جامعة محمد بن زايد للذكاء الاصطناعي
MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Core Courses Syllabi

CS701 - Advanced Programming

Title	Advanced Programming
Code	CS701
Loading	4 Credit-hours
Prerequisites	Basic programming skills in any computer language
Catalog Description	This course provides a comprehensive introduction to advanced programming methods. It builds upon fundamental concepts in programming, and introduces object-oriented programming, parallel programming, data structures, algorithms, and many other programming and related tools.
Goal	This elective course aims to familiarize students with advanced methods in programming, including object-oriented programming, parallel programming, data structures, algorithms, and many other programming and related tools. This course aims to instill in students programming tools necessary to write better code and understanding of concepts necessary to improve and accelerate code performance.
Content	The course covers the following major modules: **(I)** Object-Oriented Programming, **(II)** Data Structures and Algorithms, **(III)** Parallel Programming, **(IV)** Computing Tools: GUI, Dockers, Version Control, Debugger, Regular Expressions.
Recommended Textbooks	1. Kenneth Lambert, *Fundamentals of Python: Data Structures*, 2nd edition, Cengage learning, 2019. 2. Mark Lutz, *Programming Python: Powerful Object-Oriented Programming*, 4th edition, O'Reilly Media, 2010. 3. Peter Wentworth et al. *How to Think Like a Computer Scientist: Learning with Python 3*, 2012
Recommended References & Supplemental Material	Instructors will provide reading material for additional topics covered in the course.

Teaching Week	Topics
1	**Object Oriented Programming** **Lecture** • Introduction to object-oriented programming • Object, class, inheritance, polymorphism, abstraction, encapsulation • How to organize a program using objects **Lab** • Instructor walk-through of a programming exercise to demonstrate OOP thinking
2	**Object Oriented Programming** **Lecture** • Design considerations • Polymorphism and working with different data types **Lab** • Instructor-led demonstration of inheritance and composition
3	**Object Oriented Programming** **Lecture** • Introduction to Design Patterns **Lab** • Instructor-led demonstration - refactor a program using a design pattern paradigm
4	**Data Structures and Algorithms** **Lecture** • Review of time and space complexity (big O) • Array, strings, linked list, stacks, queues **Lab** • Practice problems to calculate time and space complexity • Instructor-led demonstration related to linked-list
5	**Data Structures and Algorithms** **Lecture** • Tree and graph algorithms • Hashing **Lab** • Practice problems in using trees, graphs, and hashing.

Teaching Week	Topics
6	**Data Structures and Algorithms** **Lecture** • Recursion and Dynamic Programming • Sorting and searching **Lab** • Practice problems in recursion and dynamic programming
7	**Data Structures and Algorithms** **Lecture** • Advanced topics • Topological sort • Dijkstra's algorithm • Hash Table Collision Resolution • Rabin-Karp Substring Search **Lab** • Practice problems related to advanced topics
8	**Parallel Programming** **Lecture** • Introduction to parallel programming • Memory, threads, process, pool • Identifying a parallel problem **Lab** • Introduction to parallel programming tools for Python
9	**Parallel Programming** **Lecture** • Parallel programming models • Process interaction – shared memory and message passing • Synchronization, race conditions, locks **Lab** • Practice problems in parallel programming
10	**Parallel Programming** **Lecture** • Problem decomposition – task and data parallelism • MapReduce **Lab** • Practice problems in parallel programming

Teaching Week	Topics
11	**Parallel Programming** **Lecture** • Designing parallel programs • Load balancing, granularity • Case studies **Lab** • Practice problems in parallel programming
12	**Parallel Programming** **Lecture** • Introduction to GPU programming using PyCuda • Host vs device construct, memory arrangement • Cuda kernels • Devices and contexts • Concurrency and streams **Lab** • Instructor-led demo of GPU programs
13	**Parallel Programming** **Lecture** • Optimizing GPU programs • Identifying potential performance benefits with GPU programs **Lab** • Instructor-led demo of GPU programs
14	**Special Topics** **Lecture** • Introduction to GUI programming using QT • QT Signals and slots, widgets, buttons, scrollbar • GUI layout using QT, adding features **Lab** • Instructor-led demo of: 1) displaying images, videos, text, and graphs, 2) capturing user interaction
15	**Special Topics** **Lecture** • Regular expressions • Version control • Dockers • Debugger **Lab** • Practice problem on topics of the week