. Numpy

Basic matrix manipulation, range, how numpy copies stuff, data types in numpy, indexing and slicing, array creation.

Students should be able to :

3.4.1. Introduction to Numpy	A. Justy the importance of computational library like Numpy B. Compare the differences between a normal Python list and numpy arrays in terms of memory management (how the memory is structured) and homogeneity (of elements) C. Learn about the different data-types available: float64, int64, etc. *(Don't have to go into detail or list everything. Can link to doc)* D. Print and use numpy array attributes: shape, size, dtype, ndim when required E. Create numpy arrays using different methods: np.arrays(...), np.ones, np.zeros, np.random.rand, np.random.randint, np.arange, np.linspace, np.full F. Use of np.random.seed to generate deterministic random numbers for experiments	
3.4.2. Indexing, Slicing and reshaping	A. Use basic indexing and slicing (start:end:step) from Python including negative indexing and the right to left sequence B. Use tuple based indexing and slicing. *Eg: (<row-range>, <column-range>) for slicing 2d array* C. Index a numpy array using int-array and boolean-array D. Reshape arrays: 2D to 1D and 3D to 2D. Also, pass -1 as a shape parameter. E. Use np.ravel	
3.4.3. Matrix Operations and Broadcasting	A. Multiply matrices using np.dot B. Perform element-wise operations on numpy arrays. C. Perform aggregation: sum, max, argmax. Also by specifying the axis. D. Know about np.transpose and the \`T\` property for finding transposes E. Use np.linalg.inv and np.linalg.det F. Describe the concept of broadcasting and its importance in numpy	
3.4.4. Numpy for Faster Computation	A. discuss the importance of measurement of code performance for code optimisation B. Use %timeit and %time magic commands as ways to profile timings of a program C. Use time.time function in profiling D. Use python profiler E. Use decorator using the line profiler F. Find and spot algorithmic changes that can speed up execution G. Use vectorized operations and broadcasting with Numpy. H. State options available: cython, numba.	
Solving Linear Systems	A. Use np.linalg.solve to solve a system of linear equations	

Unit Assignment	Simple linear regression	

3.5. Matplotlib

Students should be able to :

3.5.1. Components of Plots	A. ! Explain the object hierarchy in matplotlib, i.e figure holds Axes, Axes hold everything else. B. ! appreciate and use the Matplotlib backend such as agg, C. ! Use components of a Plot Title, Axes ticks/ name, Figure size, Color, Marker, legend using a line plot D. Learn about the two APIs of Matplotlib E. Use commands to save figures using plt.savefig	
3.5.2. Line Chart, Scatter Plots, Histograms, Images	A. Plot histograms, line charts, scatter plots, and images using imshow. B. Determine, at a high level, when to use what kind of plots	
3.5.3. SubPlots	A. Plot subplots in different arrangements	
Unit Assignment	Subplots	

3.6. Pandas

Data manipulation using pandas, idea behind dataframe and series, indexing, slicing, merging, join, filter, grouping, view,

Students should be able to :

3.6.1. Creating DataFrame and Series	A. Know the motivation and usage of a library like pandas B. Know about the main constructs in pandas: Series and DataFrame; what they represent C. See Series as an enhanced version of numpy array D. Know about the \`name\`, and \`index\` properties of \`Series\`. For \`DataFrame\`, additionally: \`shape\`, \`info()\`, \`describe()\`. E. Load datasets a. Use nrows to load only load a chunk of data F. Basic ways to access the data in a Series: numerical indexing/slicing and indexing/slicing using the actual index. G. Be familiar with the common ways to create DataFrames: from a dictionary, from lists, by reading a csv.	
3.6.2. Selecting Rows and Columns	A. Access data using the df[columns] approach as well as iloc, loc, iat and at. B. Know the differences between \`iloc\` vs \`loc\` and when to use which C. Know the differences between \`loc\` and \`at\` and when to use which D. Use boolean indices to select rows. Eg: df[boolean_condition]. Also, with multiple boolean conditions.	

3.6.3. Groupby	A. Understand the split-apply-combine computational construct that is the normal usage of groupby B. Group data by certain attributes and perform computations on the groupby object: sum, count, max. C. Learn about the use of the `aggregate` method.	
3.6.4. Apply, Join and Merge	A. Perform left, right, inner, and outer join using `join` and `merge`. B. Combine DataFrames using `concat` C. Apply methods to the data using `apply` and `map`. D. Handle missing data using `isna`, `dropna` and `fillna`	
3.6.5. Pandas Plotting	A. Use the plotting functionalities in pandas to create common plots: line plots, histograms, heatmaps, scatterplots. B. Correlation matrix and plot	
Unit Assignment: Data Preprocessing	- Merging data from different Data sources - Data Cleaning - Data Transformations - Data Visualisation	

3.7. Web scraping

Students should be able to :

3.7.1. Request Module Downloading files from the web	A. Use request module to send HTTP Request B. describe different types of HTTP requests C. Understand the response code of the sent request D. create API queries with parameters and authorization	
3.7.2. Introduction to Beautiful soup and parsing HTML with beautiful soup	A. Parse HTML and XML files with Beautiful Soup B. Navigate through the contents of the parsed file C. Create datasets/data frames by extracting contents from table, paragraphs	
3.7.3. Finding elements of a web page	A. Use requests and beautiful soup module to get an HTML Page B. Inspect the source of a page to get required information.	
Unit Assignment	Data Scraping	

3.8. Module Summary

Students should be able to :

3.8.1. Module Summary	*A.* Summarise the contents covered in the module B. Relate the contents covered in this module to AI development *C.* Relate the contents covered in this module with other topics covered in the course	
Module Project	Scraping a Data then applying linear regression on the data	

Module 4. Data Structures and Algorithms

4.1. Introduction to the Module

Students should be able to :

4.1.1. Introduction to the Module	A. Understand the motivation to learn the content in the module B. Understand the flow of the contents in the module (overview)	

4.2. Algorithm Analysis

Space complexity, Time complexity, worst case O(n), best case(Theta(n)), average case, lower bound W(n), upper bound.

Students should be able to:

4.2.1. Introduction to algorithmic analysis	A. Understand what algorithms are. B. Understand that algorithms are language agnostic. C. Understand time and space complexities.	
4.2.2. Asymptotic analysis	A. Understand best case and worst case scenarios for analysing algorithms. B. Understand the Big-O, Big-Omega and Big Theta Notations. C. Calculate Big-O notation for a given algorithm.	
4.2.3. Algorithm design strategies	A. Understand the different algorithm design strategies. B. Understand Brute force, Divide and conquer, and Greedy strategies. C. Visualise an example problem on all design strategies.	

4.3. Data Structure

Students should be able to :

4.3.1. Array and Lists	A. Understand array as an ordered list, and its importance. B. Distinguish between Array and Python List	
4.3.2. Stack and Queue	A. Understand Stack as a LIFO B. Implement custom Stack data structure that support push, pop and peek operation. C. Understand Queue as FIFO data structure. D. Implement custom Queue data structure that supports *enqueue* and *dequeue* operation	
4.3.3. Linked Lists	A. Understand the concept of Nodes. B. Understand Linked List as a connection of Nodes. C. Understand how elements are added and removed from Singly Linked Lists D. Understand its advantages and disadvantages over Arrays. E. Implement custom singly linked list data structure to add, remove and check if element exist in linked list or not.	

4.3.4. Doubly Linked Lists	A. Understand Doubly Linked List B. Understand how elements are added and removed from Doubly Linked Lists C. Understand advantages and disadvantages over Singly Linked List D. Implement custom doubly linked list data structure to add, remove and check if element exist in doubly linked list or not.	
4.3.5. Tree	A. Understand what makes up a tree (nodes and edges) B. Understand the terminology associated with tree i.e depth, height, root node, leaf node, parent, child…. C. Understand binary and ordered trees. D. Introduced to tree traversal techniques [Pre order, Post order and inorder] E. Implement Tree [Only in Reading Materials]	
4.3.6. Graphs	A. Understand directed and undirected graphs. B. Understand weighted and unweighted graphs. C. Understand how graphs differ from trees D. Understand the Adjacency matrix E. Introduced to graph traversal techniques F. Implement Graphs [Only in Reading Materials]	
4.3.7. Hash	A. Understand how hash tables and hash functions work. B. Visualize hashing with examples. C. Understand how hashing makes finding data faster. D. Suggest other uses of Hashing (Eg. data integrity verification) (SHA, MD5) E. Understand how hashing conflicts are resolved.	

4.4. Algorithm

Sorting, graph traversal, searching, dynamic programming, greedy algorithms

Students should be able to :

4.4.1. Recursion	A. Understand what recursion is, when it can be used and how it differs from iteration through a coding example. B. Understand it's advantages and disadvantages over iteration. C. Write a recursion strategy to solve a problem [Assignment] D. use Stack to trace a recursive Function	Fast Fibonacci algorithms
4.4.2. Sorting Algorithms	A. Understand the sorting algorithms and code up a. Bubble Sort [Video] b. Merge Sort [Video] c. Insertion Sort [Assignment] B. Mention there exist other sorting algorithm too and	

	compare their complexity.	
4.4.3. Searching Algorithms	A. Introduced to linear and binary search(divide & conquer) B. Understand the complexities of both.	
4.4.4. Uninformed Search Agents: BFS, DFS	A. Understand uninformed search strategies B. Compare the working and complexities of BFS and DFS C. Show an example to find a node in a tree using BFS and DFS. [Without coding] D. Implement of BFS and DFS [Only in Reading materials]	
4.4.5. Informed Search Agents: Greedy Search, A* Search	A. Understand Informed Search Agent B. Compare Greedy search algorithm with A* Search Algorithm. Show complexity as well. C. Show an example to find a node in a graph using Greedy search and A* search. [Without coding] D. Implementation of Greedy and A* Search [Only in Reading materials]	
AI Application	informed/uninformed Search Problem, solution with BFS/DFS/A* Search	

4.5. Code Optimisation

Students should be able to :

4.5.1. Optimization of code	● Understand the need of code optimization. ● Understand how code optimization is achieved using faster programming languages and parallelization,
4.5.2. Code Profiling	●
4.5.3. Writing faster code	● Optimisation of Algorithms ● Understand and use the common techniques to write faster numerical code
	● Explain when to sacrifice runtime for memory usage and when to sacrifice memory usage for faster runtime ●

4.6. Module Summary

Students should be able to :

4.6.1. Module Summary	*A.* Summarise the contents covered in the module B. Relate the contents covered in this module to AI development *C.* Relate the contents covered in this module with other topics covered in the course	
Module Project		

Module 5. Database

5.1. Introduction to the Module

Students should be able to :

5.1.1. Introduction to the module	A. Understand the motivation to learn the content in the module B. Understand the flow of the contents in the module (overview)	
5.1.2. Database and DBMS	A. Understand what is Database. Contrast it with the file system. B. Appreciate the importance of DBMS C. Understand types of DBMS: a. SQL b. NoSQL	

5.2. SQL

Select, join, where, view, trigger.

Students should be able to :

5.2.1. RDBMS Concepts	A. Understand the relational model of data apprehending the basic concepts such as a. the structure of relational databases(tables), b. database schemas, c. keys(candidate key, superkey, primary key, foreign key), d. schema diagrams and e. relational query languages
5.2.2. Introduction to SQL	A. Define, delete and modify relations using commands provided for Data Definition B. Insert, Select, Update and DeleteTuples using commands provided for Data Manipulation. C. specify integrity constraints such as primary-key constraints and foreign-key constraints
5.2.3. SQL: Queries and Subqueries	A. Execute query involving basic set operations on relations: union, intersection and except(difference) B. Understand several types of joins including inner and outer joins and several types of join conditions. C. Use aggregate functions: avg, min, max, sum, count D. Execute query containing joins and aggregate functions by adding “group by” and “having” clauses E. Understand about the different constraints: Integrity constraints, Domain constraints, Unique constraint F. Execute nested subqueries in the “where” and “from” clauses of an outer query

5.3. NoSQL

Students should be able to :

5.3.1. Introduction to NoSQL	A. Distinguish between structured and unstructured data B. Understand the problems associated with Relational Database Model. C. Describe the main terminologies of NoSQL databases: Basic Availability, Soft State, Eventual Consistency(BaSE properties), sharding and CAP Theorem D. Compare SQL and NoSQL with advantages and disadvantages. E. List out the various types of NoSQL databases based on their data model.
5.3.2. Introduction to MongoDB	A. Understand the importance of document databases B. Perform CRUD operations in MongoDB
5.3.3. Queries and Subqueries: MongoDB	A. Execute query involving basic set operations, types of joins aggregate functions with “group by” and “having” clauses *[Show how is done in MongoDB if possible for the cases that we have covered in* ***SQL: Queries and Subqueries****]*

5.4. Module Summary

Students should be able to :

5.4.1. Module Summary	*A.* Summarise the contents covered in the module B. Relate the contents covered in this module to AI development *C.* Relate the contents covered in this module with other topics covered in the course

Module 6. Application Development

6.1. Introduction to the Module

Students should be able to :

6.1.1. Introduction to the Module	A. Understand the motivation to learn the content in the module B. Understand the flow of the contents in the module (overview)	

6.2. Software Development Life Cycle

Feasibility analysis, requirement analysis, use cases, testing, integration, classical methods, agile methods, documentation.

Students should be able to :

6.2.1. Software Development Life Cycle	A. Understand what software engineering is and why it is important B. Understand that the development of different types of software system may require different software engineering techniques C. Understand general software process models and when they might be used [Waterfall, iterative, prototype, agile….] D. Describe the fundamental process activities of software: a. requirements engineering(analysis and design) b. software development c. quality assurance, and d. evolution(Operation and Maintenance);	
6.2.2. Agile Methodology	A. explain the rationale for agile software development methods B. Differentiate between plan-driven [Waterfall model] and agile development. C. Know about important agile development practices and principles such as customer involvement, Incremental planning and delivery, continuous integration, refactoring, pair programming, small releases and test-first development. D. Frameworks of Agile: a. Scrum b. Kanban	
6.2.3. Documentation	A. Appreciate the importance of documentation B. Describe different types of documentation formats used in python a. Numpy style b. Google style	
6.2.4. Software Testing	A. differentiate verification and validation B. Understand the three stages of development testing with its importance:	

	a. Unit Testing, b. Component Test(Integration Test), c. System Test C. Understand the types and importance of user testing D. Use pytest to write and run test cases E. Use circleCI to show the continuous integration example.	
6.2.5. Version Control	A. Describe the essential functionality that should be provided by a version control system, and how it is realized in centralized(subversion) and distributed version control systems(git); B. Make use of basic Git workflows [including issue tracking] to coordinate parallel development on a code base and to maintain the quality of code scheduled for release.	

6.3. Web Frameworks

This Module allows students to develop web documents and web applications.

Students should be able to :

6.3.1. Web Application Basics	A. Describe how a web browser works for retrieving, presenting, and traversing information on WWW B. Distinguish between static and dynamic web pages C. Describe how a web server handles a request D. Compare frontend and backend E. Describe Model View Controller, Model View Template design pattern of different web frameworks	
6.3.2. HTML	A. markup, styling and interactiveness B. Understand Basic Principle of HTML, Dynamic HTML C. Get acquainted different HTML tags and attributes	
6.3.3. CSS	A. Use different selectors and properties; CSS Syntax B. Apply in-line, internal and external CSS styles to HTML C. Understand basic Box model layout D. Introduce and Use bootstrap	
6.3.4. JS	A. Describe Document Object Model(DOM) and its elements B. Implement simple functions in JS to trigger an event	
6.3.5. Rest API	A. Describe REST API B. Describe REST API valid methods: GET, PUT, POST, DELETE, PATCH C. Explain the architectural constraints: a. Client Server b. Statelessness c. Uniform Interface d. Caching e. Code On Demand	
6.3.6. Flask	A. Basic terminology associated with Flask. B. Show CRUD operation in Flask. C. Develop a RESTful API in Flask.	

6.4. Deployment

Students should be able to :

6.4.1. Virtual Environment	A. Understand the need of virtual environment B. Create virtual environment: [venv and conda] C. Activate and deactivate virtual environment [venv and conda]	
6.4.2. Docker	A. Describe the advantages of a containerized software development & deployment B. Install docker in Linux C. Build and manage docker containers D. Use Docker engine features necessary for running containerized applications	
6.4.3. Deployment	A. Understand basic workflow during docker deployment in aws.	

6.5. Module Summary

Students should be able to :

6.5.1. Module Summary	*A.* Summarise the contents covered in the module B. Relate the contents covered in this module to AI development *C.* Relate the contents covered in this module with other topics covered in the course	

Glossary of command words

This glossary table below is a guide to develop learning outcomes for each chapter that would help to develop, teach and learn content as well as for evaluation.

Command words	**AO**	**Indication (What it means)**	**Use Cases**
Define,	1	Formal statement/phrases of definition	equations, terms
What is meant by	1	Definition + Significance or context of the term	
Describe	2	State the main points, with diagrams, examples or so on	Phenomena, experiment, observation
Explain	2	Reasoning with reference of theory applied	Relationship, cause-effects,
State	1	Express in concise words, without supporting arguments	Theorem, law, fact, value without calculation
List	1	List down number of points without elaboration	
Exemplify	1,2	Provide examples	
Discuss	5	Critical account on the topic	critique,
Deduce, solve, Predict	4	Produce the answer through logical connections than just by recall	
Suggest, propose	4	applying knowledge to a new situation or when there is no unique idea	
Calculate, find out, workout, carry out	3	get a numerical from given data, some value, through some work	Workout to calculate a value
Determine	3	Quantity calculated with certainty	Magnitude, scale ,
Show	3	Derive result through a structured explicit evidence	
Justify, support	4	Support a case with evidence/arguments	
Verify, prove	4	Confirm that a given statement/result is true	
Estimate	4,5	Reasoned order of magnitude for the quantity	
Sketch	3,4	Make freehand sketch drawing/curve with key features	Diagrams, graphs, figures
Compare	4	Provide similarities and differences	
Recognise, identify, name, select	1	identify from having encountered them before	
Implement, code	3,6		

Create, design, construct,	6		
understand			
appreciate			