AING 214 - Advanced Programming Techniques for Artificial Intelligence

Department: Computer Engineering

Instructor Information

Name: Tansel Sarıhan	**E-mail:** tansel.sarihan@emu.edu.tr	**Office:** CMPE 202	**Group:** 01

Assistant Information

Name: TBA	**E-mail:** tba@emu.edu.tr	**Office:** CMPE__	**Group:** 01 (Coordinator)
Name: TBA	**E-mail:** tba@emu.edu.tr	**Office:** CMPE ___	**Group:** 00
Name: TBA	**E-mail:** tba@emu.edu.tr	**Office:** CMPE ___	**Group:** 00

Program Name: Artificial Intelligence Engineering	**Program Code:** 2L

Course Code	Credits	Year / Semester
AING 214	4	2025 – 2026 Spring

☒ Required Course ☐ Elective Course

Prerequisite(s)

AING 211 – Object-oriented Programming

Catalog Description

Introduction to functional programming, anonymous (lambda) functions, map/reduce/filter, list/dictionary and set comprehensions, iterables, iterators, generators. First class and higher-order functions. Decorator functions and their use-cases. Freezing funtion arguments for partial execution. Modular, reusable and scalable software development using encapsulation, multiple inheritance, abstraction, metaclasses and attribute management. Operator overloading for user-defined objects. Lightweight concurrent execution using multi-threading. Multi-core processing using parallel pogramming. Massive parallelism using GPU computing. Robust programming through exception-handling.

Course Web Page

https://staff.emu.edu.tr/tanselsarihan/en/teaching/aing-214-advanced-programming-techniques-for-artificial-intelligence

Textbook(s)

Ramalho, L. (2015). *Fluent Python: Clear, concise, and effective programming*. " O'Reilly Media, Inc."

Schedule and Classrooms

Tuesday	08:30 – 10:20	CMPE 236 (Lab)
Tuesday	14:30 – 16:20	CMPE 127
Thursday	10:30 – 12:20	CMPE 027

Topics Covered and Class Schedule (4 hours of lectures per week)	
Week	**Topic**
February 23 – 28	Introduction to Functional Programming
March 2 – 7	Anonymous (Lambda) Functions
March 9 – 14	Map, Reduce, Filter Functions
March 16 – 21	List, Dictionary and Set Comprehensions
March 23 – 28	Iterables, Iterators and Generators
March 30 – April 4	First Class and Higher-order Functions
April 6 – 11	Decorators
Midterm Exams (April 10 - 25)	
April 27 – May 2	Freezing Function Arguments for Partial Execution
May 4 – 9	Modular, Reusable, and Scalable Software Development with Functional Programming
May 11 – 16	Operator Overloading for User-Defined Objects
May 18 – May 23	Lightweight Concurrent Execution
May 25 – 30	No Lecture
Jun 1 – 6	Multi-core Processing and Parallel Programming
Jun 8 – 13	GPU Computing
Final Exams (Jun 15 – 27)	
Lab Schedule	
Week	**Topic**
March 9 – 14	Lambda Functions and Functional Programming
March 16 – 21	Map, Reduce, and Filter Functions
March 23 – 28	List, Dictionary, and Set Comprehensions
March 30 – April 4	Iterables, Iterators, and Generators
April 6 – 11	First-Class Functions, Higher-Order Functions, and Decorators
May 4 – 9	Freezing Function Arguments (Partial Execution) and Metaclasses
May 11 – 16	Operator Overloading, Multi-threading, Exception Handling
Jun 1 – 6	Computing with GPU Parallelism

Course Learning Outcomes

Upon successful completion of the course, students are expected to have the following competencies:

1. Solve functional programming problems using anonymous (lambda) functions
2. Apply map, reduce, and filter operations in appropriate scenarios
3. Perform data transformations using list, dictionary, and set comprehensions
4. Explain the differences between iterables, iterators, and generators, and write example code for each
5. Develop reusable components using first-class and higher-order functions
6. Write decorator functions and apply them in real-world use cases
7. Demonstrate freezing function arguments for partial execution (partial functions)
8. Design modular software using encapsulation, abstraction, and multiple inheritance
9. Apply metaclasses and attribute management mechanisms through simple examples
10. Implement operator overloading for user-defined objects
11. Develop robust programs using exception handling techniques
12. Write lightweight concurrent applications using multi-threading
13. Apply parallel programming approaches on multi-core systems
14. Explain GPU computing concepts and develop basic massively parallel applications
15. Design modular, reusable, and scalable architectures for large-scale software systems
16. Analyze the performance of parallel and concurrent code and propose optimization strategies

	Method	Number	Percentage
Assesment	Midterm Exam	1	40%
	Final Exam	1	44%
	Lab Works	8	16%
	Attendance	Every lecture	0%

Policy on Makeup

For eligibility to take a makeup exam, the student should bring (submit) a doctor's report ***within 3 working days*** of the missed exam. You will have only one make-up for midterm or final exams only. Make-up will be organized after final exam period and will cover all the materials covered during the semester.

Policy on the NG Grade

NG grade will be given in case of missing both midterm and final exams without official excuse.

Policy on Attendance

Attendance will be taken in every lecture but will not be graded.

Policy on Missed Labs

There will be ***no makeup*** for missed labs. If you cannot attend a lab for some reason, you should contact the assistant beforehand so that you can present your work in advance.

Policy on Cheating and Plagiarism

Any student caught cheating in exams or in any other graded course work will automatically fail from the course and may be sent to the disciplinary committee at the discretion of the instructor.

Relationship of the course to ABET Student Outcomes

The course has been designed to contribute to the following student outcomes:

1. an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics

Prepared by: Tansel Sarıhan	**Date:** 19 February 2026